AFRL-RI-RS-TR-2015-252

SIPHER: SCALABLE IMPLEMENTATION OF PRIMITIVES FOR HOMOMORPHIC ENCRYPTION

RAYTHEON BBN TECHNOLOGIES

NOVEMBER 2015

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Defense Advanced Research Projects Agency (DARPA) Public Release Center and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2015-252 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
CARL THOMAS
Work Unit Manager

/ S /
MARK LINDERMAN
Technical Advisor, Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
NOVEMBER 2015	FINAL TECHNICAL REPORT	JAN 2011 – FEB 2015

4. TITLE AND SUBTITLE

SIPHER: SCALABLE IMPLEMENTATION OF PRIMITIVES FOR HOMOMORPHIC ENCRYPTION

5a. CONTRACT NUMBER
FA8750-11-C-0098

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

David Cousins, Kurt Rohloff, Christopher Peikert, and Daniel Sumorok

5d. PROJECT NUMBER
SIPH

5e. TASK NUMBER
ER

5f. WORK UNIT NUMBER
RC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2015-252

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. DARPA DISTAR CASE # 25476
Date Cleared: 4 Nov 2015

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of SIPHER was to improve the efficiency and scalability of Fully Homomorphic Encryption (FHE). Significant improvements were made to optimize FHE processing, improve primitives (FHE basic algorithms) and produce a Field-Programmable Gate Array (FPGA) hardware as a FHE accelerator. Through the program FHE implementation speed was increased up by several orders of magnitude.

15. SUBJECT TERMS
Fully Homomorphic Encryption (FHE), Somewhat Homomorphic Encryption (SHE), Secure Multiparty Computation (SMC), Cryptosystem, Voice Over IP (VOIP), Keyword Search (KWS), Field Programmable Gate Array (FPGA)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**CARL R. THOMAS**
U	U	U	UU	402	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

III. LIST OF FIGURES

V. LIST OF TABLES

VII. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Drew Dean and Dr. John Launchbury of DARPA I2O for their continued support of this research, for their ongoing technical guidance and their shared vision for advancing the state of computational privacy in our world.

The authors would also like to acknowledge experimentation support received from Dave Archer and Thomas DuBuisson of Galois over the course of this project.

1 SUMMARY

Prior to the Proceed program, the main challenges preventing practical demonstrations and use of Fully Homomorphic Encryption (FHE) were efficiency and scalability. At the start of the Program, the state-of-the-art FHE implementations were both inefficient and not scalable. Our work in Scalable Implementation of Primitives for Homomorphic EncRyption (SIPHER) has brought FHE into the realm of practice, bringing several orders of magnitude runtime improvement, and resulting in FHE implementations that can be executed on single and multicore computers (including iPhones). Furthermore, our implementation of an FHE hardware accelerator on a Virtex 7 Field Programmable Gate Array (FPGA) can speed up core FHE functions by over three orders of magnitude.

Previous FHE schemes were inefficient because the underlying algorithms and their implementations take too long to run at an appropriate level of assured security. Similarly, these FHE schemes were not scalable because memory requirements for encrypting practical-length messages with a reasonable level of security exceed the abilities of highly parallel computation devices like FPGAs. These issues are driven by several factors:

- The very large keys required for an assured level of security and large expansion of unencrypted plaintext messages to encrypted ciphertext.
- The large computation depth needed for Bootstrapping/Recryption circuits (an efficiency bottleneck of FHE schemes).
- The lack of scalable and highly optimized implementations of basic modulus ring operations, which are building blocks used across many lattice FHE schemes.

Our activities culminated in many orders of magnitude improvement for these bottlenecks. We achieved this revolutionary improvement by significantly advancing the state of the art in a number of independent focus areas:

- Multiple foundational improvements in the underlying FHE scheme for more efficient and scalable implementations of FHE operations. These improvements include a new approach to FHE Recryption, and the use of modulus and ring reduction to limit ciphertext expansion.
- Parallelizable, efficient algorithm design for scalable implementations of basic computational primitives at the core of lattice FHE schemes improving runtime of all FHE operations.
- Advanced code development approach for efficient and flexible embedded and FPGA implementations.

Figure 1 shows the layered SIPHER approach. We provide software interfaces for our optimized basic FHE operations. This lets users construct general applications computing on encrypted data. Core lattice-based primitives form the heart of our FHE implementations. Our modular approach allowed us to: 1) construct and experimentally modify multiple implementations of FHE operations and 2) easily deploy code on FPGA hardware to run the primitives on cost-effective, massively parallel hardware, providing 3 orders of magnitude improvement in basic FHE operation runtimes.

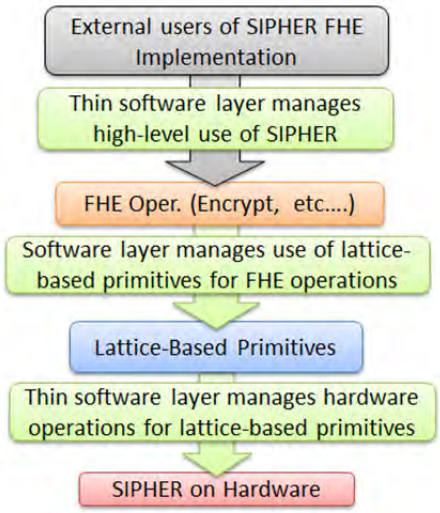


Figure 1: Layered SIPHER approach

2 INTRODUCTION

This report is organized as follows:

In Section 2 we introduce our work and describe our NTRU-based FHE cryptosystem that motivates our design choices in the SIPHER library. We introduce our two demonstration applications: Multiparty Voice over IP (VOIP) Teleconferencing and FHE based Keyword Search of Encrypted Documents. We finally introduce our work in FPGA hardware acceleration of FHE Primitives.

In Section 3 we discuss our Methods, Assumptions, and Procedures. Specifically we discuss our methodology of a three pronged research approach: theory, software design, and hardware acceleration. We then detail the SIPHER crypto system issues and developmental approaches from the theoretical, software, and hardware perspectives. We discuss the issues of FHE parameter selection for VOIP and Encrypted Keyword Search. Finally, we discuss approaches to parallelism for accelerating both our software and hardware implementations.

In section 4 we discuss actual implementation details for the SIPHER from the point of view of theory, software and hardware. We then present key experimental results, including basic SIPHER software library operation, VOIP and Key Word Search performance and Hardware Acceleration speedup.

Section 5 is a discussion of our insights and conclusions.

Section 6 contains our recommendations for future work for furthering the applicability of our FHE efforts towards wide spread practice.

2.1 Practical Somewhat and Fully Homomorphic Encryption (SHE and FHE)

Recent breakthroughs in Fully Homomorphic Encryption (FHE) have shown that it is theoretically possible to securely run arbitrary computations over encrypted data without decrypting the data [1], [2]. Known FHE schemes have several nice properties which make them very attractive techniques to consider addressing pressing cybersecurity issues. For one, FHE schemes are believed to be very secure. The security of FHE schemes are derived from the hardness of mathematical problems which are currently believed to be hard to solve even with quantum computing devices [3]. Known FHE schemes are consequently labelled as post-quantum, meaning that there are no known algorithms which are computationally efficient that can be used to practically break these schemes, even for execution on quantum computing devices. In addition to security, a practical FHE capability would be game-changing for cybersecurity researchers and practitioners. With practical FHE, sensitive data could be encrypted and placed into a low cost cloud computing environment for processing without having to share decryption keys. This could greatly reduce the operational costs of highly regulated industries such as medical, legal, financial and government industries where regulatory compliance restricts the ability to outsource computation to low cost cloud computing environments. Practical FHE could also greatly reduce the impact of insider attacks by greatly restricting who can access sensitive data within an organization, but still permitting processing of this data.

Despite the attractiveness of known FHE schemes, there are practical limitations that have prevented their broad practical use. As indicated in early implementation research [4], runtime is a major obstacle to be overcome before homomorphic encryption technologies become practical.

Solutions to FHE runtime challenges have been explored through several means, including by improving the theoretical efficiency of the underlying scheme [5]– [9], and by developing more efficient implementations of these schemes [10]– [16]. Despite these advances in FHE schemes and implementations, there have been little results on the application of these technologies. Many of the implementation-focused papers have used the homomorphic evaluation of the AES circuits as benchmarks [10], [13], [14]. Beyond this is [17] which uses the HELib library [15] to support statistical operations such as linear regression.

2.2 The Scalable Implementation of Primitives for Homomorphic EncRyption (SIPHER) Library for SHE and FHE

In this report we discuss our experience designing a general lattice encryption library called SIPHER (Scalable Implementation of Primitives for Homomorphic EncRyption) to support both limited depth computations for Somewhat Homomorphic Encryption (SHE) and full depth computations for FHE. We discuss using this library to support an encrypted end-to-end VoIP teleconferencing application on an embedded processor (iPhone). We also discuss using the same library to support Encrypted Keyword Search (EKS) on single and multicore Linux computers. Finally, we discuss accelerating the single core EKS with an attached FPGA based FHE accelerator, showing that a single Virtex 7 FPGA will execute FHE primitives 1600x faster than a single core and 35x faster than a 64 core system.

Unlike prior libraries, SIPHER is intended to be as adaptable and extensible as possible, with modular software architecture to support rapid prototyping of the library, easier integration of the library into a broader computing infrastructures and possessing increased parallelism. Our motivation with the SIPHER library is that as prior homomorphic encryption implementations have reduced absolute runtime, there has been limited attention paid to software engineering issues that need to be addressed for these libraries to be flexibly adapted to application contexts.

We implement in software specialized lattice primitives such as Ring Addition, Ring Multiplication and the Chinese Remainder Transform (CRT).We use our primitive implementations to construct the FHE operations of Key Generation (KeyGen), Encryption (Enc), Decryption (Dec), Evaluation Addition (EvalAdd), Evaluation Multiplication (EvalMult) and Bootstrapping (Boot). We use supporting Modulus Reduction (ModReduce), Ring Reduction (RingReduce) and Key Switching (KeySwitch) operations to augment the EvalMult operation and support larger depth computations before Bootstrapping or decreasing the security of our scheme. Finally, Recryption (Bootstrapping) is a function that will refresh a ciphertext that has previously been operated on, in order to enable further processing. These primitives are shown schematically in Figure 2.

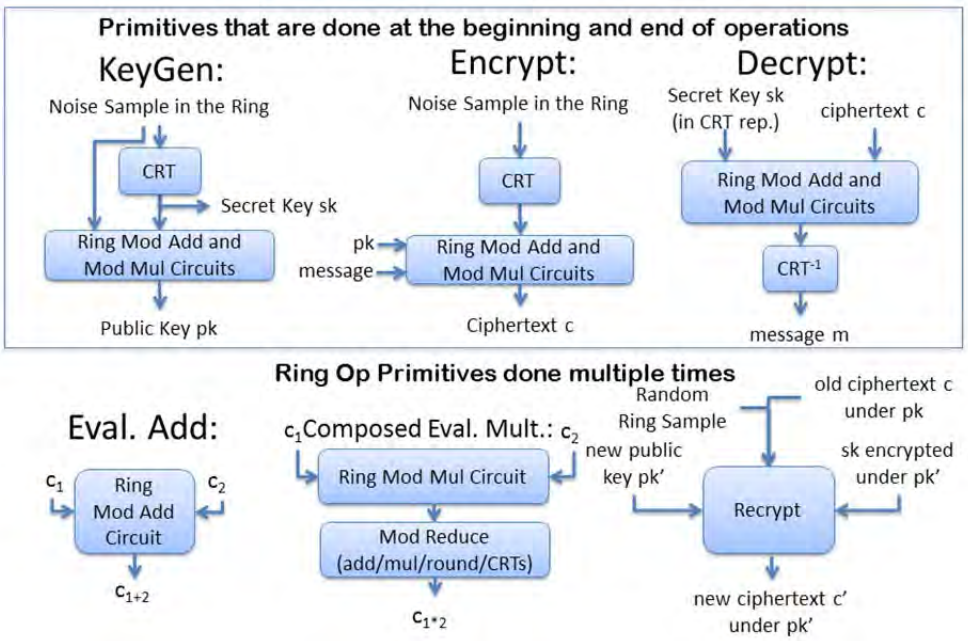


Figure 2: SIPHER lattice library primitives

Although SIPHER can support general FHE capabilities, we can also target our design on a more focused SHE capability which supports the execution of limited-depth programs on encrypted data. Known FHE schemes are built from SHE schemes with the addition of Bootstrapping to extend the depth of computation supported by SHE. Our Bootstrapping approach is based on [18]. We particularly focus our SHE/FHE design and implementation efforts on a variant of the LTV scheme [8] which is itself based on NTRU [19]. We also focus on implementations for enterprise computing environments with multi-core x86 computing infrastructure and hardware acceleration on FPGAs to support parallelism.

Although it is possible to support parallelism at multiple levels in our design, we focus on providing parallelism using the "double-CRT" representations of ciphertext [10]. Prior SHE and FHE implementation designs [4], [11], [20], [21], for the most part, rely on single-threaded execution on commodity CPU-type hardware, partially due to the difficulty of or lack of native support for multi-threaded execution with underlying software libraries [22], [23]. Although there have been prior implementation efforts that support double-CRT representations to reduce runtime by reducing the bit-widths of ciphertexts to be less than 64 bits, ours is the first implementations that leverages double-CRT implementations to reduce runtime through parallelism.

2.3 Application of SHE to multiparty Voice over IP Teleconferencing

Despite our design foci on a general SHE/FHE library for enterprise environments, we show how our library is adaptable to support a practical end-to-end encrypted VoIP teleconferencing prototype running on commodity iOS-based iPhones. The basis of this application is that there

has been an unmet technological need to provide a scalable capability for multiple geographically distributed people need to simultaneously converse as a group at the same time over commercial data networks. This need, until know, has been partially served by either have physically secure dedicated point-to-point communication links as provided by dedicated circuits, or through physically unsecured point-to-point communication links which are made secure with point to point encryption technologies [24], [25]. These prior approaches to secure communication are either not scalable or have not adequately addressed several important vulnerabilities to address this need. Physically secure communication links are not feasible over broad geographic areas and are not accessible by the general population. Point-to-point encryption solutions do no scale because when there are more than a handful of participants in a teleconference call, a large number of point-to-point communication links are practically difficult to setup and maintain, often leading to latency issues which would degrade the quality of user experiences. For these reasons, there is a need for such a technology to provide scalable, secure and practical teleconferencing services which can be used to host multi-party negotiations, planning, education, and information distribution of a sensitive nature.

VoIP can provide a fundamentally scalable and practical approach to teleconferencing, especially with the advent of global packet-switched information networks. Unfortunately, existing VoIP teleconferencing capabilities such as GoToMeeting, Skype and Mumble among others have not been both scalable and secure against data leaking to adversaries who wish to snoop on private or even proprietary group communication. For example, these existing VoIP technologies have been vulnerable to man-in-the-middle attacks of various types [26]. The majority of widely used existing VoIP teleconferencing capabilities require a central VoIP server to mix all of the VoIP signals from clients which are then sent back to the clients. The VoIP mixing operation, which merges the VoIP streams from the clients, has until now needed to be performed in the clear, on unencrypted VoIP data. This creates a possible opportunity for adversaries to snoop on otherwise protected VoIP data if the adversaries gain access to the VoIP server. This security vulnerability is a practical security challenge because VoIP teleconferencing servers are often hosted in a semi secure environment, such as by commodity cloud providers such as Amazon AWS or Microsoft Azure which might not be completely trusted. This induces an unfortunate trade-off of for this architecture of either requiring all participants to maintain group conversations in the clear in untrusted environments or paying a higher cost of maintaining access to a trusted VoIP server if secure teleconferencing is needed. The limitation of required server trust has until now prevented the use of VoIP teleconferencing technologies from being used in regulated industries where privacy is of an utmost concern.

Taken together, these technological deficiencies and practical needs point to a need for a VoIP teleconferencing capability where VoIP data can never decrypted except on the clients which have access to decryption keys. Thus, unlike previous VoIP attack analyses which focus on signaling attacks [27], [28] during VoIP call set-up, we are particularly interested in protecting against man-in-the-middle attacks to protect against compromise at VoIP servers.

As part of our project we developed a secure, scalable and practical method to protect against the leakage of sensitive VoIP teleconferences even on VoIP teleconference servers that have been fully compromised. The basis of our approach is to modify the SIPHER library to support an efficient, additive homomorphic encryption. Teleconferencing clients encode their voice samples with an additive encoding scheme, encrypt their encoded voice data with an additive homomorphic encryption scheme, send their encrypted voice samples to a mixer which performs

an encrypted homomorphic addition on the encrypted voice and sends the results back to the clients. The clients then decrypt, decode and play back the result. Our scheme relies on the pre-sharing of a common private key for an additive homomorphic encryption scheme, but it is possible in principle to practically generalize beyond this pre-shared key design.

We modified the SIPHER library and integrated it with an existing VoIP teleconferencing application to provide this capability on commodity iPhone clients and the current lowest-cost Amazon EC2 server. We describe the design tradeoffs that we made to provide this capability and focus on novel VoIP encoding schemes that makes the mixing of encrypted signals possible at the VoIP server. We discuss related engineering tradeoffs we make so this capability provides relatively high sound quality with full-duplex 100 kbs data rates. This initial implementation is intended to be a proof-of-concept capability, with the possibility of improving upon this technology with existing key management technologies [29] and session initiation technologies [30] with additional engineering investment and little or no research risk.

A preliminary version of some of the material in this report was published [31], but without full discussion of the software engineering and design tradeoffs of the SIPHER library, and without any discussion of the end-to-end encrypted VoIP application.

2.4 Application of FHE to keyword search of encrypted documents

As a demonstration of our FHE capability, we focused on the e-mail filtering problem. We selected a use model where users encrypt e-mail messages on their (trusted) computer, and the messages are sent to an untrusted mail server for forwarding to a destination. In this use model, the mail server also acts as a "border guard": each e-mail message is checked for the presence of certain strings, and is only passed on to its destination if those strings are absent. Because the e-mail messages are private and for that reason encrypted, the border guard must perform this checking without decrypting the messages. Such a use model might appear in a corporate enclave where users need the ability to send encrypted messages out of the enclave and over the Internet, yet the administrators of the enclave need the ability to ensure that certain information is not allowed beyond the enclave. Interestingly enough, this application was one of the few applications identified during the Proceed program as having an appropriate security model for practical application for FHE.

2.1 Hardware acceleration of FHE primitives

One of our main contributions to the Proceed program has been the development of FPGA based hardware primitives to accelerate computation on encrypted data. Cipher texts in our scheme are represented as rectangular matrices of 64-bit integers. This bounding of the operand sizes has allowed us to take advantage of modern code generation tools developed by Mathworks to implement VHDL code for FPGA circuits directly from Simulink models. Furthermore the implicit parallelism of the scheme allows for large amounts of pipelining in the implementation in order to achieve efficient throughput. The resulting VHDL is integrated into an AXI4 bus "Soft System on Chip" using Xilinx platform studio and a Microblaze soft core processor running on aVirtex7 VC707 evaluation board for use as an attached processor over Gigabit Ethernet. The resulting system can also be hosted directly in a computer using the PCIe Express interface for direct access by the host CPU (eliminating the need for the Microblaze processor).

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Our methodology (the three pronged research approach)

Our methodology for SIPHER was to adopt a three-pronged research approach, as shown in Figure 3. Our work falls into three specific categories. The first focus is on the theoretical aspects of FHE. Our FHE expert provides the latest in theoretical innovations as applied to our constrained design space. The second focus is on implementation of software designs that will support practical FHE operations. Often algorithms are developed by the theory group and passed on to the design group, iterating back and forth as improvements are found. The third thrust area is in highly parallelized implementations, for example on multi-core processors, but also in the extreme case with FPGA implementations. We use a common development environment and code generation tools to rapidly prototype our FHE operations in these highly parallelized implementations. The constraints imposed by the implementations are actually fed back into the theory group, where it drove the search for a more efficient implementation of the algorithms from first principles.

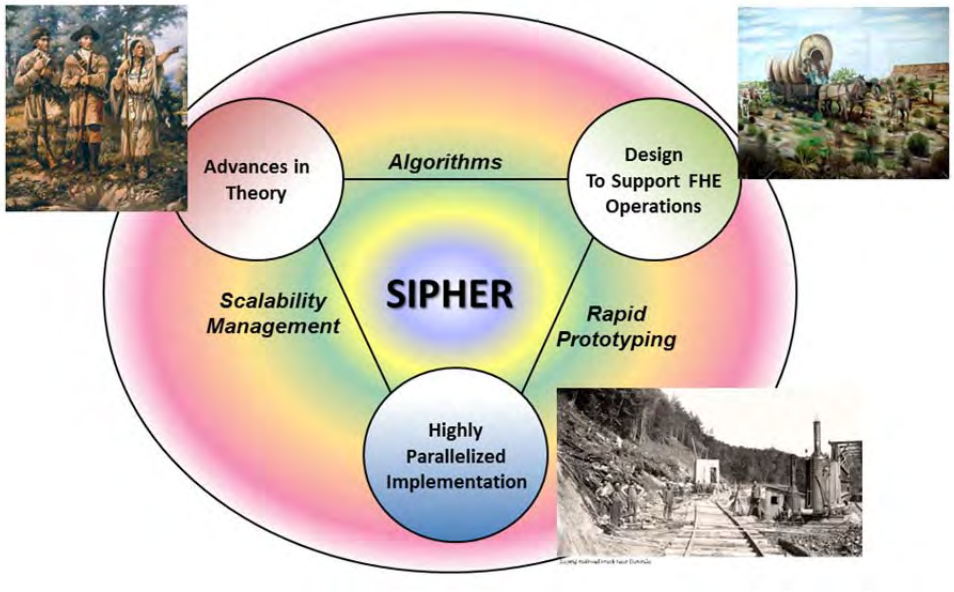


Figure 3: The multi-pronged research approach of SIPHER

3.2 Procedure of approach

Over the course of our four year program, developments generally cascaded from theory to design then to acceleration. Each research thrust was generally six months behind the progress of the one before it. Figure 4 shows a timeline of the various milestones of the program with the major contribution of each six month period. The initial year focused on basic SHE algorithms and early hardware implementation of key primitive operations. This provided a basic functionality, and also required addressing implementation issues such as efficient modulo arithmetic in hardware, and ways to limit the number of bits required for operations. The second year focused on scaling up the software implementations, and implementing the first FHE Processing Unit (FHEPU) on an FPGA (first a Virtex 6, then later a Virtex 7). The third year added the breakthrough theory of Power of 2 bootstrapping, and also focused work on the two

applications mentioned in this report (secure VOIP teleconferencing and encrypted keyword search). The final year focused on accelerating the KWS with FHE (including bootstrapping) using FPGA acceleration.

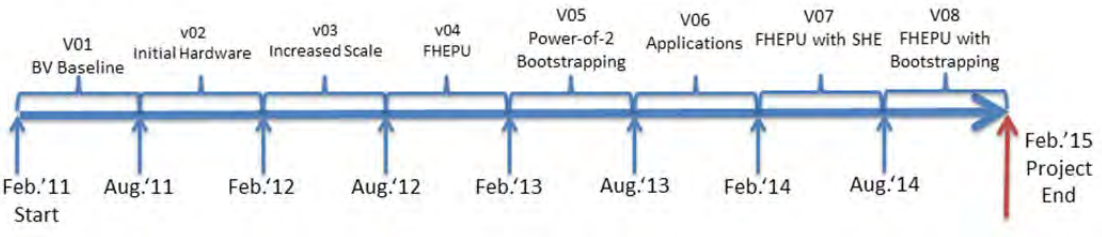


Figure 4: A timeline of SIPHER advances over the duration of the program

3.3 SIPHER cryptosystem

3.3.1 SIPHER cryptosystem design

We begin by describing the overall target of cryptosystem target of the SIPHER library. We use this cryptosystem based on [8] to motivate design and engineering tradeoffs in our SIPHER library, even though we target SIPHER to be as modular and adaptable as possible to support variations of this scheme.

We begin with some mathematical preliminaries based on our focus on power-of-2 cyclotomics. For n a power of 2, we define the ring $R = \mathbb{Z}[x]/(x^n + 1)$ (i.e., integer polynomials modulo $x^n + 1$) where, and for any positive integer q, define $R_q = R/qR$ (i.e., integer polynomials modulo $x^n + 1$, with mod-q coefficients). The message space is R_p for some integer $p \geq 2$, and most arithmetic operations are performed modulo some $q \gg p$ that is relatively prime with p. Fast addition and multiplication in R_q can be performed by using the mod-q CRT representation of elements.

Basic functions

With these mathematical preliminaries, the mathematical description of the basic LTV-variant scheme with "least significant bit" message encoding is as follows. (Concrete parameters and implementation discussions are given later.)

- KeyGen: choose a "short" $f \in R$ such that $f = 1 \bmod p$ and f is invertible modulo q, and a "short" $g \in R$. Output $pk = h = g \cdot f^{-1} \bmod q$ and $sk = f$.
  Note that f is invertible modulo q if and only if each of its mod-q CRT coefficients is nonzero. The CRT coefficients of f^{-1} (modulo q) are just the mod-q inverses of those of f. Concretely, the "short" elements f and g can be chosen from discrete Gaussians. E.g., we can let $f = p \cdot f' + 1$ for some Gaussian-distributed f'.
- Enc($pk = h, \mu \in R_p$): choose a "short" $r \in R$ and a "short" $m \in R$ such that $m = \mu \bmod p$. Output $c = p \cdot r \cdot h + m \bmod q$.
  Concretely, m can be chosen as $m = p \cdot m' + \mu$ for a Gaussian-distributed m'. In some cases it may be better to choose m as a zero-centered random variable congruent to μ modulo p.
- Dec($sk = f, c \in R_q$): compute $\bar{b} = f \cdot c \bmod q$, and lift it to the integer polynomial $b \in R$ with coefficients in $[-q/2, q/2)$. Output $\mu = b \bmod p$.

Homomorphic operations

- EvalAdd(c_0, c_1): output $c = c_0 + c_1 \bmod q$.
- EvalMult(c_0, c_1): output $c = c_0 \cdot c_1 \bmod q$.

With the use of EvalMult, the decryption procedure changes slightly. Define the "degree" of ciphertexts as follows: a freshly generated ciphertext has degree 1, and the degree of c = EvalMult(c_0, c_1) is the sum of the degrees of c_0 and c_1. Then decryption of a degree-d ciphertext c is the same as above, except that we compute $\bar{b} = f^d \cdot c \bmod q$.

Key Switching

Key switching converts a ciphertext of degree at most d, encrypted under secret key f_1, into a degree-1 ciphertext c_2 encrypted under a secret key f_2 (which may or may not be the same as f_1). This requires publishing a "hint" $a_{1\to 2} = m \cdot f_1^d \cdot f_2^{-1} \bmod q$ for a "short" $m \in R$ congruent to 1 modulo p. (Concretely, we can choose $m = p \cdot e + 1$ for a Gaussian-distributed e.)

- KeySwitch(c_1, $a_{1\to 2}$): output $c_2 = a_{1\to 2} \cdot c_1 \bmod q$.

(Note that $a_{1\to 2}$, c_1, c_2 can all be stored and operated upon in CRT form, so key switching is very efficient – just one coordinate-wise multiplication of the CRT vectors.)

Modulus Reduction

Modulus reduction converts a ciphertext from modulus q to a smaller modulus (q/q′), where q′ divides q (and so is also relative prime with p), while also reducing the underlying noise by a q′ factor.

The basic description is as follows: given a ciphertext $c \in R_q$, we add to it a small integer multiple of p that is congruent to −c mod q′. This ensures that the underlying noise remains small, that the plaintext remains unchanged, and that the resulting ciphertext is divisible by q′. Then we can divide both the ciphertext and modulus by q′, which reduces the underlying noise term by a q′ factor as well.

Note that the final step (of dividing by q′) implicitly multiplies the underlying message by $(q')^{-1} \bmod p$. We can either keep track of these extra factors as part of the ciphertext and correct for them as the final step of decryption, or we can just ensure that $q' = 1 \bmod p$, so that division by q′ does not affect the underlying message.

The following formal procedure uses the fixed (ciphertext-independent) value $v = (q')^{-1} \bmod p$, which can be computed in advance and stored.

- ModReduce(c, q, q′):
  1) compute $d = c \bmod q'$ (in coefficient form).
  2) let $\delta = (vq' - 1) \cdot d \bmod (pq')$, with all of δ's entries in $[-pq'/2, pq'/2)$.
  3) let $d' = c + \delta \bmod q$. In coefficient form, all the entries of d′ should be divisible by q′.
  4) output $(d'/q') \in R(q/q')$

The above is most efficient to implement when $q = q_1 \cdot \cdots \cdot q_t$ is the product of several small, pairwise relatively prime moduli; when q′ is one of those moduli (say, $q' = q_t$ without loss of generality); and when c is represented in "double-CRT" form, i.e., each of c's mod-q CRT coefficients is itself represented in (integer) CRT form as a vector of mod-q_i values, one for each i. Then the above steps can be performed as follows:

1) Computing $d = c \bmod q_t$ (in coefficient form) is done by inverting the mod- q_t CRT on the vector of mod- q_t components of c (leaving the other mod- q_i components untouched).

2) Computing δ is done by just multiplying the coefficients of d by the fixed scalar $(vq_t - 1)$ modulo p q_t, and putting the results in the desired range.

3) Adding δ to c is done by computing the double-CRT representation of δ (i.e., applying each mod- q_i CRT to δ), and adding it entry wise to c's double-CRT representation.

Note that the mod- q_t CRTs of δ and c are just the negations of each other (by design), so their sum is the all-zeros vector. Therefore, there is no need to explicitly compute the mod- q_t CRT of δ (though it can be done as a sanity check).

4) Computing d'/ q_t is done by dropping the mod- q_t components in the double-CRT representation of d' (which are all zero anyway), and multiplying every mod-q_i component by the fixed scalar $q_t^{-1} \bmod q_i$. (These scalars can be computed in advance and stored.)

Composed EvalMult

We use the Key Switching, Ring Reduction and Modulus Reduction operations as supporting functions with EvalMult to improve noise management and enable more computation between calls to the Bootstrapping operation. Taken together, we form a composite operation, which we call ComposedEvalMult (CEM), from the sequential execution of an EvalMult, Key Switching and Modulus Reduction operation.

Ring Reduction is called during some CEM operations, depending on the level of security provided by a ciphertext resulting from the result of the Ring Reduction operation.

As Modulus Reduction operations are performed the security provided by a ciphertexts increases (as described in the next section). Ring Reduction correspondingly reduces the level of security provided by a ciphertext. We implemented our FHE library such that a minimum level of security δ' is provided at all times, and this level of δ' is a parameter selectable by the library user. If a call to a Ring Reduction operation will result in a level of security $\delta \leq \delta'$, then the RingReduction is performed in the CEM operation.

Our conception is that due to the ModReduction and RingReduction component of ComposedEvalMult, it is feasible to coordinate the choice of the original ciphertext width t and the scheduling of CEM operations so that the final ciphertext resulting from secure circuit evaluation and which needs to be decrypted is only one column wide with respect to a single modulus q_1 and provides a level of security at least as great as the original ciphertexts resulting from the encryption operation. More explicitly, if we need to support a depth $t - 1$ computation, the initial encryptions should only be t columns wide to ensure that the final ciphertext is 1 column wide. Whereas the runtime of Encryption, EvalAdd, CEM depend on the ring dimension and depth of computation supported, the Decryption operation would hence depend only on the final ring dimension after all ring switching has been completed. If we need to decrypt a ciphertext that has multiple columns in our double-CRT representation, we could perform multiple ModReduction operations to reduce this $t > 1$ ciphertext until we are left with a single mod-q_1 column.

Bootstrapping

The addition of the Bootstrap operation enables the noise that accumulates in our ciphertext to be "refreshed", allowing further CEM operations to be performed until the need for the next Bootstrap "refresh". This can be repeated a large number of times, converting our SHE scheme into an FHE scheme.

In our scheme, a ciphertext is "fresh" if is encoded with a large modulus (i.e., at a high "tower" level t in a double-CRT representation) and at a large ring dimension. As we perform computation on the ciphertext, we iteratively perform modulus reduction operations with every ComposedEvalMult operation, and RingReduction operations scheduled with ComposedEvalMult operations to ensure that a minimum security level is always maintained.

Starting from a "fresh" large ciphertext, with the iterative application of ComposedEvalMult operations, we eventually obtain a resulting ciphertext with a minimal double-CRT representation that consists of a single ciphertext vector (that corresponds to a single-tower level - the base tower level). The Bootstrapping operation refreshes the ciphertext back up to a larger ring dimension and tower level. Bootstrapping does this by first switching to a larger ring, and the largest tower possible. Bootstrapping then performs a homomorphic rounding, thus consuming several of the tower levels to refresh the noise. As such, the resulting ciphertext output by bootstrapping is at a lower level than a ciphertext output by an encryption operation, but still supports computation on itself.

Our design goal was to support operations with ciphertext as large at t=32 and n=16384. In practice, this is at a lower level of security than would be justifiable for high-security applications, and a maximum level of t=16 and n=16384 was seen as a practical limit. Due to limitations in the bit overflows of our ciphertext encoding both in 64 bit CPU and our FPGA implementations, we were practically limited to ciphertext moduli which were less than 64 bits and subsequently were unable to support bootstrapping larger ring dimensions within the remaining scope of the program.

We found that the bootstrapping operation consumed anywhere from 6 levels of a ciphertext towers for n=512, and up to 12 for n=16384. As such, if we support a maximum of 16 levels in practical use of our scheme, we can support depth 4 computations between bootstrapping activities.

The basis of our bootstrapping approach comes from a new approach to homomorphic rounding. This approach to bootstrapping is described in detail in [18]. We provide a high-level overview of this operation here, simplified for our restriction to power-of-2 rings.

This Bootstrap operation has the following steps:

1) Round the ciphertext: For each entry v for residue i, we output round($v * q/q_i$), where the inner expression is rational, and "round" means taking the nearest integer. Generally $q = 2^{\ell}$ is chosen experimentally, but as small as possible.
2) Convert the plaintext modulus: This is a null operation under our simplifying assumptions.
3) Lift the ciphertext and plaintext moduli: This is also a null operation under our simplifying assumptions.
4) Scale the ciphertext: We scale up the ciphertext by a Q/q' factor (rounding to nearest integers in the power basis), and embed into dimension N (new ring dimension) as well. The plaintext modulus is still q'.

5) Compute the homomorphic trace: The following steps are performed iteratively log2(N) times:
    a. ”Lift” the ciphertext modulus to 2Q, which has the effect of making the plaintext modulus 2q.
    b. Apply the automorphism from [18], with appropriate key switching to put the result into the same key as the original ciphertext in the iteration.
    c. Sum the original and resulting ciphertexts.
    d. Divide the ciphertexts by 2.
6) Perform a homomorphic rounding: This operation is described in full detail in Appendix B of [18].

Note: bootstrapping is not a reciprocal of modulus reduction. The reciprocal of modulus reduction is modulus switching (to a larger modulus).

Bootstrapping does not completely reset noise. If we start with a ciphertext with noise level, say n_f, every modulus reduction operation changes the noise level from noise n_i to level n_{i-1} where $n_{i-1} > n_i$. We bootstrap when the noise level reaches n_1. If Boostrapping takes b levels to complete, then the output of a optimally configured bootstrapping operation will always be $n_f - n_b >= n_1$. While not completely resetting noise to the original level, it always resets it to the same level.

3.3.2 SIPHER parameter selection for SHE/FHE

The selection of n and $q_1, \ldots, q_t$ depends heavily on the plaintext modulus p, the depth of computation that needs to be supported, and the desired security level. We capture the primary concerns influencing the selection of a ring dimension n and the moduli $q_1, \ldots, q_t$ at a high level as follows:

- The necessary ring arithmetic should be easily supported on the computation substrate – i.e., that mod-q_i operations (for $i \in \{1, \ldots, t\}$) require few clock cycles.
- The moduli $q_1, \ldots, q_t$ are sufficiently large to enable sufficient noise shrinkage via modulus reduction.
- The ring dimension n and noise parameters are sufficiently large so the scheme provides adequate security.
- The ring dimension n is not so large that it becomes overly time-consuming and memory intensive to manipulate the ciphertexts.
- The plaintext modulus p and any noise added to the ciphertext during encryption is sufficiently small that we can evaluate reasonably sized circuits with correct decryption.

We choose to add discrete Gaussian noise to the fresh ciphertexts where r = 6 represents the selected probability distribution parameter. We have found theoretically that the smallest modulus q_1 needs to satisfy the expression $q_1 > 4pr\sqrt{n}w$ in order to ensure successful decryption, where the parameter $w \approx 6$ represents an “assurance” measure for correct decryption (essentially, the probability of decryption failure is bounded by the probability that a normally distributed variable is more than $w\sqrt{2\pi}$ standard deviations from its mean), and $p \cdot r$ is the Gaussian parameter of the noise used in fresh ciphertexts. (Hence r is the Gaussian parameter of the underlying NTRU-like problem.)

After selecting q_1, we select the remaining $q_i \in \{q_2, \ldots, q_t\}$ such that $q_i > 4p^2r^5n^{1.5}w^5$, which ensures that modulus reduction by a factor of qi sufficiently reduces the noise after a

ComposedEvalMult operation. For implementation simplicity, we set q_1 to be the smallest feasible solution to $q_1 > 4p^2r^5n^{1.5}w^5$. Consequently all q_i are represented by $\log_2(q_t)$ bits, leading to simpler implementations. Table 1 shows how many bits are required to represent $q_1, \ldots, q_t$ for varying ring dimensions for p = 2. Note that all $q_1, \ldots, q_t$ can be represented in less than 64 bits.

Table 1: Dependence of bit lengths of moduli q_i, as a function of ring dimension for p = 2.

Ring dimension n	512	1024	2048	4096	8192	16384
Bit length $\log_2(q_i)$	44	45	47	48	50	51

Following [32]–[35], we use the standard "root Hermite factor" δ as the primary measure of concrete security for a set of parameters. The most recent experimental evidence [32] suggests that δ = 1.007 would require roughly 240 core-years on recent Intel Xeon processors to break. Using the estimates from [33], [34], we found that in order to achieve a security level δ for a depth of computation d = t − 1 using the t moduli $q_1, \ldots, q_t$, we need to ensure that $n \geq \lg(q_1 \cdot \cdot \cdot q_t)/(4 \lg(\delta))$.

Table 2 shows how δ varies as a function of the ring dimension and depth of computation supported. Based on our analysis, if we impose the requirement that $\delta \leq 1.007$, then we would need to use ring dimension n = 16324 to support depth d = 13 computations. The colors correspond to roughly equivalent security level.

Table 2: Security level δ, as a function of depth of computation supported (columns) and ring dimension (rows) for p = 2

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8	52772	28157	15024	8016	4277	2282	1217	649	346	184	98	52	28	14	8	4	2	1.2152
16	276	199	144	104	75	54	39	28	20	14	10	7.	5	4	2	2	1.5256	1.1024
32	20	16	14	11	10	8	7	5	5	4	3	2	2	2	1.7754	1.4929	1.2554	1.0556
64	4	4	3	3	3	3	2	2	2	2	1.9204	1.7563	1.6061	1.4689	1.3433	1.2285	1.1235	1.0274
128	2	2	2	1.9704	1.8792	1.7923	1.7093	1.6302	1.5548	1.4828	1.4142	1.3488	1.2863	1.2268	1.1700	1.1159	1.0643	1.0150
256	1.5246	1.4879	1.4520	1.4171	1.3830	1.3497	1.3172	1.2855	1.2545	1.2243	1.1949	1.1661	1.1380	1.1106	1.0839	1.0578	1.0323	1.0075
512	1.2494	1.2335	1.2177	1.2022	1.1868	1.1716	1.1567	1.1419	1.1273	1.1129	1.0987	1.0846	1.0708	1.0571	1.0436	1.0302	1.0171	1.0041
1024	1.1210	1.1136	1.1063	1.0990	1.0918	1.0846	1.0775	1.0704	1.0634	1.0564	1.0494	1.0425	1.0357	1.0288	1.0221	1.0153	1.0087	1.0020
2048	1.0619	1.0582	1.0546	1.0509	1.0473	1.0437	1.0400	1.0364	1.0329	1.0293	1.0257	1.0222	1.0186	1.0151	1.0116	1.0081	1.0046	1.0011
4096	1.0312	1.0294	1.0276	1.0258	1.0239	1.0221	1.0203	1.0185	1.0167	1.0149	1.0131	1.0113	1.0095	1.0077	1.0059	1.0041	1.0023	1.0006
8192	1.0162	1.0153	1.0144	1.0134	1.0125	1.0115	1.0106	1.0096	1.0087	1.0078	1.0068	1.0059	1.0050	1.0040	1.0031	1.0022	1.0012	1.0003
16384	1.0083	1.0078	1.0073	1.0068	1.0064	1.0059	1.0054	1.0049	1.0044	1.0040	1.0035	1.0030	1.0025	1.0021	1.0016	1.0011	1.0006	1.0001

We will show in our results section later, that while this suffices for most cases, when Bootstrapping is used, there are issues associated with our parameter selection that limit the security of our FHE operations in our current software and hardware implementations (due to limitations on FPGA circuit modulus bit width).

3.4 SIPHER library and software architecture for SHE and FHE

We implemented our scheme in the Mathworks Matlab environment and used the Matlab Coder toolkit [36] to generate an ANSI C representation of our implementation. We subsequently hand-modified our auto-generated ANSI C to incorporate the pthreads library [37]. This leveraged

efficient mechanisms for light weight parallelism. We compiled this ANSI C using gcc to run as an executable in a Linux environment. We believe that additional performance improvements could be obtained by implementing our FHE scheme natively in C.

We chose to implement our scheme in Matlab because it provides an interpreted computation environment for rapid prototyping with native support for vector and matrix manipulation which simplifies implementation development. We found the Matlab syntax to be a natural fit for writing software to support the primitive lattice operations needed for our double-CRT NTRU-based SHE design.

We wrote our Matlab implementation of our double-CRT NTRU SHE scheme using the Matlab fixed-point toolbox. The Matlab fixed-point toolbox also provides a path toward generated HDL implementations of our design that can be deployed for practical use on highly parallel computing hardware such as FPGAs. Part of our vision for the use of our SHE design was to develop an FPGA implementation of FHE [38], [39] as discussed later in this report.

3.5 Application of SIPHER to SHE VOIP and resulting architecture

3.5.1 Design goals for SHE VOIP teleconferencing

We identified several design goals and metrics of performance with which to evaluate and reason over our end-to-end encrypted VoIP teleconferencing designs and implementations. Our primary high-level design goals and metrics are:

1) Sound Quality: The end-to-end encrypted VoIP teleconferencing capability should provide sound quality at least as good as a Public Switched Telephone Network (PSTN), preferably with full-duplex.
2) Latency: The end-to-end encrypted VoIP teleconferencing capability should provide an end-to-end latency ideally of less than 100ms for trans-continental VoIP teleconference session, a generally accepted reasonable latency for VoIP technologies, but more latency is acceptable for inter-continental operations.
3) Scalability: The end-to-end encrypted VoIP teleconferencing capability should be able to support four people speaking simultaneously while tens of participants listen without degradation in sound quality or latency.
4) Secure: The end-to-end encrypted VoIP teleconferencing capability should provide an encryption work factor roughly at least as good as the work factor for AES-128. This means that the VoIP data, when encrypted, should require at least as much computational effort to obtain the unencrypted data without a key as is needed for AES-128, a commonly used point-to-point secure encryption technology.
5) Resource Efficient: FHE schemes have been known to require encrypted data which is much larger than the original source data. Early schemes provided a ciphertext expansion of several orders of magnitude larger than the source data. The end-to-end encrypted VoIP teleconferencing capability should ideally require less than an order of magnitude ciphertext expansion.
6) Wide Geographic Area: The end-to-end encrypted VoIP teleconferencing capability should operate with users and the VoIP mixing server over a wide geographic area, ideally transcontinental if not inter-continental without an unacceptable degradation in sound quality or latency.
7) Portable: The end-to-end encrypted VoIP teleconferencing capability should be easily ported to other client and server types.

8) Easily Deployable: The end-to-end encrypted VoIP teleconferencing capability should be easy to deploy, such as with small binaries.
9) Usable: The end-to-end encrypted VoIP teleconferencing capability should be intuitive and easy to use.
10) Extensible: The end-to-end encrypted VoIP teleconferencing capability should be easy to modify to add additional and more advanced functionality at a later date.

3.5.2 Architecture for VOIP

Figure 5 shows a high level example illustrative application of this privacy-preserving VoIP teleconferencing technology with end-to-end encryption. Each of the client's samples users' voice data, encodes it, encrypts it and sends the result to the VoIP mixer. The mixer sends a result back which is then decrypted, decoded and played back to the clients' users. Any encryption system could be used that supports an additive homomorphism which could be implemented in a practical manner. A representational scheme that supports additive homomorphisms is NTRU which can be made both Somewhat Homomorphic (SHE) and Fully Homomorphic (FHE) in addition to additive homomorphic.

Our approach uses a shared secret key, but more general designs are possible that generalize beyond this initial shared secret key design. Input voice streams from clients are sampled and homomorphically encrypted using a client's public key. The encrypted voice samples are sent to an FHE-enabled VoIP server that does not have access to encryption keys. The VoIP server combines and balances the encrypted audio feeds. The combined output is then forwarded to the client handsets, where it is decrypted and played back for the user. Our HE-based solution processes streaming audio at 10 kBytes/s per voice.

The output of the processing is sent to the client, where it is decrypted using the clients private key. No keys are stored on the teleconference server, so privacy is preserved even if an adversary views all communication links and operations on the server. No trust of the communication links or teleconference server is required to provide privacy. The level of security provided in the current prototype is roughly at the level of AES-128, but parallels between the security levels of the encryption scheme and other current standards are not exact. We can increase the security of our teleconference capability to be arbitrarily higher at the expense of voice quality by decreasing sampling rate and dynamic range.

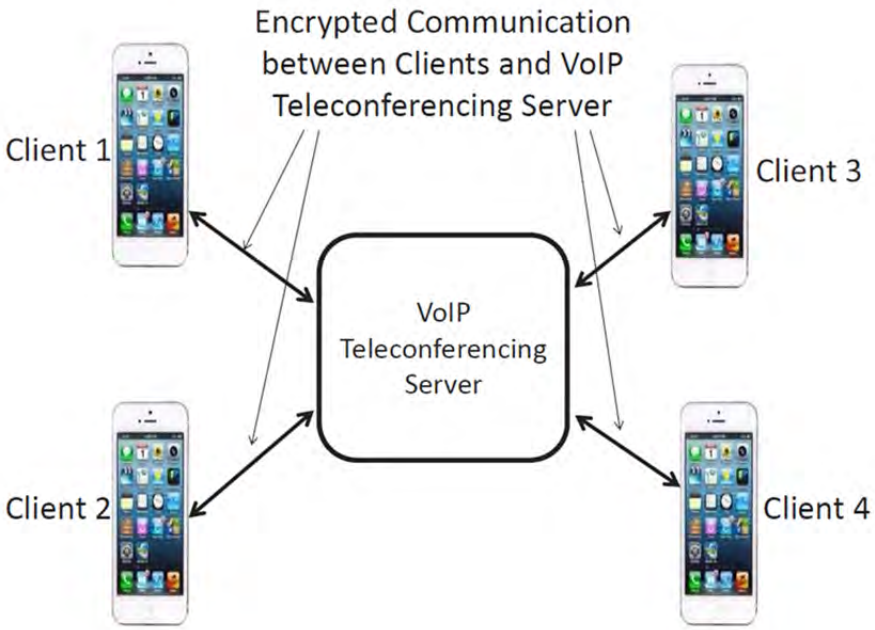


Figure 5: High level VoIP Teleconferencing Design

Figure 6 shows how the clients support data flows internally. In the top of the diagram, data from the microphone is sampled and fed to the encoder, encrypted using an additive homomorphic encryption scheme and sent to the mixer. As seen in the bottom of the figure, the result returned from the mixer is decrypted, decoded and played back over a speaker.

Figure 7 shows how the VoIP mixer takes encrypted input from various clients and returns a common output. For a representational VoIP system with clients (c_1, c_2, c_3, . . . , c_m), a client c_i would want ($c_1+c_2+. . .+c_{i-1}+c_{i+1}+. . .+c_m$). This summation can be performed in a tree fashion as illustrated in Figure 7. For our representational NTRU scheme, the ciphertexts are vectorized in blocks of m, and all additions are performed modulo some large integer q pre-specified by the key generator.

Our encoder/decoder is additive so that we can rely on an additive homomorphism such as the EvalAdd operation to mix VoIP signals. Because we require only an efficient secure EvalAdd operation to support encrypted VoIP mixing, our design builds on the recent efficient FHE design and implementation discussed in [31]. We simplified this prior work such that we remove the ability to support EvalMult operations. As such, because we only need to support much smaller circuits, we do not need the parallelism capabilities as discussed in [31] for our VoIP application and integration with the existing Mumble/Murmur open-source VoIP systems. We also use much smaller parameters than the designs advocated in [31] because we require much more greatly reduced functionality. Thus, the basis of our encryption approach is a special limited version of FHE called Additive Homomorphic Encryption which allows an untrusted computation host to compute the encrypted sum of encrypted integers.

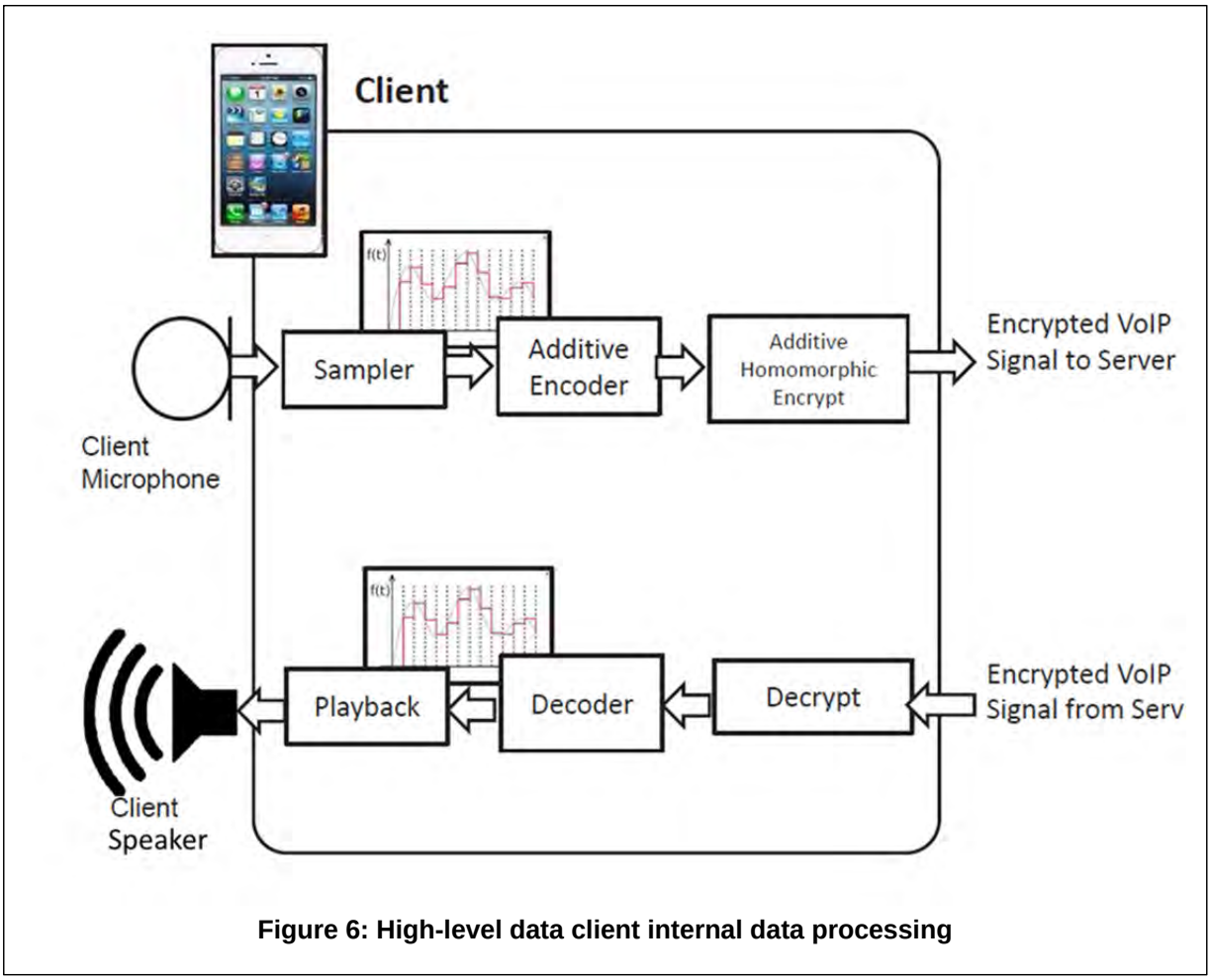


Figure 6: High-level data client internal data processing

Client Vocoder

We have a developed a vocoder technology which takes voice samples from a client and encodes the voice samples as vectors of integers. This vocoder is linear so that it can be used, for example, with an additive homomorphic encryption scheme to provide an encrypted VoIP teleconferencing capability. In this example, the encoded voice samples are encrypted using the additive homomorphic encryption scheme. These operations are performed on multiple clients. The resulting ciphertexts are sent to a VoIP mixer which queues and adds the ciphertexts from the clients. The resulting added ciphertext can be sent back to the clients. When decrypted with the additive homomorphic decryption scheme, decoded using our decoding scheme and played back to the clients, the resulting audio is a mixing of the audio from the clients.

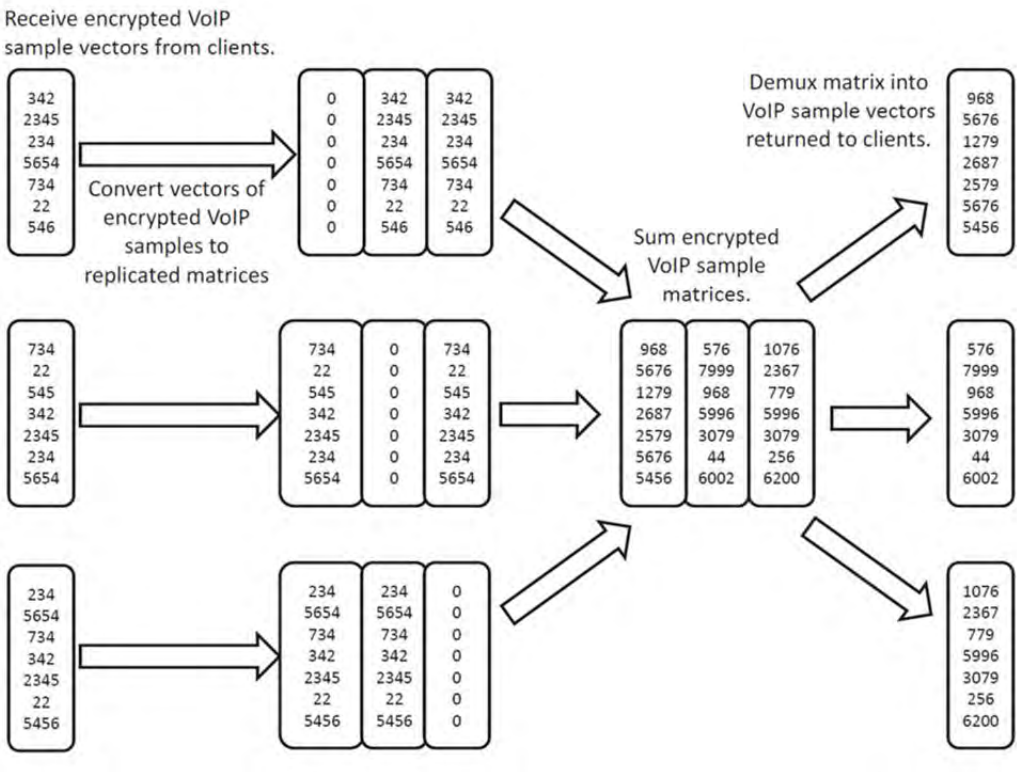


Figure 7: Encrypted VoIP Server mixing for three clients

Our encoding goal is to convert a length-m data frame of y-bit VoIP samples into a length- n frame of integers with the property that Encode($input_1$) + Encode(input2) = Encode($input_1$ + $input_2$). As seen in the left hand side of Figure 8, we split the length m sample input into multiple blocks of $n = 2^{(floor(log2(m)))}$-length vectors and a single mod(m − n)-length vector if mod(m − n) > 0. The first step is to shift the samples so they are centered around 0, mod 2^y. For the z^{th} block of samples, we multiply the integers in this block by $2^{(y + z - 1)}$. We also pad the m − n block of samples with 2n − m zeros so this vector is n samples long. As seen on the right hand side of Figure 8, we sum these vectors. These operations are all highly efficient as they only involve splitting vectors, multiplication be two and bitwise concatenation, which are all extremely efficient to implement. This result is the encoded vector and has the desired Encode( ) property above. This encoded data is subsequently used for encryption.

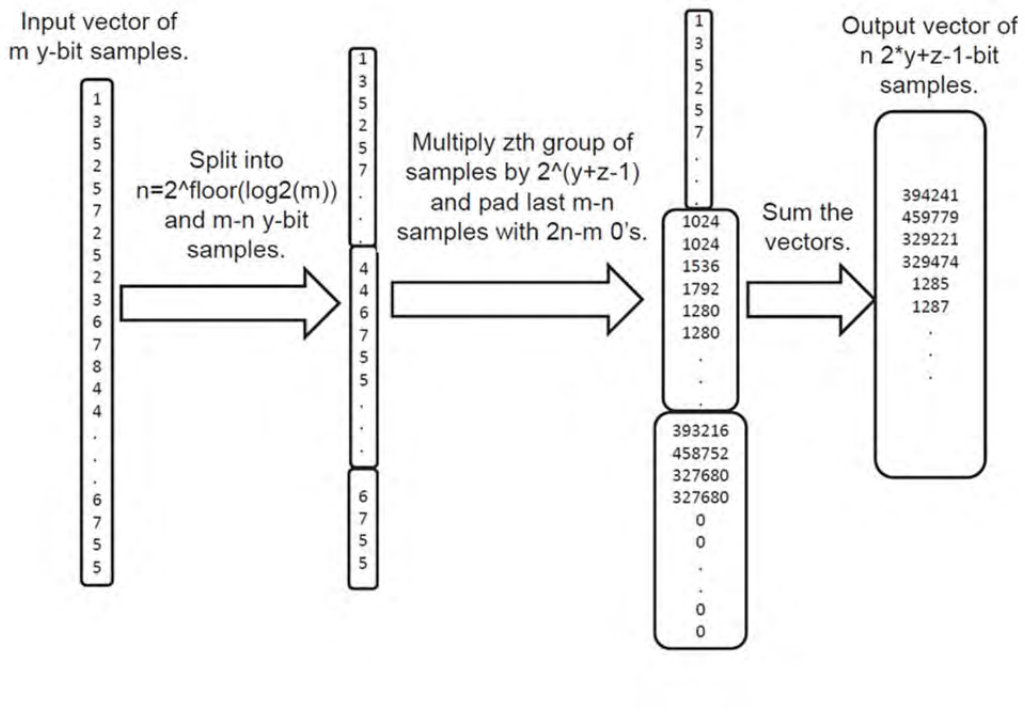


Figure 8: Encrypted VoIP encoding

Figure 9 shows our decoding process. On the right hand side of this figure we take the input vector. We make copies of this block and perform an integer division by $2^{(y+2*z-1)}$ for the z^{th} block. We then concatenate these vectors and return the result. Like for the encoding operation, these operations are all highly efficient as they only involve splitting vectors, multiplication be two and bitwise concatenation, which are all extremely efficient to implement.

3.5.3 Homomorphic encryption and Key Generation for SHE VOIP

In this subsection we describe the additive homomorphic cryptosystem we use to construct the end-to-end encrypted VoIP capability built on [31]. This cryptosystem is very similar to the NTRU system [19], [40], though it was not until recently that its homomorphic properties were noticed independently by L´opez-Alt et al. [41] and Gentry et al. [42]. A more general version of this cryptosystem was discussed in [31], but we discuss here a more limited version of the cryptosystem which is simplified for more efficient end-to-end VoIP encryption.

The discussion of this simplified cryptosystem has a high degree of overlap with the more general cryptosystem. Our simplifications reside primarily in the encryption and decryption operations, but we include the full key generation and evaluation addition operations which are

also modified, but to a lesser extent, for the sake of completeness. The modifications are primarily in the avoidance of any ciphertext decomposition to parallelize operations when the ciphertext modulus $q > 2^{64}$. We have found that we can parameterize the cryptosystem to support the vector addition of adequately large plaintext vectors such that requiring a larger ciphertext modulus is not needed. As such, because we can limit ourselves to 64-bit operations, our simplified cryptosystem can be implemented to run highly efficiently on native 64- and 32- bit processors without the parallelism advances obtained in [31] for more efficient more general computations.

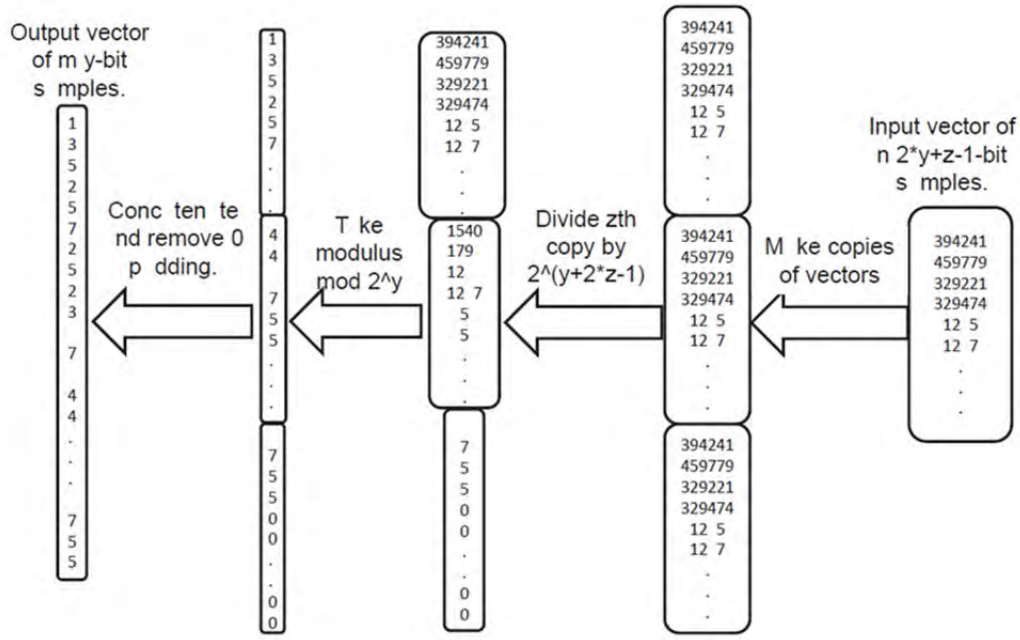


Figure 9: Encrypted VoIP decoding

The basic design of the cryptosystem was detailed in section 3.3.1. For the cryptosystem the message space is R_p for some integer $p \geq 2$. We use a mod-q Chinese Remainder Transform (CRT) representation of elements to provide fast addition. The basic operations of the scheme are as the same as section 3.3.1, with the exception that the Homomorphic Operations EvalMult and Bootstrapping are not used.

3.5.4 Engineering tradeoffs in parameterizing SIPHER for SHE VOIP

We need to choose parameters for both the vocoder and the cryptosystem so that:

- VoIP signal data is encoded into VoIP plaintext.
- The VoIP plaintext can be securely encrypted into VoIP ciphertext.

- The summation of multiple VoIP ciphertexts can be successfully decrypted back into VoIP plaintext.
- The output VoIP plaintext can be decoded into an undistorted VoIP signal.
- All these operations need to be run efficiently on commodity hardware, such as 64- and 32-bit ARM and x86 processors.

Generally, these concerns mean that:

- The bitwidth of the VoIP data P needs to be sufficiently large so that given a VoIP integer signal vectors from y speakers $v_1, v_2, \ldots, v_y$, we are guaranteed that $v_1 + v_2 + \cdots + v_y = (v_1 + v_2 + \cdots + v_y) \bmod P$.
- The number of layers in the encodings (and hence the ring dimension n) and the plaintext modulus $p = 2^x$ need to be sufficiently large with respect to P so that for the encodings $z_1, z_2, \ldots, z_y$ where $z_i = \text{encode}(v_i)$, we have $z_1 + z_2 + \cdots + z_y = (z_1 + z_2 + \cdots + z_y) \bmod p$.
- The ciphertext modulus needs to be sufficiently small that we can support computations on the ciphertext efficiently. For modern smart phones this means that the ciphertext modulus is at most 2^{64}, so we can use native 64-bit computations.
- The selection of parameters needs to provide a non-trivial root Hermite factor to provide security guarantees.

The selection of the ring dimension n and ciphertext modulus q parameters depends heavily on the desired security level and the plaintext modulus p. The plaintext modulus p depends on the VoIP data modulus P, the number of VoIP streams that need to be mixed without distortion y and the VoIP data bit width P. The selection of a ring dimension n and the modulus q follow the same procedure described previously in section 3.3.2.

3.5.5 Integration of SIPHER library with a VoIP teleconferencing framework

We evaluated our end-to-end encrypted VoIP capability by implementing our vocoder and homomorphic encryption library and then integrating them with an existing open-source VoIP teleconferencing capability. This activity resulted in an end-to-end encrypted VoIP client for teleconferencing clients running in an Apple iOS environment composed of a) the open-source Mumble VoIP client modified integrated with b) a custom linear codec of our design written in ANSI C and c) an FHE encryption library ported from Matlab to ANSI C. We also wrote and deployed the VoIP server capability running on Linux computing devices to perform the homomorphic mixing operation. We describe the implementation of this capability in this section.

Codec and Homomorphic Encryption implementation

As with the cryptosystem design, our implementation used for an additive homomorphic encryption library is a customization of the design introduced in [31]. We implemented our scheme in the Mathworks Matlab environment and used the Matlab Coder toolkit [36] to generate an ANSI C library of our implementation. We believe that additional performance improvements could be obtained by implementing our HE scheme natively in C.

As mentioned in section 3.4, we chose to implement our scheme in Matlab using the Matlab fixed-point toolbox. We implemented the vocoder capability in native ANSI C. We compiled this capability using the gcc tool to create a vocoder library which we then integrated with the homomorphic encryption library and a VoIP teleconferencing substrate.

VoIP Teleconferencing substrate

Rather than construct a VoIP capability from whole cloth, we decided to construct an end-to-end encrypted VoIP teleconferencing capability by integrating our additive homomorphic encryption library and our vocoder library with an existing open-source VoIP teleconferencing library. We selected the Mumble VoIP library (http://mumble.sourceforge.net) for this integration because the Mumble is mature, and offers high sound quality and runs on a variety of platforms.

We decided to implement our end-to-end encrypted VoIP teleconferencing capability for iOS clients because the native iOS development environment uses Objective C, a dialect of ANSI C. However, even though we only developed iOS clients, there is no reason our client library could not be integrated in other environments such as for Android, Windows, Mac or Blackberry clients.

By integrating with the Mumble library, our end-to-end encrypted VoIP library has the same usage and deployment models as the standard Mumble capability. Notably, Mumble clients present the user a simple, easy to use, graphical user interface that can be easily understood with minimal training. An image of the modified client running on an iPod Touch can be seen in Figure 10 where the client is running in push-to-talk mode. This client is indistinguishable from the standard iOS Mumble client. The Mumble software can also be deployed through an app store model, or as binaries which can be loaded onto iOS devices through XCode.

We integrated the iOS capability so that client handsets encrypt their audio streams using the client's public key. The proxy server computes over that encrypted data without decrypting the data or sharing keys. The output of the processing is sent to the client, where it is decrypted using the clients private key. No keys are stored on the teleconference server, so privacy is preserved even if an adversary views all communication links and operations on the server.

This integration was relatively straight forward with several notable exceptions to reduce packet drops and improve sound quality:

1) The client application generated voice packets that contained 480 samples at 48 KHz, or 10 mSec worth of sound. The sound driver, however, generated slightly larger packets. As a result, the period of the sound packets was slightly larger than 10 mSec, and every so often two sound packets were generated back to back. The original server set a 10 mSec timer and just accepted one packet every 10 mSec. We added a small queue at the server so we did not drop packets when we received two packets in a row very quickly.
2) We generated new frame numbers at the server as opposed to re-using the client frame numbers. The clients correlated the frame numbers with time. This cut down on the time jitter with regard to frame numbers.
3) The encryption and decryption operations for our applications were processor intensive, and were run in batches of several audio packets at once. We moved the encryption and decryption operations to a low priority thread and had the higher priority thread accept and queue new audio packets (both from the network, and from the microphone). This helped prevent a situation where we audio packets were dropped because we were too busy decrypting or encrypting.

The goal of these changes was to reduce the drop rate of packets (an issue with initial prototypes). This in turn, allowed us to increase the audio sampling rate. As a result, we achieved

sampling 10 bit samples at a rate of 48 kHz. This configuration provides a sound quality substantially better than PSTN as long as there are only a few packet drops.

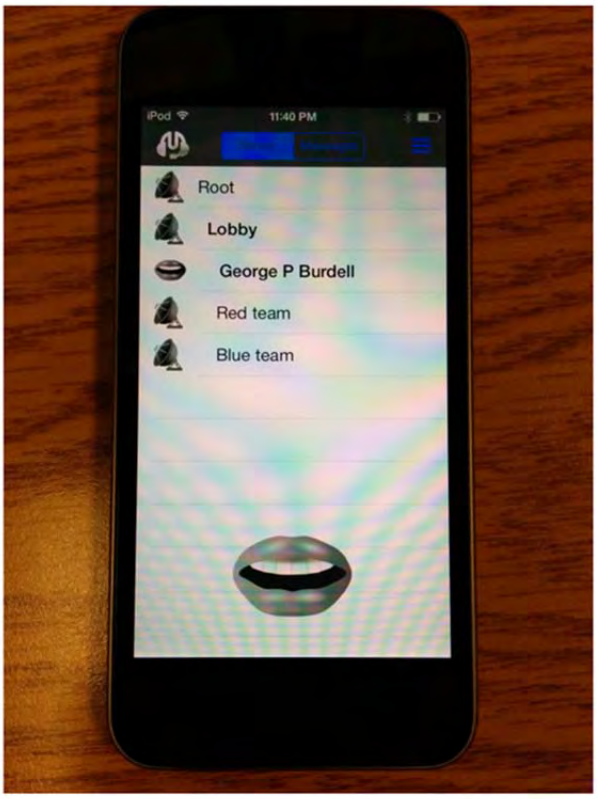


Figure 10: The Push-To-Talk client GUI

After sampling the audio, we queue and encode 90 mSec blocks of this data into our encoder. We designed the system to accommodate 4 speakers, resulting in the homomorphic mixer to add four 10-bit integers homomorphically, resulting in a 12-bit plaintext without the encoding layering. If we use a ring dimension n = 1024, we are required to use 2-layer encoding and have a resulting plaintext modulus of $p = 2^{24} = 16777216$. This encoding and encryption results in a root Hermite factor of $\delta = 1.006$ which is currently believed to be at least as secure as AES-128. With these parameter settings we observed that when running on an iPhone 5s, the encoding and encryption operation took a mean time of 9.2 mSec and decryption and decoding took 4.6 mSec. The summation on the VoIP server took 0.5 mSec. Transport of encrypted VoIP traffic from Cambridge MA to the Northern Virginia Amazon AWS servers took an average of 15 mSec. This resulted in a mean latency much less than our 100 mSec threshold for VoIP traffic, well within the bounds of reasonable, both in theory and in practice.

3.6 Application of SIPHER to FHE keyword search (KWS) of encrypted documents

We developed a FHE application that searches for encrypted keywords on encrypted text. This method relies on a homomorphic string comparison operation that is repeated for all keywords in all locations in an encrypted message. We imported this technology into a mail-guard-type scenario to provide outsourced mail filtering based on keywords of interest to email clients. A sketch of use of this technology can be seen in Figure 11.

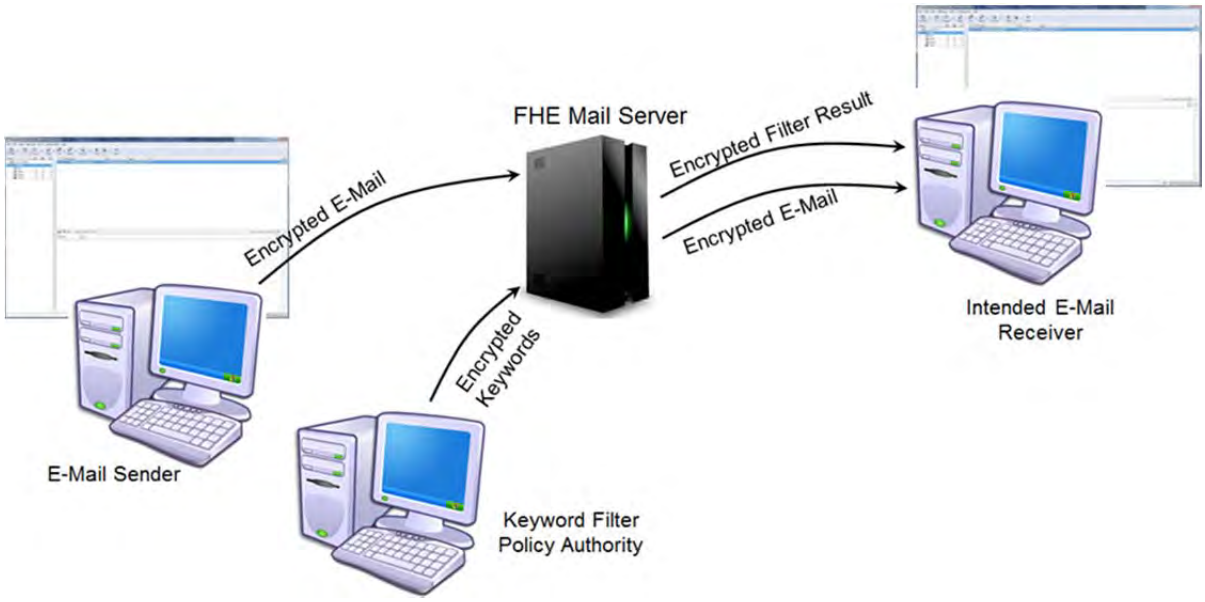


Figure 11: Architecture of the FHE-based Encrypted keyword search E-Mail Guard

The basis of our approach is a homomorphic "Secure Symbol Matching" method that relies on a set of symbols (called an alphabet) that is initially mapped to an integer representation. This approach is seen in Figure 12. In this example, ASCII characters in a text file maps to an integer between 0 and 128. We can compute the equality of two encrypted characters by homomorphically evaluating their difference and then homomorphically raising the difference to the (p-1) power. Using a related technique we can logically AND the results for all the characters in a keyword into a single encrypted true/false bit. For a k bit keyword, we repeat the k-bit string comparison multiple times, over the entire encrypted message, each time shifting the starting point in the encrypted text to the next letter. The logical AND of all the string search results is done in a binary tree structure to minimize the number of repeated AND operations the data must undergo. (ANDs are performed with ComposedEvalMult operations, so a *t*-level SHE implementation can only perform *t* repeated ComposedEvalMult on any one piece of encrypted data – a binary tree AND lets us compute 2^t concatenated ANDs without Bootstrapping.)

Because our keyword search was done on a regular computing platform (PC) we did not develop any novel encoding techniques as we did for VOIP. Rather we focused on accelerating the KWS using our FPGA hardware accelerator. Our results will be presented in section 4.

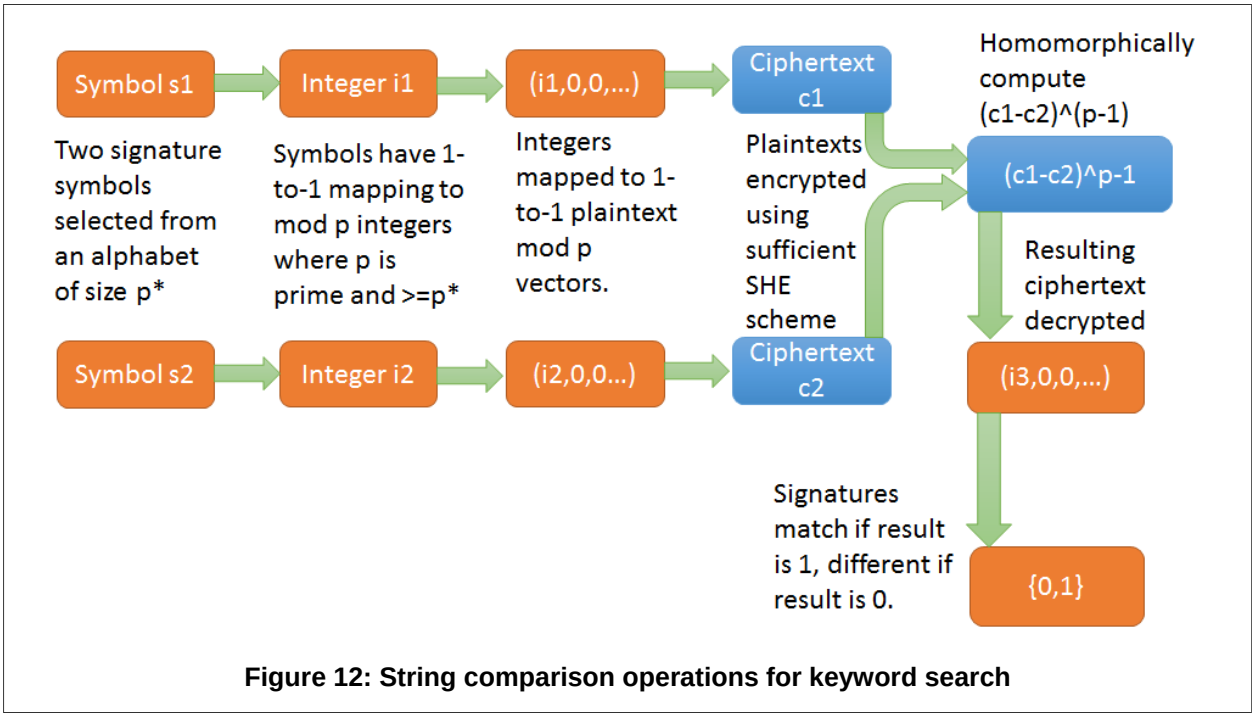


Figure 12: String comparison operations for keyword search

3.7 Implementation issues for parallelism and FHE hardware acceleration

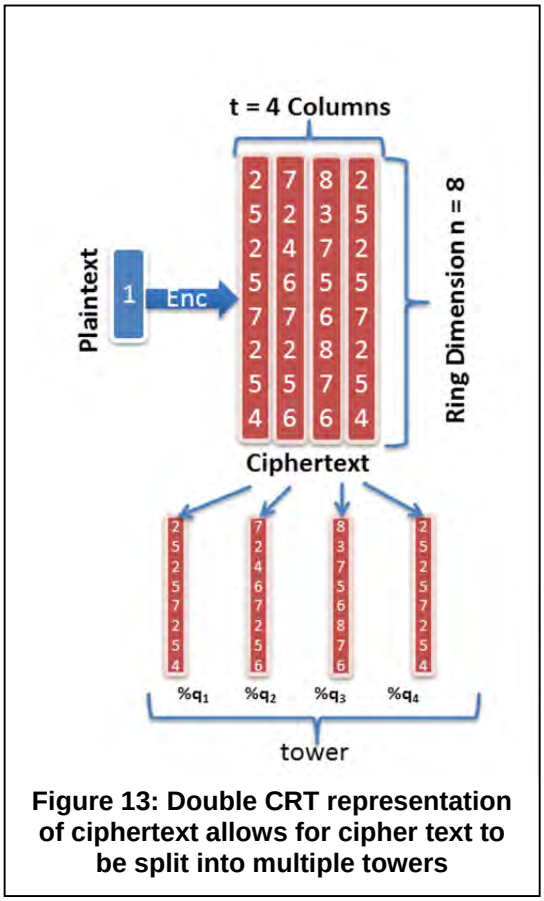


Figure 13: Double CRT representation of ciphertext allows for cipher text to be split into multiple towers

We have covered the underlying crypto system in detail, but now we will focus on aspects that are important when implementing accelerators in hardware. The main advantage of our system is the use of the "double-CRT" representation of cipher texts which is discussed in [43]. With this double-CRT representation, we can select parameters so that cipher texts are secure when represented as matrices of 64-bit integers, but still support the secure execution of programs on commodity computing devices without expending unnecessary computational overhead manipulating large multi-hundred-bit or even multi-thousand-bit integers. Additionally, the parallelism implicit in this data representation is easily exploited to achieve efficiencies during implementation. Figure 13 shows a schematic representation of how ciphertext is represented in this double CRT format.

Our implementation encrypts a plaintext bit into a two dimensional array of 64 bit unsigned integers[1]. We use a residue number system implementation to represent cipher texts as *T* sets of length-*N* integer vectors. A ring in the tower entry *t* has a unique modulus q_t which bounds all entries in that ring. The *n* dimension is known as the *ring size*, and the *t* dimension as the *tower size*. This representation allows us to operate in parallel on the smaller bit width mod- q_t values instead of on a single modulus *q* of much larger bit width, where $q = q_1 * q_2 * \ldots * q_T$ for pairwise co-prime moduli q_t..

As outlined in section 3.3 previously, our implementation requires only a few elementary operations to be implemented on the FPGA hardware in order to achieve large run time speedups over conventional CPU implementations. These operations are:

- RingAdd: $c_{n,t} = (a_{n,t} + b_{n,t}) \% q_t$
- RingSub: $c_{n,t} = (a_{n,t} - b_{n,t}) \% q_t$,
- RingMul: $c_{n,t} = (a_{n,t} * b_{n,t}) \% q_t$.

All three of the above operations can be parallelized or pipelined over both *n* and *t* . Also required are the

- CRT and Inverse CRT, which are implemented as a Number Theoretic Transform [44] coupled with a pre or post RingMul with an appropriate Twiddle Vector.
- Round: A function to perform modulo rounding using different tower moduli (detailed below).

The two repeated key ring operations EvalAdd and EvalMult are the core functions in FHE. When our parameters are chosen such that a single plaintext bit is encrypted, the resulting operations on the encrypted data are XOR and AND respectively. These two operations allow us to implement any Boolean operation of input cipher text[2].

As mentioned previously, this crypto system, like many FHE systems is random (noisy) in nature. Because of this, only a limited number of operations can be performed on the encrypted data before the noise dominates and decryption is no longer guaranteed. EvalAdd does not add noise to the system, so an unlimited number of such operations are allowed to be chained together. EvalMult however does add noise, and this limits the number of such operations that can be chained together. The double CRT representation allows a very straightforward implementation that controls this noise. This requires the use of both key switching and modulus reduction whenever an EvalMult is performed. The combination of these three steps is known as a Composed EvalMult. The property of CEM is that for a pair of inputs of a given tower size *t*, the output is a cipher text of tower size *t-1*. Thus for an initial tower size of *T*, at most *(T-1)* CEM operations can be performed, allowing SHE. Figure 14 shows the impact CEM has on system parallelism.

[1] While the actual number of bits is determined by the parameter selection of the cryptosystem, we select 64 as our maximum dimension for FPGA implementation.

[2] Any arbitrary Boolean function can be constructed from NAND operations. Since NOT(a) == XOR(a, 1), and NAND(a, b) == NOT(AND(a, b)), the two Homomorphic operations are a sufficient set.

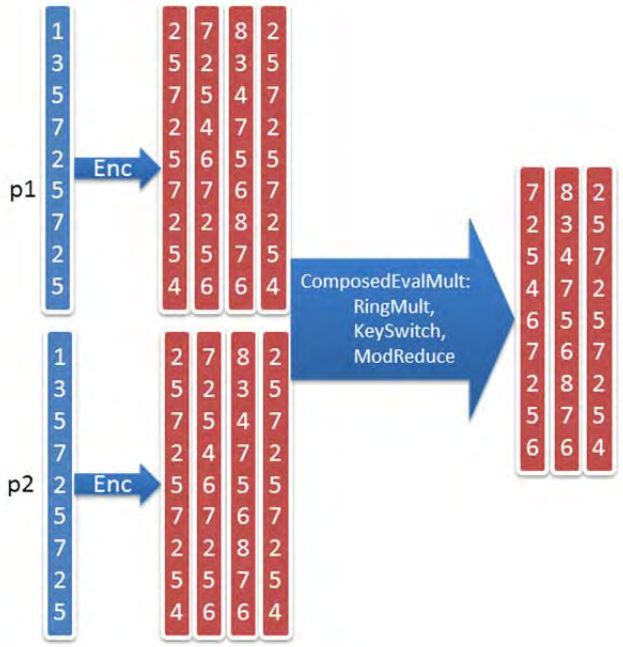


Figure 14: Composed Eval Mult is only Operation not parallelized. Each CEM has a mod reduction step that removes one ciphertext column

The dimensions of the cryptosystem are determined algorithmically, and are a function of security required, and the number of CEM operations required to implement the desired application. If the number of operations required by the application exceeds O(16), then Bootstrapping will be required to reset the noise generated by the cryptographic operations. Bootstrapping is currently on the order of 10 CEM equivalent operations for reasonable security parameters. Bootstrapping has the property of taking a cipher text of tower size *t*, and generating a new 'refreshed' cipher text of the systems original tower size *T*. Thus an unlimited number of operations can be performed on the data, enabling FHE.

All ring operations other than CEM are embarrassingly parallel, i.e. all data needed for an operation stays within a particular tower. In fact in the CEM, everything except the Round operation (discussed later) of the modReduce is also embarrassingly parallel. This parallelism can be exploited in our implementation both on multicores and on the FPGA as show in Figure 15.

Our current SHE scheme relies on operations that are generally inefficient to implement on standard CPU architectures (i.e. modular arithmetic with a large modulus). For convenience, most of the previously published SHE and FHE implementations have used standard tools such as the GNU Multiple Precision Arithmetic Library (GMP) [45], which enable researchers to code operations using very large integers. This limits their focus to operations on CPUs and does not allow them to take advantage of specialized parallel computation hardware like FPGAs which provide highly cost-effective parallelism. Our approach to developing the FPGA code for

implementing efficient ring operations is to develop arithmetic circuits that will achieve high throughput by using parallelism and pipelining on the FPGA as seen in Figure 15.

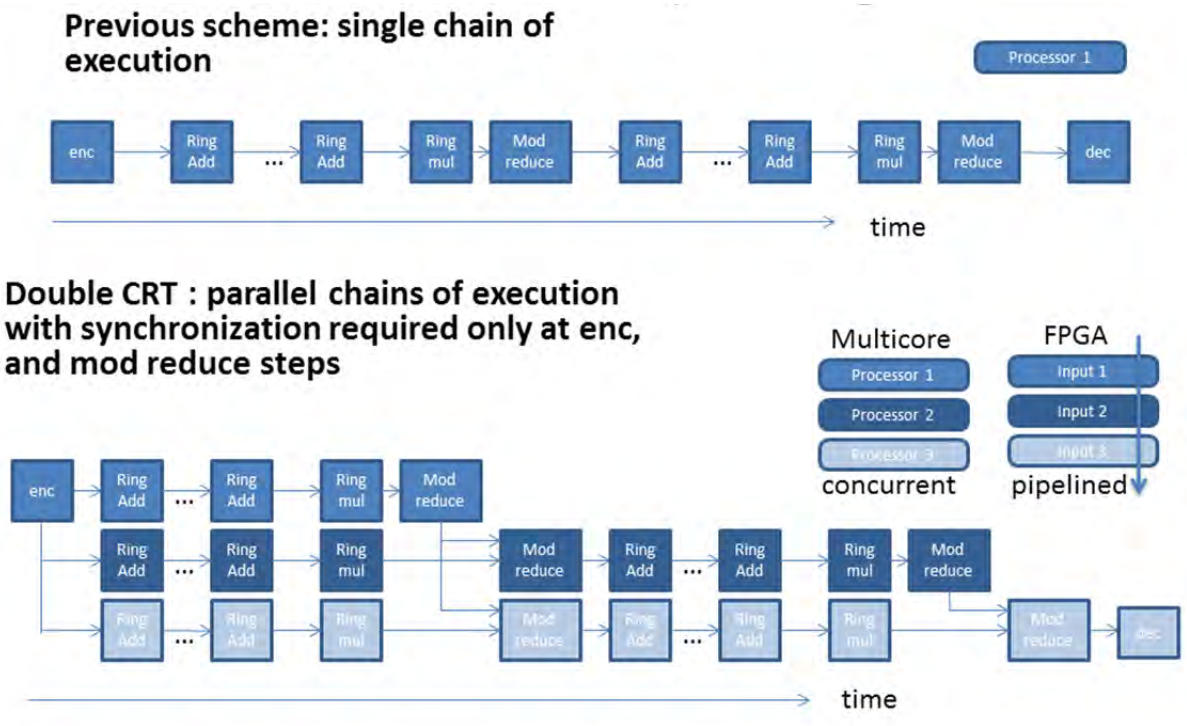


Figure 15: Ciphertext towers enable enables ring operations to be performed on each tower separately. Parallelism is acieved through concurrency on a multicore, or pipelining on an FPGA

Our approach was to develop prototype descriptions in Matlab using the fixed point toolbox. We then re-implemented these primitives in Simulink in a stream-oriented style that allows conversion to VHDL. The results of the two implementations are then directly compared to verify correctness. A conversion from Simulink to VHDL is done in a completely automated fashion using Mathwork's HDL coder. This tool chain provides us the means to develop our primitives, including testing of the resulting VHDL on FPGA hardware, much faster than traditional methods. Some examples of efficiency are:

- The Matlab and Simulink Models are driven with the same fixed point data variables, and generate the same format output, simplifying test and comparison
- The bit width of the circuits is specified at compile time by specifying the bit width of the input data. The sizing of intermediate mathematical operations is done automatically by the fixed point toolbox. Thus many of the same models can be used for 8 through 64 bit inputs.
- The resulting VHDL is vendor independent. This allows for rapid benchmarking on multiple architectures. However, hand optimization of VHDL may be required for optimum performance in order to take advantage of vendor specific IP.
- Mathwork's HDL verifier allows automatically generated FPGA in the loop testing to verify the operation of the resulting VHDL on actual hardware very early on in the program.

4 RESULTS AND DISCUSSION

4.1 SIPHER SHE library functions experimental timing results

We ran our compiled C code (auto-generated from Matlab) SIPHER library implementation on the DARPA Deathstar 64core server with 2.1GHz Intel Xeon processors and 1TB of RAM in a CentOS environment. Although we had access to many resources, we used at most 10 GB of memory and 20 cores during the evaluation of our software implementation.

We collected data on the runtime of the Encryption, EvalAdd, ComposedEvalMult, and Decryption operations over selections of depth of computation supported and ring dimension. We ran 100 iterations of this collection procedure for each combination of t and ring dimension. We used different randomly selected key sets, plaintexts and encryption noise with every iteration to mitigate minor variations in performance that may arise due to these experimental random variables on every iteration. Tables of the raw mean runtime results can be seen in Table 14 through Table 17 in Appendix A.

We collected data on the runtime of the Encryption, EvalAdd and ComposedEvalMult operations for settings of $t \in \{2, 4, 6, ..., 20\}$ and for ring dimensions $n \in \{512, 1024, 2048, 4096, 8192, 16384\}$. We collected data on the runtime of the Decryption operation of final ciphertexts, for computations with fresh (input) ciphertexts with ring dimensions $n \in \{512, 1024, 2048, 4096, 8192, 16384\}$ and depth of computation $t-1$ for $t \in \{2, 4, 6, ..., 20\}$. Note that due to ring switching, the decryption runtime is dependent only on the dimension of the final ciphertext. This is a function of the initial ciphertext and depth of computation. We did not collect data on the runtime of the Bootstrapping operation at this point but present them later on in this report. As discussed in [40], the depth of computation required for bootstrapping is logarithmic in the ring dimension.

Our experimental results shows that run times grow linearly with ring dimension n and the ciphertext width t where $t - 1$ is the depth of computation supported before bootstrapping or decryption could still be performed and have a high probability of recovering a correctly decrypted ciphertext. This makes intuitive sense because as we double either the ring dimension or the ciphertext width, we roughly double the amount of computation that needs to be performed with every Encryption, EvalAdd and ComposedEvalMult operation. Similar results hold for Decryption (Table 17) which shows a linear dependence of runtime on ring dimension, but under the assumption that decryption occurs after $t - 1$ ModReduction operations, including ModReduction operations bundled in ComposedEvalMult operations. Our initial results show that Bootstrapping runtime is similarly linear with respect to the maximum ring dimension. As compared to the results reported in [4], [11], [21], our FHE software implementation provides order-of-magnitude improvements in the runtime of the FHE operations.

4.2 SHE VOIP Teleconferencing experimental results

We experimentally evaluated the performance of the VoIP service by deploying our encrypted VoIP servers in each of the Amazon AWS data centers across the world. We then connected iPod Touch clients to each of the servers through various connection types in the metro area of a United States city in southern New England. These connections included 802.11n wireless enterprise gateway connected to a high-speed enterprise Internet connection, the 4G LTE, 3G and 2G connections over the T-Mobile commercial wireless service and an AT&T DSL connection in a rural area outside the city.

We measured the upload and download throughput of the connections, the drop rate of VoIP packets routed through the various server locations and the subjective quality of the VoIP teleconference session as defined by the experimenters. The upload and download throughput was measured by Ookla throughput measurement app [46] on the client devices. VoIP drop rates were measured experimentally by modifying the VoIP servers to measure drop rates. Voice quality was measured in comparison to PSTN voice quality where "Excellent" means the VoIP conversation was better than PSTN, "Good" means the VoIP conversation was comparable PSTN, "Poor" means the VoIP conversation was worse than PSTN but still usable for communication, and "Unusable" means the connection was useless for communication.

All of the experiments were run over a 2 hour period on a weekday evening using 2 iPod Touch clients with servers deployed on the Amazon AWS t1.micro instances [47]. Each of the clients were on independent connections to the Internet at all times, so there was low likelihood of one client contributing substantially to congestion for the other client.

Table 3 shows the upload and download throughput observed by each of the clients for each of the connections. Note that the rural DSL service provided better throughput than the 2G connection and better download throughput than the 3G connection.

Table 4 shows the packet drop rates observed at each of the servers at the various Amazon AWS locations for the various client connection types. Note that distance between the client and server had only a minor impact on drop rates, while the connection type had a very large impact on drop rates. This implies that the connection could be a bottleneck for the VoIP service.

Table 5 shows the subjective VoIP teleconference quality measurements observed through each of the servers at the various Amazon AWS locations for the various client connection types. Note that distance between the client and server had almost no observed impact on voice quality, while the connection type had a very large impact on voice quality.

We observed that all of the various connections supported acceptable VoIP teleconference capabilities except for the 2G connections. Over all of the acceptable connections, the lowest upload or download throughput observation was on the 3G download: 0.43Mb/sec Because the VoIP download and upload data flows are symmetric, this implies at least a 0.43Mb/sec upload and download throughput connection is required to support VoIP teleconferencing using our prototype.

In addition to our tests of connection-server pairings, we also tested the scalability of the number of clients that could be supported on a single server. For this experiment we connected 7 iPod Touch and/or iPhone 5s clients at various connections on the eastern United States seaboard to a single VoIP server in the Amazon AWS Northern Virginia data center. With these 7 connections running simultaneously with 4 people speaking simultaneously we were able to hold as good as a conversation possible with 4 people speaking simultaneously and no voice distortion was observed by the 3 non-speaking client users.

Table 3: Experimentally measured data throughput in Mb/s for connection types

Connection Type	Upload Rate (Mb/sec)	Download Rate (Mb/sec)
Enterprise 802.11n	38.22	36.53
4G LTE	35.82	17
3G	6.31	0.43
2G	0.2	0.16
Rural DSL	2.55	0.47

Table 4: Packet Drop Rates for various server locations and client internet connection types

Server Location	Client Location	Enterprise 802.11n	4G LTE	3G	2G	Rural DSL
N. Virginia	S. New England	0%	10%	10%	66%	33%
Oregon	S. New England	0%	2%	3%	71%	35%
N. California	S. New England	0%	7%	8%	67%	34%
Ireland	S. New England	0%	7%	7%	73%	38%
Singapore	S. New England	5%	2%	2%	68%	39%
Tokyo	S. New England	1%	3%	4%	69%	37%
Sydney	S. New England	5%	3%	3%	67%	34%
Sao Paulo	S. New England	0.30%	4%	6%	76%	34%

Table 5: Teleconference Quality for various server locations and client internet connection types

Server Location	Client Location	Enterprise 802.11n	4G LTE	3G	2G	Rural DSL
N. Virginia	S. New England	Excellent	Good	Good	Unusable	Poor
Oregon	S. New England	Excellent	Good	Good	Unusable	Poor
N. California	S. New England	Excellent	Good	Good	Unusable	Poor
Ireland	S. New England	Excellent	Good	Good	Unusable	Poor
Singapore	S. New England	Excellent	Good	Good	Unusable	Poor
Tokyo	S. New England	Excellent	Good	Good	Unusable	Poor
Sydney	S. New England	Excellent	Good	Good	Unusable	Poor
Sao Paulo	S. New England	Excellent	Good	Good	Unusable	Poor

4.3 VOIP SHE discussion

Up to now, advances in secure VoIP technologies have focused on providing security for data in transit [24], [25] among other general security challenges such as DDoS attacks [48], identity and key management [49] among many others [50], [51]. These are all important challenges for secure VoIP teleconferencing capabilities, but a reliance on point-to-point encryption between participants has too often led to complicated VoIP teleconferencing systems and protocols [52]. In general, the complicated layering of protection mechanisms is often difficult to execute in practice, leading to overly complicated systems which are difficult to build and maintain. Further, these complicated systems are often difficult to perform security audits on [53]–[55]. Although all of the partial security solutions have worked very well in isolation and have served their purposes as a rule, the at time complicated layering of these protocols has resulted in the introduction of possible security holes which has enabled data leakage.

To the best of our understanding, there have been no VoIP teleconferencing technologies which provide end to- end encryption. Our solution seeks to provide a clean-slate data protection capability that is also compatible, or at least easily integrated with existing VoIP protocols and

architectures. Because we provide end-to-end data encryption, our solution protects data against leakage even when layered with existing VoIP protocols for signaling and transport. Besides providing security against data leakage due to compromised servers, end-to-end encrypted VoIP teleconferencing has the possibility for greatly simplifying existing VoIP protocols, resulting in much simpler implementations and designs, thus resulting in more efficient VoIP implementations that are easier to audit.

The basis of our design and implemented prototype for end-to-end encrypted VoIP teleconferencing is driven by and builds on recent breakthroughs in practical Fully Homomorphic Encryption (FHE). Recent breakthroughs in Homomorphic Encryption have shown that it is theoretically possible to securely run arbitrary computations over encrypted data without decrypting the data [1], [2]. There has been recent work on designing and implementing variations of homomorphic encryption schemes [4], [11], [15], [17], [20], [21], [41], [56]–[58]. These implementations have become increasingly practical with published results on both the runtime of isolated secure computing operations for some implementation [4], [11], [21] and evaluations of composite functions like AES [17], [20], [58].

Current approaches to design FHE schemes rely on a special, highly complex and computationally difficult operation called bootstrapping [18] to support the encrypted execution of arbitrary functions. As such, we use a simplification of the general FHE designs called ”leveled” homomorphic encryption or Somewhat Homomorphic Encryption (SHE) the supports limited depth computations, such as vector addition, which is much more efficient because it does not require the use of bootstrapping.

Besides the runtime challenges of HE designs, there are serious applications issues associated with data structures and representations [17]. Furthermore, it has not been well explored how to convert existing data structures and algorithms into forms that can be efficiently executed using FHE technologies. This is because FHE provides a very different computation model from existing RAM computing devices and the porting of known data structures and algorithms (such as for VoIP mixing) is non-trivial, especially for highly efficient encrypted execution of these algorithms over the encrypted input data. As an example of limitations, early uses of FHE relied on encrypting individual bits in ciphertext. These limitations, in addition to the inherent computational cost of secure computing using known FHE schemes, has until now prevented the practical use of FHE. Our innovation comes from designing a set of data structures, data encoding method (which we refer to as a vocoder) and a homomorphic mixing operation which supports a practical implementation of end-to-end encrypted VoIP teleconferencing.

In particular, a key innovation of ours is to go beyond simple bit-per-ciphertext encodings by placing entire VoIP data frames into each ciphertext. These codec designs are in some sense much simpler than existing modern codecs, such as the mu-law encoders [59] which are much more common in modern VoIP systems. There have been prior known approaches to Additive Homomorphic Encryption, such as Paillier encryption [60], but these approaches have not been practically employed to support encrypted VoIP mixing. Further, there has been no prior work that has investigated the data structures required to support end-to-end encrypted VoIP teleconferencing with homomorphic mixing.

There have been few other approaches to providing secure VoIP teleconferencing that approach to providing security properties such as end-to-end encryption. Most relevant is the work in [61] which discusses a VoIP teleconferencing approach based on Secure Multi-Party Computation

(SMC) [62]. This prior SMC-based approach is also built by modifying the Mumble/Murmur software and our team received implementation advice from the authors of [61]. Unlike our HE based approach which requires only one untrusted server for end-to-end encrypted VoIP teleconferencing, the MPC-based approach in [61] requires that every participant in the teleconference have at least one trusted server.

4.4 FPGA accelerator -- FHE Processing Unit (FHEPU)

4.4.1 VHDL implementations of fast modulus arithmetic using Simulink HDL Generator

Software implementations of modulus usually use some form of trial division to determine the remainder operation. Implementing modulus integers with large numbers of bits in an efficient manner requires the use of special numerical algorithms that have been developed, such as the Montgomery Reduction [63], and the Barrett Reduction [64]. These algorithms avoid division by q, but rather scale the integers so that many of the divisions can be performed by a power of 2, requiring only simple bit shifts. Our SHE scheme requires circuits for fast modulo addition and multiplication (to directly implement the EvalAdd and EvalMult mentioned above). In addition, our scheme relies heavily on the Chinese Remainder Transform (CRT) [65], which can be implemented as an EvalMult of the input with a twiddle table, followed by an FFT [66] that uses modulo integer instead of complex arithmetic (also known as a Number Theoretic Transform or NTT). Our implementation of this FFT uses a standard radix 2 ‘Butterfly’ operations which uses one addition, one subtraction and one multiply, all modulo the residue q. Thus to implement a CRT we need to implement modulo subtraction as well.

Initially, our selection of lattice based SHE led to looking at relatively modest sized modulus, on the order of twenty bits. An implementation using Montgomery Reduction based arithmetic was built that was be relatively efficient, requiring hardware multipliers on the order of 40 bits. However, later research showed that for any reasonable security requirements our SHE scheme would need O(64) bits for our modulus. Our implementation of Montgomery arithmetic in Simulink required us to double our bit width to represent intermediate values represented in Montgomery form. We found that there is an intrinsic limitation of 128 bit width in Simulink even when using the fixed point toolbox. This meant that we could not compile our multipliers for bit widths on the order of 64 bits.

Additionally, our early arithmetic models were all designed for a single value of modulus q to be used for all operations. During the development of our SHE scheme we found that it was more efficient to decompose large bit width numbers into a set of smaller related moduli using the Double CRT representation. This resulted in far more efficient implementations. Thus our circuits would need to operate with multiple (but not unlimited) values of q. As a response to these new requirements we eliminated Montgomery arithmetic entirely and take a simpler approach to modulo addition and subtraction.

Figure 16 shows the Matlab code and the resulting Simulink block for performing a streaming EvalAdd. This circuit requires the inputs to be constrained to less than a given modulus q (which is the native representation for our FHE scheme). The model can operate on one pair of inputs every clock cycle. For simplicity, the model shown does not have any additional pipeline registers (which are modeled as unit delays), but they can be easily added to the model in order to increase the maximum clock speed of the resulting VHDL, at a cost of additional latency. In our applications we expect to process streams of input on the order of several thousand entries,

so this additional pipeline latency is trivial. Additionally, our circuits have an index into a lookup table for the value of q (i.e. which tower index we are operating on), since the bit width of this index is much smaller than that of q (4 or 5 bits vs. 64 bits).

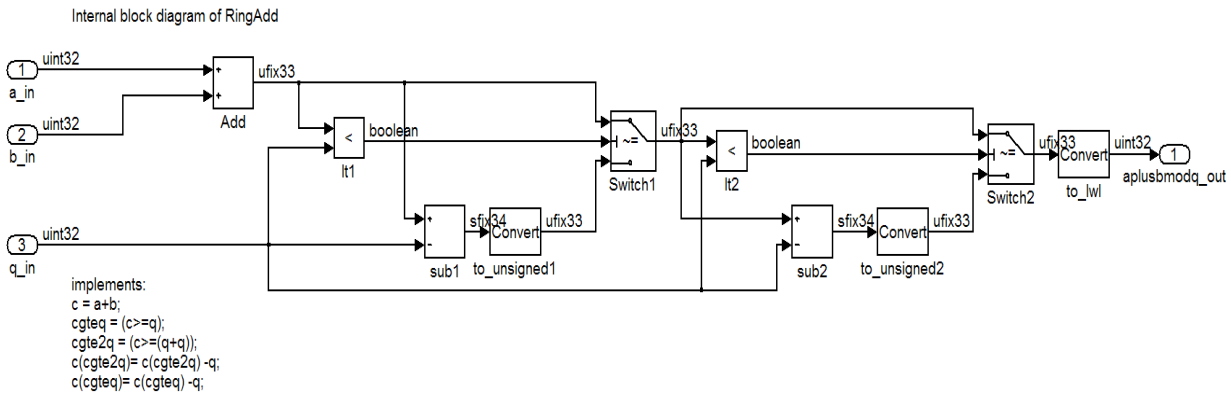


Figure 16: Internal structure of Simulink HDL-ready streaming modulo-add primitive.

Figure 17 shows the Matlab and resulting Simulink block for modulo subtraction. The same comments about pipelining the circuit apply.

Modulo multiplication is a much more complicated operation than either add or subtract, even if the input multiplier and multiplicand are bounded by q. This is because the range of the output of the latter two are bounded by a small integer multiple of q, and can be adjusted within the range of [0…q-1] by simple comparisons and subtraction of q. However, for multiplication the product is approximately twice the bit width of q, so this trick cannot be used. Furthermore, we determined in our earlier work that the VHDL code generated by Simulink for large multiplications is not automatically pipelined, so the resulting (large bit width) multiplies severely restrict the resulting clock rates of the circuits. To address these two constraints, we adopted a recently developed interleaved modular multiplication based on a generalized Barrett reduction [67]. This multiplier has the following properties:

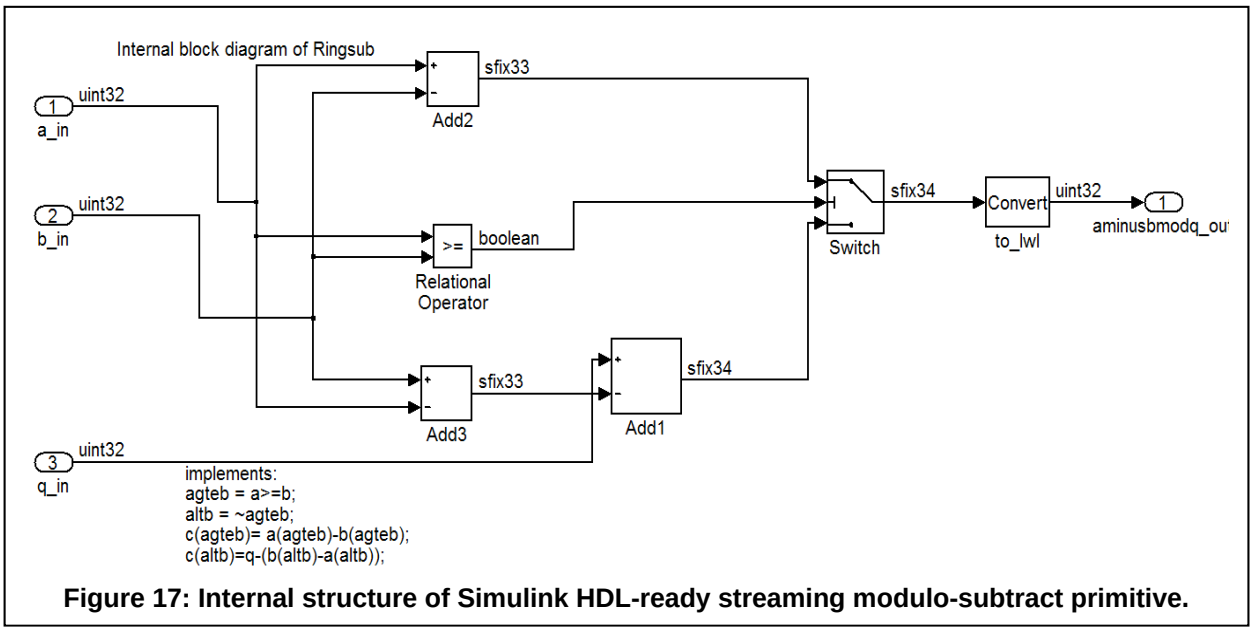


Figure 17: Internal structure of Simulink HDL-ready streaming modulo-subtract primitive.

- Long words of bit length L can be represented by n smaller words of bit length S (i.e. four 16 bit words to represent a 64 bit modulus).
- The multiplication is performed in n stages, where each stage performs one modulo multiplication that is L+S bits long. The stage can be pipelined to perform one modulo multiply per clock cycle.
- Each stage has a Barrett modulus performed on the partial product, which reduces overall bit growth of the partial products to L+S. Each stage requires 3 multiplies, and all divisions required by the Barrett algorithm are implemented as simple bit shifts.
- One circuit can support multiple moduli towers. All parameters that are specific to a given modulus tower can be stored in lookup tables and indexed, in the same manner as q is for our add and subtract circuits.

Figure 18 shows the structure of our resulting multiplier for S=16, and L = 64 = 4*S, resulting in a four stage, 64x64 bit multiplier. This model will produce compile-able VHDL code, i.e. no single operation exceeds 128 bits in width. The red box in the figure shows the model for a single stage in the pipeline. All stages use the same model. This implementation uses 47 stages of pipelining in a single stage order to achieve fast clock rates.

Once the models were maximally pipelined, we identified several large (64 x 64 bit) product blocks within our RingMul Barret multiplication implementation as being the slowest components, and re-implemented them as an expanded multiplication model consisting of four parallel 32x32 bit products, and a pipelined accumulation of partial sums. These are shown in the green and blue boxes in the Figure. This further increased the achievable clock speeds. We discovered that adding additional pipelines of length four, both before and after each resulting smaller product block further allowed the Xilinx optimizer to break these product blocks into multiple DSP48E multipliers in a distributed fashion. This allowed the RingMul circuit to perform at speeds in excess of 350 MHz, well in excess of our target 200 MHz.

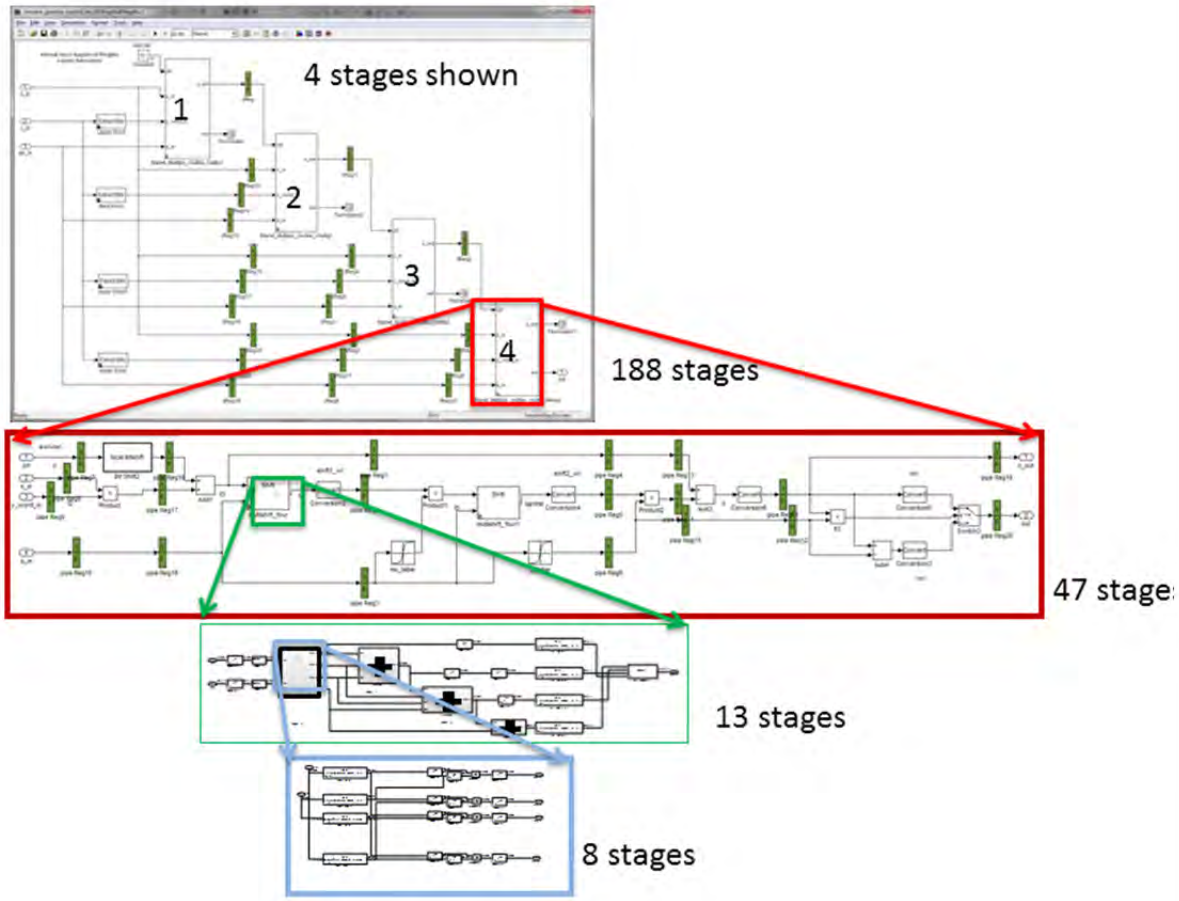


Figure 18:The structures of Simulink HDL-ready, four-stage Barrett 64x64 bit modulo multiply primitive.

The resulting Barrett multi-word circuit supports 32 different moduli (i.e. towers up to 32 in size). Furthermore, the implementation is strictly agnostic to the particular FPGA technology used, and is easily tunable in the generation software to re-optimize for a different FPGA technology.

4.4.2 VHDL implementations of fast forward and inverse CRT using Simulink HDL Generator

The workhorse function for our scheme is the CRT and its inverse, both of which rely heavily on modulo arithmetic. We have developed a Simulink model for performing a fast CRT, based on the modulo arithmetic primitives discussed above. We implemented the Number Theoretic Transform using one of the standard pipeline decimation in frequency FFT architectures, known as the Radix 2, Multipath Delay Commutator [68].

The fundamental structure of the Simulink model that performs a modulo arithmetic FFT (NTT) is identical to a complex version that computes the standard FFT. The only difference is in the Simulink subsystem model that implements the radix 2 butterfly (due to the use of our modular arithmetic function). In fact, the flexibility of the Simulink approach allowed us to debug the

model using complex input and complex butterflies, and then use the same exact structure for the FFT (NNT) with only a change to the butterfly sub system block.

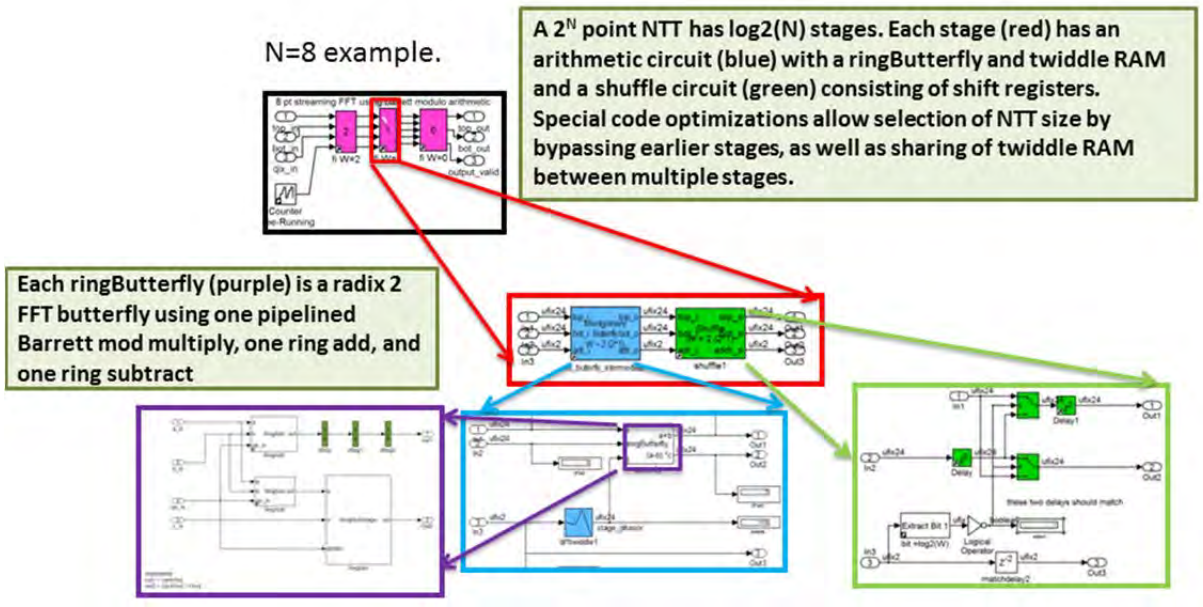


Figure 19: Simulink HDL-ready streaming pipelined CRT Structure

Figure 19 shows the structure of this pipelined CRT. The design trades off area for processing speed. For an N point transform, Nstages = log2(N) radix 2 Butterflies are required (though the last butterfly does not require multiplies). Additionally, 3/2N-2 delay elements are required for the shuffle blocks.

Note that this implementation results in the fastest possible computation rate of the CRT for a given FPGA clock speed, with one pair of output samples being generated every clock cycle. By concatenating several input vectors together sequentially, we can keep the pipeline full and, once the pipeline has filled up, run the circuit at 100% efficiency.

Note that the only difference between the forward and inverse CRT is whether the NTT is pre or post multiplied with a special “Twiddle vector” (different from the NTT/FFT twiddles). We programmed our VHDL wrapper that feeds the data to the NTT and RingMul components so that this pre or post multiplication is achieved in a pipelined manner (to be illustrated in detail later in Figure 26).

The input and output data needs to be presented to the circuit in two parallel streams, with the top stream containing the first half of the input vector and the bottom stream containing the second half. The resulting output is in bit reverse order. Rather than implement this in Simulink we incorporated these data manipulations into the VHDL wrapper around the NTT portion of the CRT as shown in Figure 20. These wrappers use double buffering to efficiently keep the pipeline full.

One shortcoming of this design is that it utilizes a large amount of FPGA area. Thus for a given bit width of q and maximum tower size T, there is a maximum number of stages that will fit into a given FPGA. Another major shortcoming is that each stage has its own “twiddle memory”. In practice, every stage’s twiddle memory is composed of exactly the same even entries in the

twiddle memory preceding it in the pipeline. It ends up that we were not able to fit this circuit as is into our candidate FPGA chip for high security applications where we need to perform CRT operations on vectors of up to 2^{14} in length. There was simply not enough RAM in the FPGA chosen.

Our solution was to develop a custom hand coded twiddle RAM to replace the SIMULINK VHDL twiddle RAM. Figure 20 depicts the final implementation of the CRT module. The data stream is first fed through the "Top/Bottom" block, that divides the single data stream into two data streams, and then through a set of 13 stages. Each stage has two 64-bit wide input streams and two 64-bit wide output streams. The inputs of each stage, except the first stage (labeled "Stage 4096") are the outputs from the previous stage. Stages 4096 down to 4 also contain a bypass capability, depicted in the left of Figure 20. When a particular stage's bypass is enabled, that stage's its input stream is passed directly to its output stream, unmodified. This allows for variable-length CRT operations. If none of the stages are bypassed, a 16384-point CRT is performed. If the first stage (labeled "4096") is bypassed, an 8192 CRT is performed. If first two stages are bypassed (labeled "4096" and "2048"), a 4096 CRT is performed, etc…

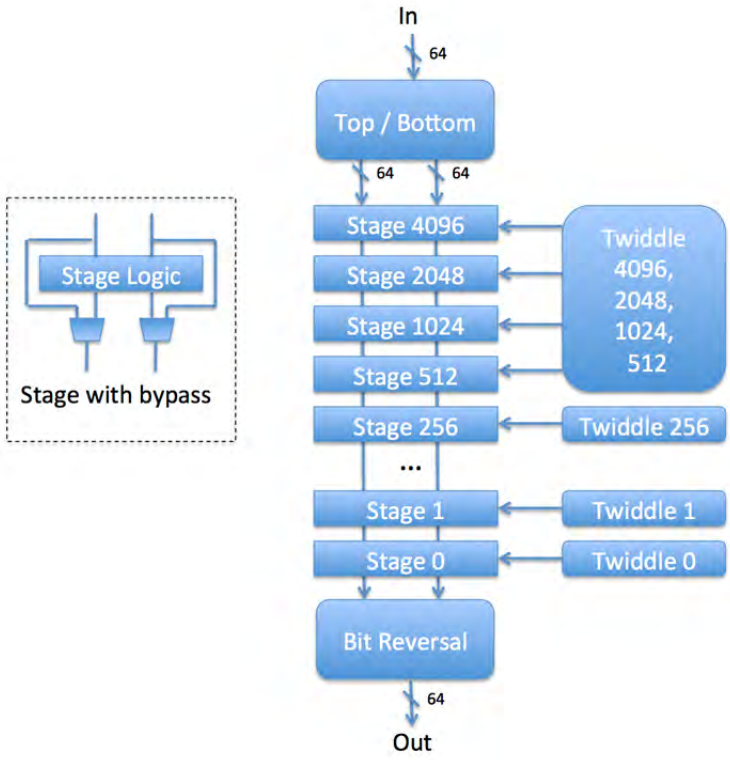


Figure 20: Inside the NTT showing customizations VHDL wrapper that performs required I/O reordering, for sharing twiddle RAM and for stage bypass

Each stage has an associated "Twiddle" Read Only Memory (ROM) table, implemented with FPGA block RAM. The first stage has the largest table, and each successive stage's twiddle table is half the size of the previous stage's twiddle table. Unfortunately, the FPGA does not have enough block RAM resources for each stage to have its own table. To work around this, the first four stages share a table. One reason this is possible is the twiddle tables have been designed with the property that each twiddle table contains same values as the values at the even

addresses in the previous table. For example, the values at addresses 0, 1, 2, and 3 in table 2048 are the same as the values at address 0, 2, 4, and 6 in table 4096. Similarly, the values at addresses 0, 1, 2, and 3 in table 1024 are the same as the values in 0, 2, 4, and 6 in table 2048, which themselves are the same as the values at address 0, 4, 8, and 12 in table 4096. Therefore, the twiddle table for the first stage actually contains the values for all the other tables, assuming it is addressed appropriately. The reason each stage needs its own table, however, is each stage needs to access its table simultaneously in order for the CRT module to achieve the desired data throughput. Fortunately, the FPGA Block Ram primitives are *Dual Port*, which means that it possible to simultaneously read from two independent addresses. By clocking the dual port block RAM at twice the rate (200 MHz) of the rest of the CRT logic (100 MHz), we were able to construct virtual quad port block RAM. For each cycle of the CRT clock (100 MHz), two addresses are presented to each port of the large twiddle table, one on each cycle of its faster clock (200 MHz). As a result, the large twiddle table is able to sustain a throughput of four independent reads per clock cycle, and saves the resources required by the twiddle tables for stages 2048, 1024, and 512. After these savings, as shown in Figure 21, the design still uses 98% of the available FPGA block RAM (BRAM) resources.

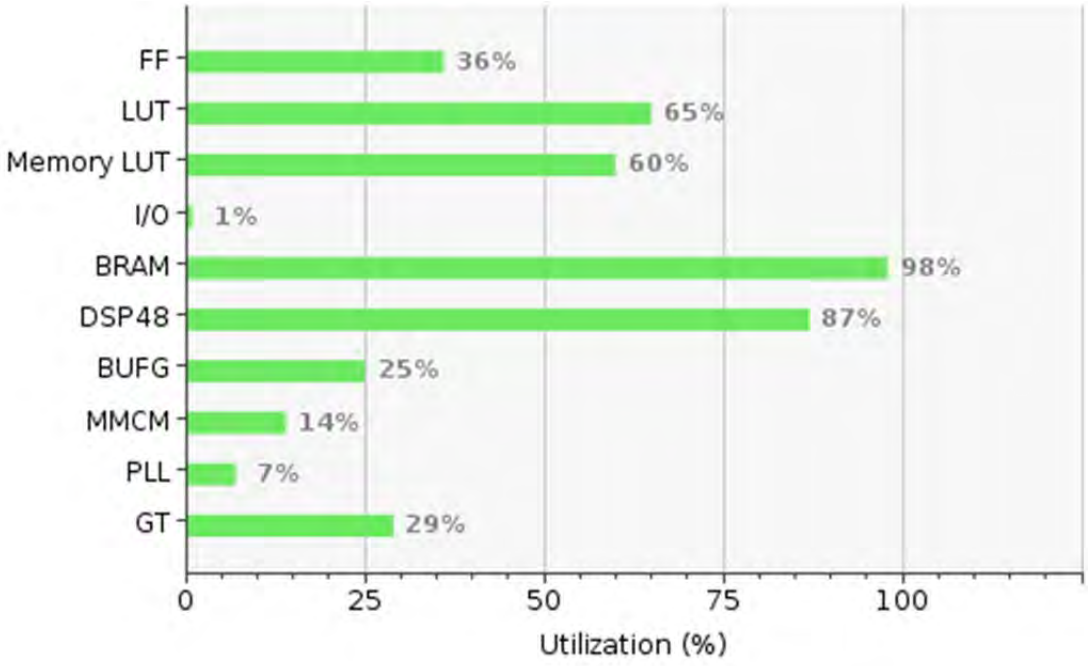


Figure 21: Virtex 7 485T FPGA resource utilization for the FHE accelerator

4.4.3 VHDL implementation of Ring Round

In addition to our implemented ring and CRT operations, we have implemented a function used in our Composed Eval Mult (CEM) called Round. The CEM function is implemented in our software in C, executed several FPGA primitives. First, a RingMul operation performs the multiply. Next a key-switch operation is performed consisting of another RingMul of the product with a hint variable defined by the cryptosystem. Then, a modulus reduction operation is performed on the single highest tower entry of the result which consists of an inverse CRT and this Round operation. Since the Round does several modulus operations not otherwise available, we implemented it in hardware.

Figure 22 shows the Simulink Model of the round which consists of a modified EvalMult operation (using a modified set of moduli q_i), and a pair of operations selected by the range of the result which ensure the output is bounded within an appropriate range. The operations are performed in a pipelined manner as well.

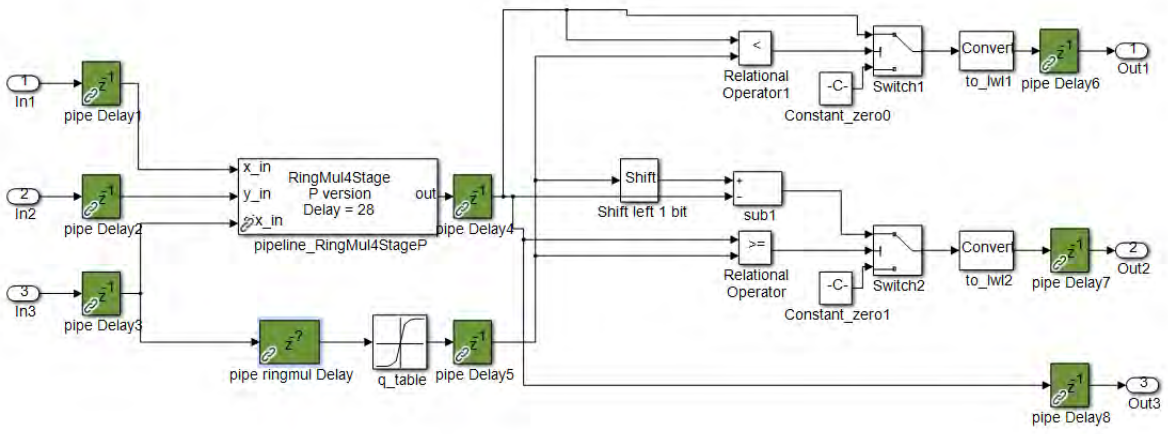


Figure 22: Simulink model of Round function

The result of the rounding operation is a *pair* of new ring vectors that are then in turn applied to each remaining tower entry to reduce the noise accumulated by the initial product. These vectors are first processed with a series of RingAdds, RingSubs and a CRT using each of the corresponding ring moduli. The end result is that the highest tower ring is eliminated from the cipher text, and the overall noise of the system remains at a usable (i.e. decryptable) level.

4.4.4 Further optimizations

Mathworks determined that by selecting synchronous vs. asynchronous reset in the Simulink to HDL generation parameters, the resulting VHDL mapped more efficiently into the registers built into the DSP48E blocks on the Virtex 7 FPGA, increasing the efficiency of the resulting mapped VHDL by eliminating extra routing traces.

The previous circuits were designed to run at a minimum speed of 100 MHz. We determined that adding explicit pipelining stages in the form of delay lines to the model enabled the Xilinx tools to better optimize FPGA mapping during place and route pipelining stages. Specifically pipelines were added between arithmetic operations within the RingAdd (4 stages), RingSub (3 stages), RingMul (188 stages) and RingButterfly (195 stages) models. Since our target ring size can be as large as $2^{\wedge 14}$, and all the towers of a variable are processed sequentially, the delay incurred from filling the pipeline is expected to be minimal.

Several of our circuits utilize lookup tables, both for storing the moduli q_i and for storing various twiddle table entries for the CRT and inverse. Our previous direct implementation of the table lookup using the Simulink Lookup function block maps the resulting ROM directly into gate circuitry. This can increase the place and route drastically for very large tables, and also can result in less efficient circuits. Mathworks determined that by placing an additional delay line, with a “ResetType = none” HDL block property let the Xilinx tools map the table to block ram in the FPGA, which is a more efficient utilization of resources on the chip.

4.4.5 FPGA hardware selection

Our FPGA selection was driven by the need for a large number of hardware multipliers on the chip. Due to cost constraints we wanted to use a commercial off-the-shelf FPGA board for our experiments. Our selection of the Virtex 7 VC707 evaluation board was driven by the following sizing requirements. Our final ring size of $2^{\wedge 14}$ requires 87% of the DSP48 blocks available on this board's Virtex 7 485T chip. Additionally, we require on-board DDR memory for storage of encrypted variables, and high speed Ethernet and PCI Express (PCIe) interfaces. All these are present on the VC707. Note in this document both PCI and PCIe are used to refer to the PCI Express interface that connects the FPGA board with the PC motherboard when it is hosted in a PC.

4.4.6 FHE Accelerator system architecture

The design goal of our FPGA system was to be able to operate as an attached processor to accelerate the FHE primitive operations in way that allows one to chain together several operations in order to minimize the overhead due to data transfer. An attached processor design was developed in which a software programmable microcontroller would manage I/O communications with the host via Ethernet or PCI memory map, manage on board data storage in the form of an encrypted register file, and manage data transfer to and from the FHE primitive modules in as efficient manner as possible. We decided to use the Xilinx Platform Studio Microblaze soft core processor and AXI4 interconnect architecture to implement the attached FHE processor. We found that this supported Ethernet based operation well. For PCI operation we determined that the Microblaze was not required, so we remove the soft core and the Ethernet controller from the FPGA for those builds.

A high level diagram of the FHE Accelerator Architecture is shown in Figure 23. At the top of the diagram is block labeled "Applications". This represents user applications, such as Matlab, that use the FHE accelerator code. User applications utilize the accelerator by communicating with the FHE Kernel Driver, which exposes a Linux "character device". The FHE character device behaves a bit like a file, or a network device. The user application sends commands and data to the FHE Kernel driver by writing packets of data to the FHE character device, and likewise the user application receives responses and status via data packets from the FHE character device. The packet packed-based communication protocol allows the user application to be agnostic to whether the FHE Accelerator is on a local PCIe bus, on separate network-connected device.

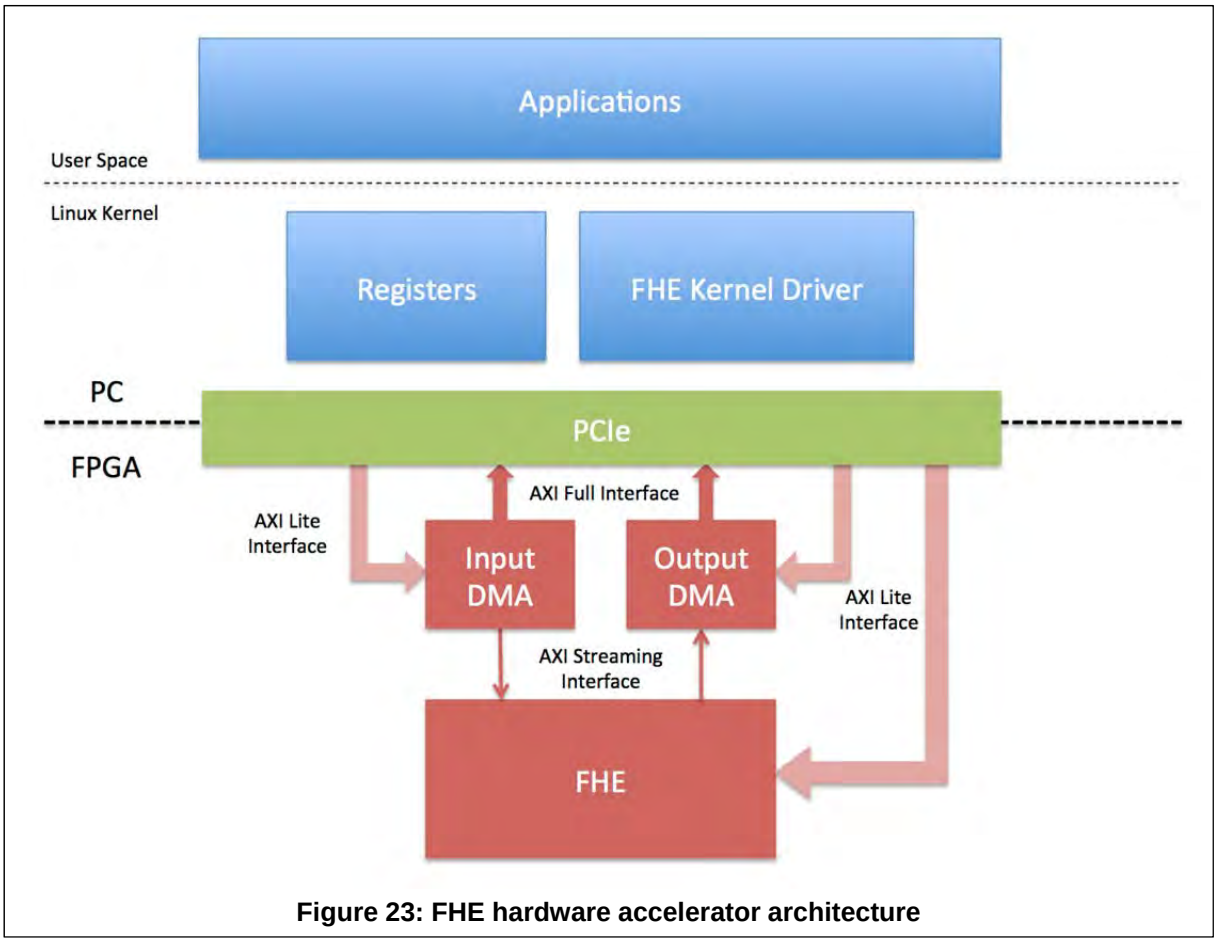


Figure 23: FHE hardware accelerator architecture

The core FHE Kernel Driver code is written in portable "C" code. Therefore, although we run this code in the Linux kernel, it can also run on a soft-core Microblaze processor inside the FPGA. This portable driver code instantiates a set registers, in memory, which store the FHE input and output data, as well as intermediate results. When the code runs on Microblaze, these registers exist in the FPGA-connected DDR memory and when the code runs in the Linux Kernel, these register exist in PC memory. Figure 23 depicts the PCI hosted case where the code runs in the Linux Kernel. For the case where the FPGA is not in a PCI bus, the registers exist on the FPGA evaluation board in DDR3 RAM, and the FHE Kernel driver code is split between the Linux Kernel and the Microblaze on the FPGA. Data packets (described below) are sent via Gigabit Ethernet rather than to mapped PCI memory.

Note that, in Figure 23, the (AXI Full Interface) arrow between the Input DMA module and the PCIe interface originates at the DMA module, even though the data flows in the other direction. This is because the DMA modules are both a Master, meaning they initiate the data transfers. One DMA Master (Input) reads from PC memory, and the other (Output) writes to PC memory. The DMA modules also have an AXI slave port, for control, which is written to and from by the PC (via the PCIe interface).

Figure 24 shows a system block diagram of the FPGA system for both the Ethernet and PCI hosted configurations. The Xilinx platform studio enables us to implement our FHE primitives as

streaming co-processors on the AXI bus. An AXI4 lite bus is used to set control parameters of our Ring operation circuits, such as ring size and tower size.

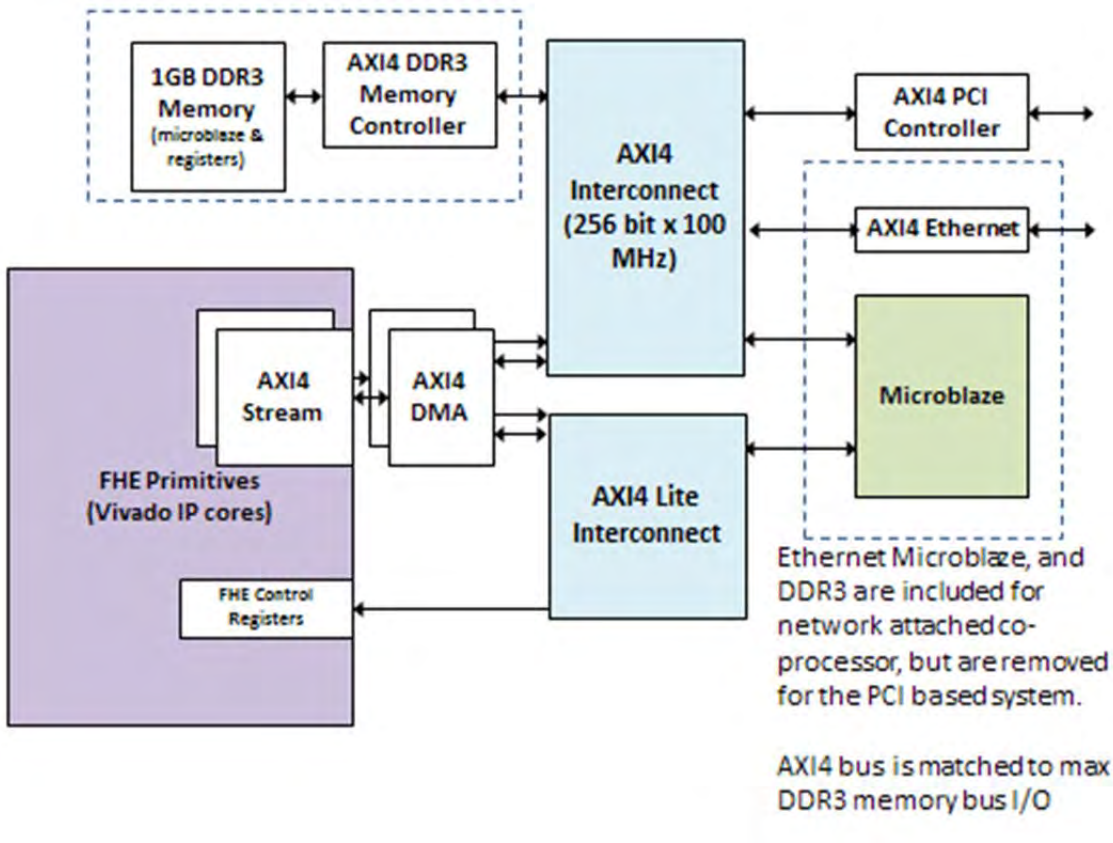


Figure 24: System Block Diagram showing major components and the AXI4 interconnect for the various implementations of the FHEPU.

Figure 25 depicts the interior of the FHE Primitives module (purple) in Figure 24. The module is fed by two 256-bit wide data streams in the AXI clock domain (shown on top), but internally processes two 64-bit streams in a different clock domain called the "math clock" domain. In this design, the math clock runs at 100 MHz, and the AXI clock runs at 125 MHz. Separating the FHE math clock domain from the PCIe clock domain give us quite a bit of design flexibility. The frequency of the AXI clock is determined by the speed of the PCIe interface. The math clock frequency, however, can be set arbitrarily so long as the FHE Core logic meets timing at that frequency. We are conservative with the math clock frequency due to the fact we have some internal (CRT twiddle) memories that are run at twice the math clock frequency (described in Figure 20). As the FHE Core logic eventually is rewritten and becomes more efficient, we will be able to increase the rate of the math clock. The AXI clock may also be doubled if we move to a PCIe Gen 3 architecture.

When running with Ethernet, the main AXI4 interconnects remain a 256 bit bus connecting the DDR3 ram with the various FHE primitives. In this mode, the I/O rate into and out of DDR3 memory limits the overall processing speed of the system.

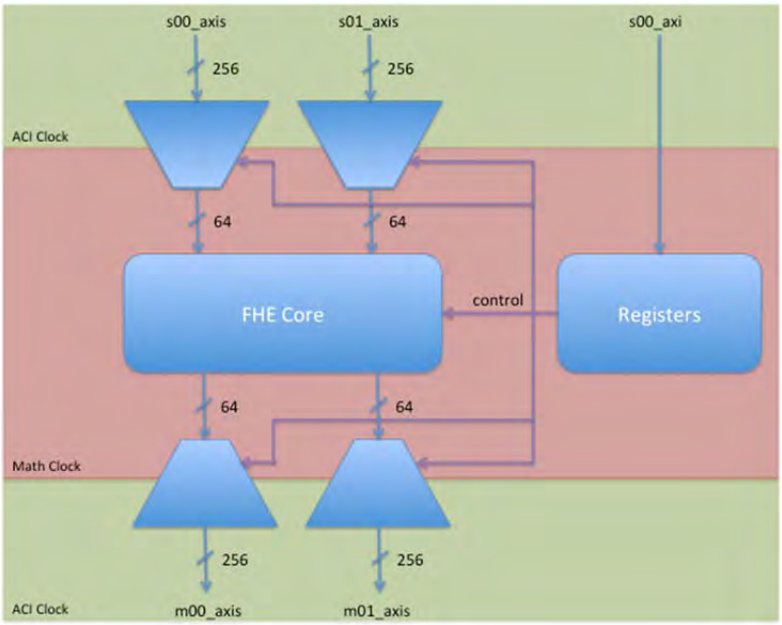


Figure 25: Integration of FHE primitives with the AXI stream data streams.

Figure 26 shows the internal structure of the FHE core and how the inputs and output streams are routed to the various FHE primitive functions. The core block performs one of several operations on the data – Add, Sub, Mul, CRT, or Round. The CRT operation can perform either inverse or forward CRTs of ring sizes from 16 to 16384 (in powers of 2). However, the CRT operation also uses the Mul block. For forward CRTs, the CRT block is fed with input In 0, and the CRT module's output is then multiplied with input In 1 using the Mul module. Similarly, for inverse CRTs, inputs In 0 and In1 are fed into the Mul module, and the Mul output is fed into the CRT module. This architecture conserves FPGA resources for a couple reasons. First, FPGA logic resources are conserved since we would otherwise need to instantiate another Mul block inside the CRT module. Second, since the Mul inputs for CRT operations (CRT Twiddle Table) arrive from In 1 (via DMA from DDR3 ram or PC memory depending on hosting), they do not need to be stored locally as ROM tables in the FPGA. This reduces the amount of FPGA block memory used by the design.

Each of the operations in purple receives their input data pipelined first by ring elements, then by tower indices. Thus all input and output for a complete double CRT is streamed in one operation. The CRT modules require slightly different interfaces that change the order of the input and output data as mentioned previously.

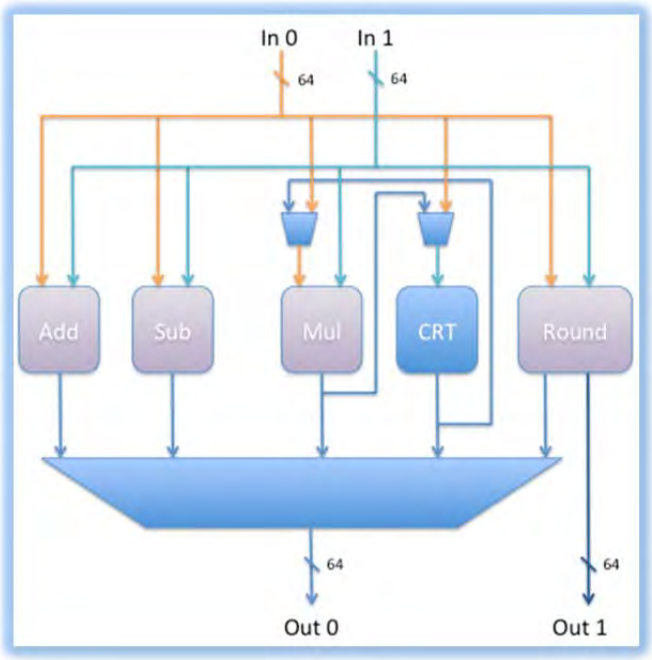


Figure 26: FHE Core. Ring operation pipelines are kept fed with two Input data streams, and produce one or two output data streams, all under direct DMA control from PCI or Microblaze.

4.4.7 Communication between the Application and Kernel Driver.

Ethernet-Hosted Mode

The Xilinx platform studio is used to implement a Microblaze soft core processor when the system is in an Ethernet Hosted configuration,. The system architecture is based on the demo hardware self-test example that is provided with the Xilinx board. The software architecture is based on the web-service example provided with the Xilinx Virtex 6 ML605 evaluation kit, updated with the Xilinx SGMII 144 Ethernet controller (our first implementations, as we later moved to the Virtex 7 and the VC707 evaluation board). The software controlling the system on the Microblaze is written in C code. The system is multithreaded to allow the use of Ethernet TCP/IP socket I/O. A network thread manages socket level I/O between the host and the attached processor. Another thread reads the incoming messages from the socket, parses the commands received and dispatches execution to various subroutines.

The DDR3 ram is partitioned into a set of register data structures, as well as a set of internal registers to store constants used in our encryption schemes. Each register can hold one encrypted bit in the form of 2 dimensional vectors of unsigned long longs that are allocated out of DDR ram. One dimension (the fastest index) is the ring size *N* and is software programmable. The other dimension, the tower size, varies with the state of the register. Typically registers are loaded into the FHE coprocessor with a fixed starting number of the tower elements (up to MAX_TOWER_SIZE = 32 elements.

The registers are allocated out of heap in the DDR3 ram. There are three flavors of registers: Input, Output and Scratch. This design decision was made in order to allow us to later segregate

I/O and scratch registers into different memory locations if that were to increase throughput. The quantity of each register type is software defined at compile time but there is usually a small numbers of Input and Output registers and as many Scratch registers as will utilize all the available heap space. Control structures mark the current tower size of each register, and if the register is used or not. Registers are allocated so they are aligned to 32 byte address boundaries in order to allow the AXI4 DMA engines to move the register data into and out of the FHE primitives. This format allows the contents of an entire register (all used towers) to be streamed with only one DMA transfer.

A Linux Driver is used to interface with the user Application code. It translates the Application Level text interface messages to a binary format message (both to be described later) and handles all Ethernet I/O to the FPGA board.

PCI-Hosted Mode

The Linux Driver for the PCI Hosted mode is written to emulate the buffer I/O of the Ethernet interface, allowing the same user application software to be used for both Ethernet and PCI operation.

In this mode, the DDR3 is not used at all, and both the Microblaze and Ethernet controllers are removed from the system build to conserve resources. Registers are allocated out of the PCI Kernel memory. All Microblaze code is executed in Linux Kernel Space.

Application Level Text Interface

The communication protocol between the user application code and Linux Driver is message based. The messages are in ASCII. Each processor instruction is then parsed. The parsing test starts with a keyword that defines the rest of the instruction format. The keywords are shown in Table 6.

Table 6: Application Level Control Protocol keywords

Keyword	Function
LOAD	Transfer the contents of the message (ascii) into a particular Input register.
GET	Request the contents of a particular output register to be loaded into a message buffer and sent back to the host.
STATUS	Generates a short report on the FPGA board console for debugging showing the contents of all used registers, a listing of the current program loaded.
PROG	Loads a sequence of operations to be performed on the register data, in a simple assembly language.
RUN	Starts a software Finite State Machine to run the stored program to completion.
CRT, ICRT, CEM	A single command that will LOAD two registers, perform a forward CRT, inverse CRT or Composed EvalMult on them and GET the resulting output. Used for accelerating applications that only require these three operations.
RESET	Resets the system to its original state.

The user application can string commands together to program the FPGA to operate on several pieces of encrypted data in the form of an assembly language. The FPGA accelerator's assembly language has the syntax shown in Table 7.

Table 7: Table of available Opcodes for Application program

Opcode	Example	Description
LOAD	R1 = LOAD(In0)	Moves data from an input register to scratch register, all active tower elements are moved.
STORE	Out4 = STORE(R3)	Moves data from a scratch register to output register, all active tower elements are moved.
RADD	R2 = RADD(R3, R4)	Sets up DMAs of the two input and one output registers to the RingAdd

		circuit. All active tower elements are processed I one large data flow.
RSUB	R2 = RSUB(R3, R4)	Sane as RingAdd, except the RingSub circuit is the target of the DMAs.
RMUL	R2 = RMUL(R3, R4)	Sane as RingMul, except the RingSub circuit is the target of the DMAs.
CRT	R3= CRT(R1, R2)	Same as RingAdd, except the input and output registers are used as endpoints for pairs of DMA transfers, each moving one half of the ring data. Note second input register is used as a scratch register so is contents are destroyed.
ICRT	R2 = ICRT(R4, R5)	Same as CRT except an inverse CRT circuit is used.
EMULC	R2 = EMULC(R3, R4)	Executes a ComposedEvalMult, in software which in turns executes several Ring primitives (see below). Note that output register is one tower smaller than the input registers.

An example simple program in now given in Table 8. The program first moves encrypted data from input register 0, to scratch register 0, then repeats the process for a second input variable to register 1. It then computes a RingAdd, RingSub and RingMul using the two inputs, and storing the result in scratch registers 2, 3 and 4 respectively. It then stores those three results in output registers 0, 1 and 2 respectively.

Table 8: Sample FHEPU program

```
R0 = LOAD(In0)
R1 = LOAD(In1)
R2 = RADD(R0,R1)
R3 = RSUB(R0,R1)
R4 = RMUL(R0,R1)
Out0 = STORE(R2)
Out1 = STORE(R3)
Out2 = STORE(R4)
```

Typical system operation would be for the user to execute two LOAD commands to load the contents of input registers 0 and 1 with encrypted data (the encryption being done on the secure host). The user then executes a RUN command to allow the Homomorphic operations to be run on the unsecure FPGA processor. Then subsequent calls to GET commands will transfer the resulting encrypted result data back to the host. Finally decryption would be done on the secure host.

4.4.8 Communication between the Kernel Driver and the FPGA

The FHE Kernel driver delegates certain primitive operations to the FPGA. This FPGA control is made possible by a set of AXI-LITE register-based interfaces, which exist in the FPGA, but are memory-mapped (made accessible) to the FHE Kernel Driver (on the PC) via the PCIe interface. The FPGA Kernel driver configures the FHE module in the FPGA for a particular operation by writing to its AXI-LITE interface. Next, the input and output DMA modules are configured by the FHE Kernel Driver via their AXI-LITE interfaces. At this point, the input DMA module fetches input data directly from PC memory via the PCIe interface and sends the data to the FHE module. The FHE module processes the data, and sends its output to the output DMA module. The output DMA module writes the data directly back to PC memory via the PCIe interface.

The driver communicates with the FPGA either over Ethernet or PCIe by sending and receiving packets. Each packet starts with a 32-byte header that contains 8 4-byte fields as shown in Table 9.

Table 9: Driver Packet header format

Field	Size	Description
totalLength	32-bits	Contains the total length of the packet, in bytes
cmd	32-bits	Defines the command (packet type)
aux1	32-bits	Command-Specific data
aux2	32-bits	Command-Specific data
aux3	32-bits	Command-Specific data
aux4	32-bits	Command-Specific data
aux5	32-bits	Command-Specific data
aux6	32-bits	Command-Specific data

All 32-bit numeric fields are in little endian byte order. The first two header fields are used by every packet, while the next 6 fields (aux1-aux6) are interpreted differently depending on the command. Some packets also contain data following the header. If a packet contains more data following the header, its totalLength field will be greater than the header size, otherwise it will be equal to the header size. Packets from the driver back to the user application start with the same packet header, however have different values in the "cmd" field. The "cmd" field may have the following values as shown in Table 10.

Table 10: Driver Packet command enumeration

Command	Enumer-ation	Description
LOAD	0	This command instructs the driver to load a register with a set of values. The aux1 field contains the index of the register to load, and the aux2 field contains the tower index within that register. The number of values to load is determined by the number of bytes left in the packet. The remaining packet bytes contain a set of 64-bit little endian encoded integers. The user application should always load tower index 0 first, as the number of elements in tower index 0 determines the ring size. After tower index 0 of a register is loaded, the driver will not allow other tower indices to be loaded unless the number of element to be loaded is equal to the ring size (defined by the number of elements in tower index 0). The ring size my be changed by re-loading tower index 0.
PROG	1	The driver may execute a set of operations on behalf of the user application. These operations are defined by a small "program" sent to the driver. Each instruction in the program is encoded by a 32-bit instruction, and this command is used to load the program – one instruction at a time. The aux1 field contains the instruction index (where it resides in the program), with the first instruction residing at index 1. The aux2 field contains the instruction itself. The driver keeps track of how many instructions were written so it knows how many instructions to execute. An existing program may be cleared by writing a new instruction to index 1.
GET	2	This command causes the driver to send back the contents of an output register, (with a respGET packet, described below). The aux1 field contains the output register number, and the aux2 field contains the tower number.
RUN	3	This command causes the driver to execute the program, set by one or more PROG commands (described above).
VERIFY	4	This command causes the driver verify the contents of an output register. The aux1 field contains the output register index, and the aux2 field contains the tower index. The packet data (following the packet header), is compared to the register contents by the driver to see if it matches. The driver will then send back a respVerify packet back to the user application to let the application know if the data matched or not. If the data size in this packet does not match the register's ring size, the driver outputs an error message to its standard output, but does not send a response packet
DUMP	5	This is a debug command that causes the driver to dump the contents of a register to its standard output. The aux field contains the register index, and the aux2 field contains the tower index.

HINT	6	This command is used to load a hint register. The aux1 field contains the hint register index, and the aux2 field contains the tower index.
respGET	8	This response, sent from the driver back to the user application, is a the response to a GET command. Its header is followed by the data in the register requested by a GET command from the application. The aux1 field is set to the register index.
respVERIFY	9	This response, sent from the driver to the user application to inform the user application the result of a VERIFY command. The aux1 field contains 0 if the register contents matched the data in the VERIFY command, and 1 if the data did not match.

The user application my send the driver a small program, using the PROG and RUN commands, described above. Each program instruction is encoded with a 32-bit little-endian word, using the format described in Table 11.

Table 11: Driver Packet instruction encoding

Bits	Field Name	Description
24-31	Opcode	Encodes the type of operation to be performed. See Table below for a description of the available operations.
23-22	Return Value Type	Encodes the register type of the return value.
16-22	Return Value Index	Encodes the register number of the return value.
14-15	Argument 2 Type	Encodes the register type of the second argument.
8-13	Argument 2 Index	Encodes the register number of the second argument.
7-6	Argument 1 Type	Encodes the register type of the first argument.
5-0	Argument 1 Index	Encodes the register number of the first argument.

There are four possible values for the register type fields. These are outlined in Table 12:

Table 12: Driver Packet register encoding

Register Type	Enumer-ation	Description
None	0	This encoding is used if an argument does not refer to a register. Its interpretation would depend on the opcode.
Input	1	Input registers are filled with data from LOAD commands – i.e. with data from user applications.
Output	2	Output registers are sent to the user application as responses to GET commands. They typically contain the results of operations.
Scratch	3	Scratch registers are contain the inputs and output of FPGA operations.

The opcodes are summarized in Table 13 below. Note that some encodings have been omitted. This is because some opcodes were defined but never implemented. The following table only contains the implemented opcodes:

Table 13: Driver Packet opcode encoding

Opcode	Enumer-ation	Description
LOAD	0	Move the content of an input register to a scratch register. Argument 1 contains the input register address, and the return value contains the scratch register address.
STORE	1	Move the contents of a scratch register to an output register. Argument 1 contains the scratch register address, and the return value contains output register address.
RADD	5	Perform the ADD operation on two scratch registers. Arguments 1 and 2 contain the two input scratch register addresses for the operation, and the return value contains the address of the output register.
RSUB	6	Perform the SUB operation on two scratch registers. Arguments 1 and 2 contain the two input scratch register addresses for the operation, and the return value contains the address of the output register.
RMUL	7	Perform the MUL operation on two scratch registers. Arguments 1 and 2 contain the two

		input scratch register addresses for the operation, and the return value contains the address of the output register.
CRT	8	Perform the CRT operation on a scratch register. Arguments 1 contains the input scratch register addresses for the operation, and the return value contains the address of the output register.
ICRT	9	Perform the ICRT operation on a scratch register. Arguments 1 contains the input scratch register addresses for the operation, and the return value contains the address of the output register.
ROUND 1	10	Perform the ROUND operation on a scratch register. Arguments 1 contains the input scratch register addresses for the operation, and argument 2 contains the tower index (register type None). The return value contains the address of the output register.
ROUND 2	11	Same as the ROUND 1 operation, but returns the second result (the ROUND operation returns to results)
EMULC	12	Perform the EMULC operation on two scratch registers. Arguments 1 and 2 contain the two input scratch register addresses for the operation, and the return value contains the address of the output register.
END	13	Stop executing the program
NOP	14	No-operation. Does no work.
MOD	15	Perform the CRT operation on a scratch register. Arguments 1 contains the input scratch register addresses for the operation, and the return value contains the address of the output register.

4.4.9 Performance results

During our evaluations of FPGA acceleration of the KWS application we noticed that RingAdd does not actually benefit from acceleration, simply because our C implementation is very efficient, and the additional overhead of data transfer to and from the FPGA eliminates any potential benefit. Our accelerations primarily benefit any functions using CRT. We focus primarily on the CRT and Bootstrap (which has only its CRTs accelerated).

We evaluated the performance of our FPGA co-processor implementation from multiple perspectives. In particular, we evaluated both the individual performance improvements of the CRT operation running in isolation on the FPGA and running the CRT in the context of a broader use-case for encrypted KWS.

The CRT text search operation was run to compare 1) a native interpreted Matlab runtime of the CRT running on the CPU of the FPGA systems, 2) a mex (i.e. compiled Matlab) version of the CRT running on the PROCEED Deathstar environment (64 cores) and 3) with calls the FPGA co-processor to run the CRT operation. All of the experiment test harnesses were run in interpreted Matlab with the variable being how the CRT was executed. We did not include FPGA setup time in our analysis. We did include the time to submit the CRT jobs to the FPGA, and the response times in our experimental analysis.

Our experimental results on CRT runtimes for the Matlab interpreted, Matlab CPU compiled and FPGA-supported CRT runtimes for various ring dimensions can be seen in Figure 27. We found experimentally that the value of the ciphertext moduli has little impact on runtime, as long as the modulus is smaller than the maximum value supported by the CRT.

In this figure we see that there is a multiple order of magnitude performance improvement by offloading CRT computations from the CPU to the FPGA. Note that the curves have a distinct "J" shape with slightly increased runtimes at smaller ring dimensions on the FPGA. This is a result of the FPGA communication times and is a common artifact of FPGA co-processor behavior. An additional view of the FPGA speed-up as compared to the Matlab and mex

runtimes can be seen in Figure 28. It can be clearly seen that the FPGA implementation is 1600 times faster than our fastest single CPU version. It is 35 times faster than the Deathstar version.

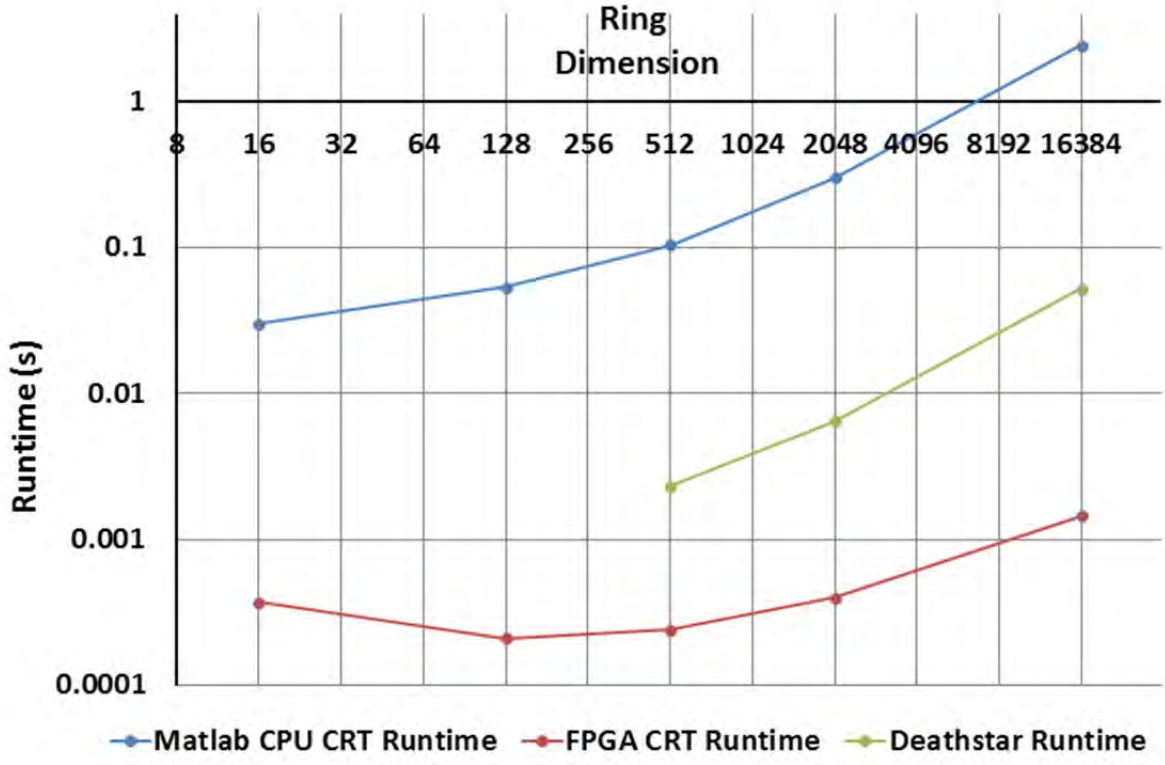


Figure 27: CRT absolute runtimes.

We attempted to use the FPGA to support our bootstrapping operation, but we ran into challenges due to parameter selection issues. In particular, we found that in order to support the necessary depth of computation and larger plaintext moduli needed for bootstrapping, we needed to support parameters with bit widths larger than 64 bits for the most secure ring sizes. This would have required rewriting all our FPGA code, which was not possible within the remaining project scope. However, in software we took a straightforward approach to implementation and attempted an FPGA implementation that supported solely the CRT operations on the FPGA, with smaller ring sizes that could still be supported by our initial <=64 bit parameter settings. Because we limited ourselves to a lower ciphertext modulus than necessary, there remains further research needed to fully push this capability into operational production. This "parameter engineering" for bootstrapping will be early low-hanging fruit for any future development effort.

We collected initial results to evaluate bootstrapping performance with only the CRT accelerated on the FPGA. The speedup can be seen in the graph in Figure 29. Figure 30 shows the speedup for bootstrapping as a function of ring dimension. We see a less dramatic performance improvement as compared to solely running the CRT on the FPGA as in earlier experimental results due to the increased amount of non-FPGA computation running on bootstrapping. Many of the operations we are still running on the CPU could also be out-sourced to an FPGA as additional low-hanging fruit for future development work.

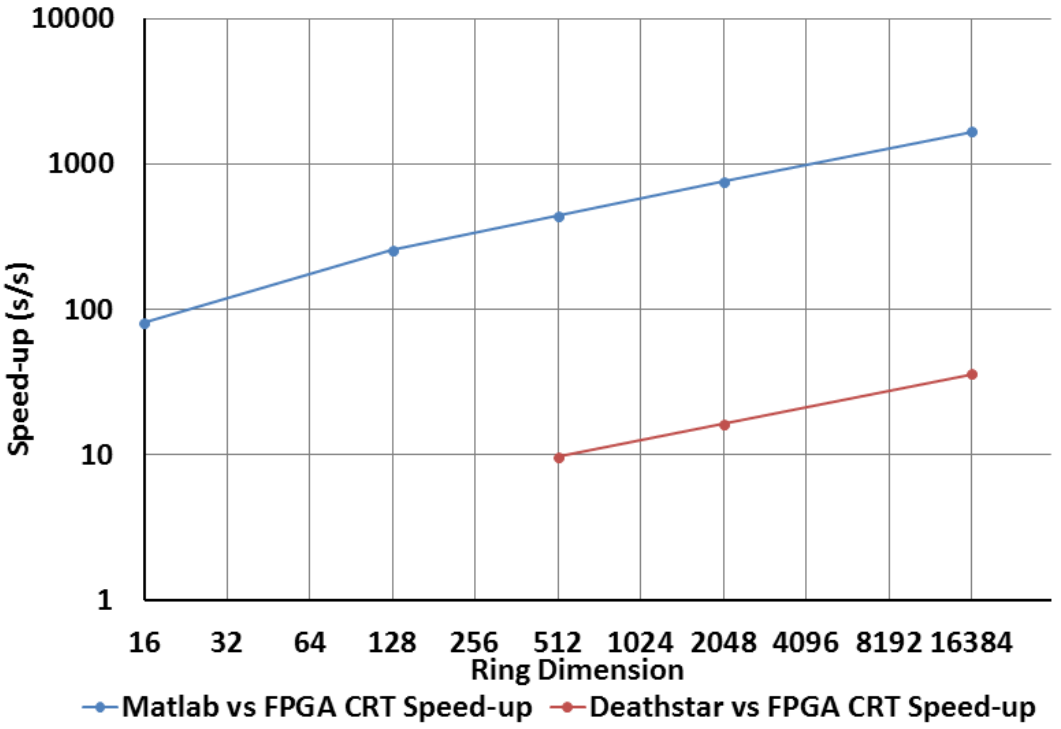


Figure 28: CRT relative speedup

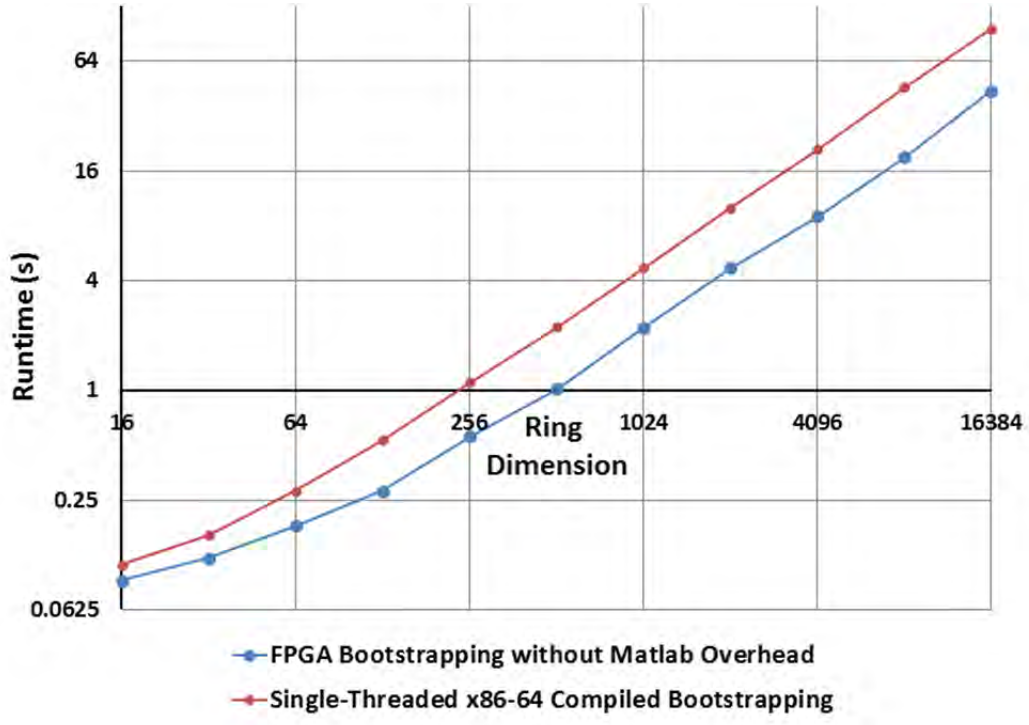


Figure 29: Bootstrapping runtime performance with the CRT running on the FPGA.

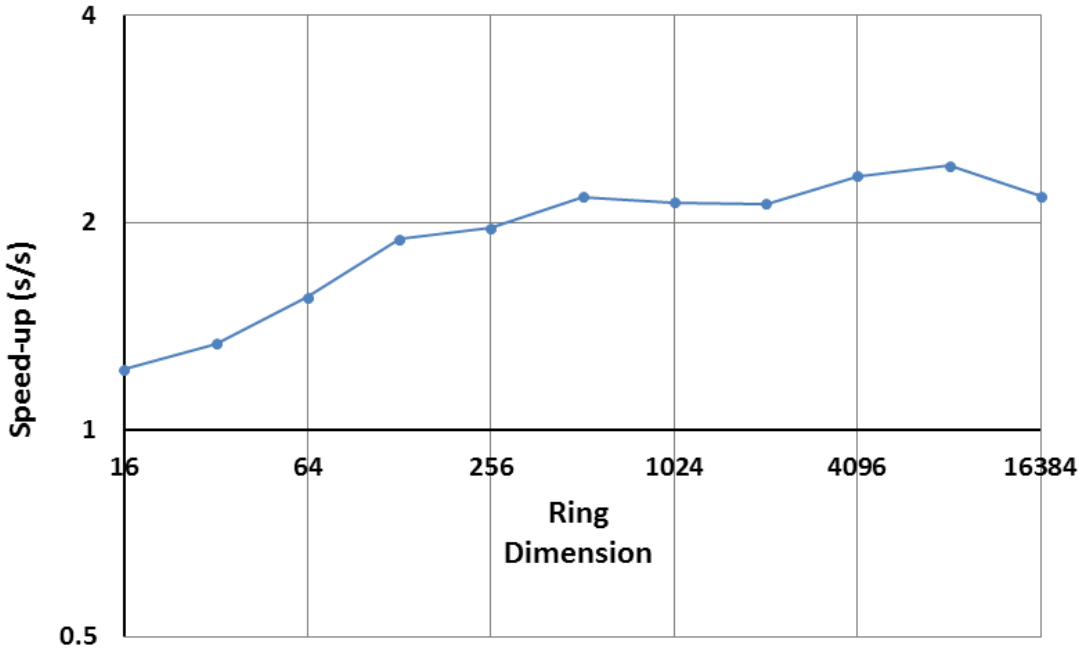


Figure 30: Bootstrapping runtime speedup with the CRT running on the FPGA.

4.5 Encrypted keyword search FHE results and discussion

We initially run our KWS implementation at a low security level ($\delta = 1.08$) using SHE computing to enable the email system to be interactive with fast response times. Our initial implementation uses a ring dimension of $n = 512$ and encrypts emails with a supported depth of computation $d = 12$. This will result in an effective ciphertext modulus q represented with 430 bits and provides a security level of less than AES-128 in terms of work-factor.

To evaluate the relative performance speed-up of the FPGA co-processor with only CRT acceleration in a more real-world setting, we developed both compiled and FPGA accelerated implementations of our string searching algorithms.

The compiled implementation supported all string search operations in mex Matlab, a C-compiled version of Matlab. The FPGA co-processor implementation ran almost all of the string search operations in the slower native Matlab interpreter, except for the CRT operation which were run on the FPGA co-processor.

To set a baseline of performance, we ran the runtime experiments on a ring dimension of n=2048, text corpuses of length 1,4, 16,64 and 256 characters long (corresponding to the length of modern text messages) and keywords of length 1,2,3,4 and 5 characters long. We supported all of ASCII in our search operations. We selected n=2048 because the runtime of the compiled string search at higher dimensions was too long to collected meaningful data over.

Figure 31 and Figure 32 show the runtime of the string search operation for the compiled string search. As we can see, the runtime grows nearly linearly with both the keyword length and corpus length.

Figure 33 and Figure 34 show the runtime of the string search operation for the FPGA string search. As we can see, the runtime grows nearly linearly with both the keyword length and corpus length, similar to with the compiled version of the operation, but substantially faster.

We created plots to show the relative speed-up of using the FPGA co-processor in Figure 35 and Figure 36. We see that the speed-up is relatively constant at 50x (1.5 orders of magnitude) despite variations in the length of keywords and corpora. We found that this 50X speed-up was relatively consistent across ring dimensions in broader experiments at lower security levels.

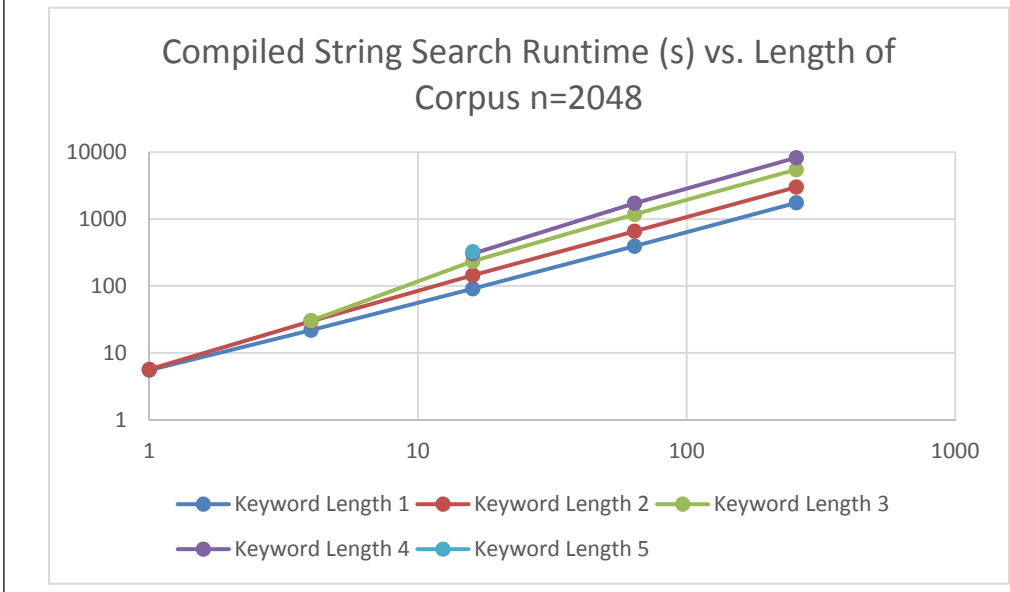


Figure 31: Runtime of the compiled string search operation vs. length of corpus

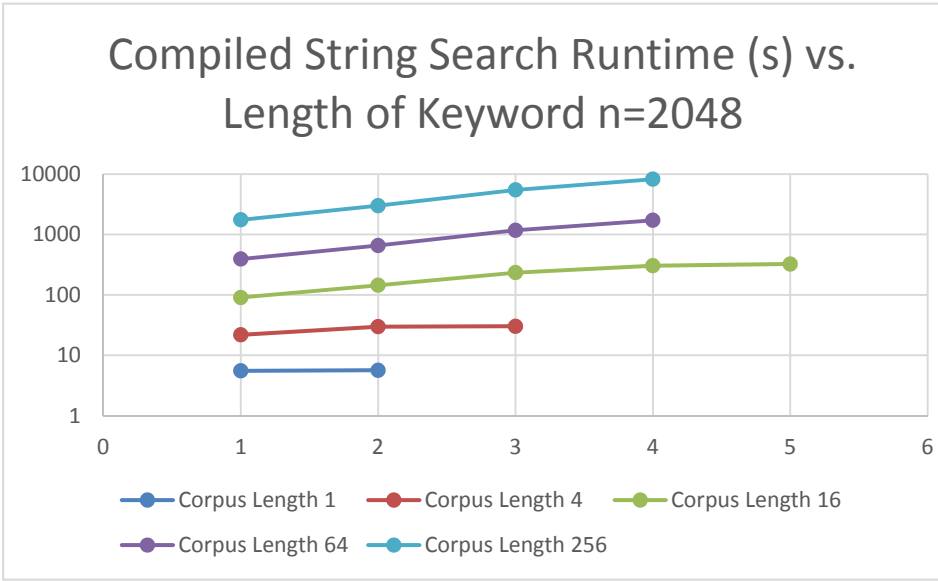


Figure 32: Runtime of the compiled string search operation vs. length of keyword

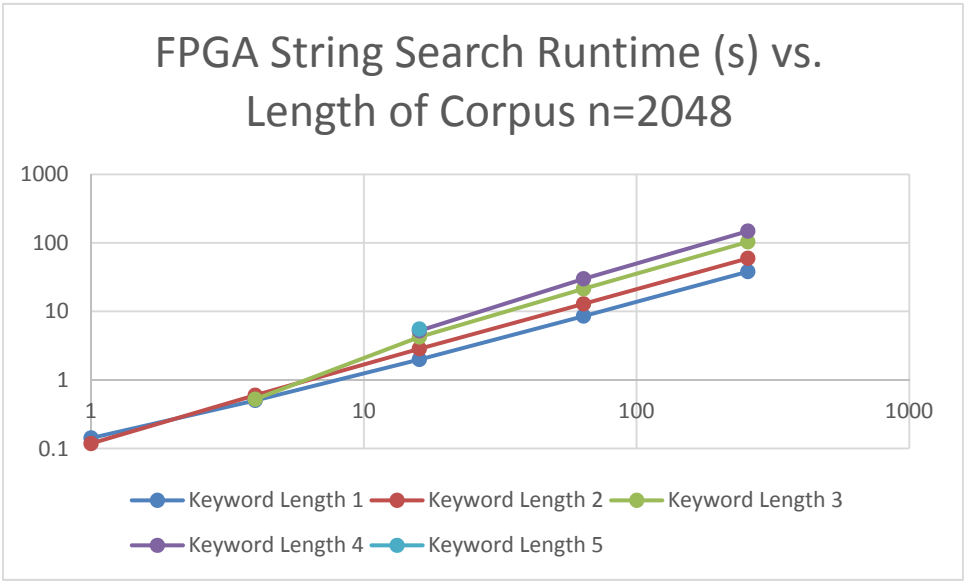


Figure 33: Runtime of the FPGA accellerated string search operation vs. length of keyword

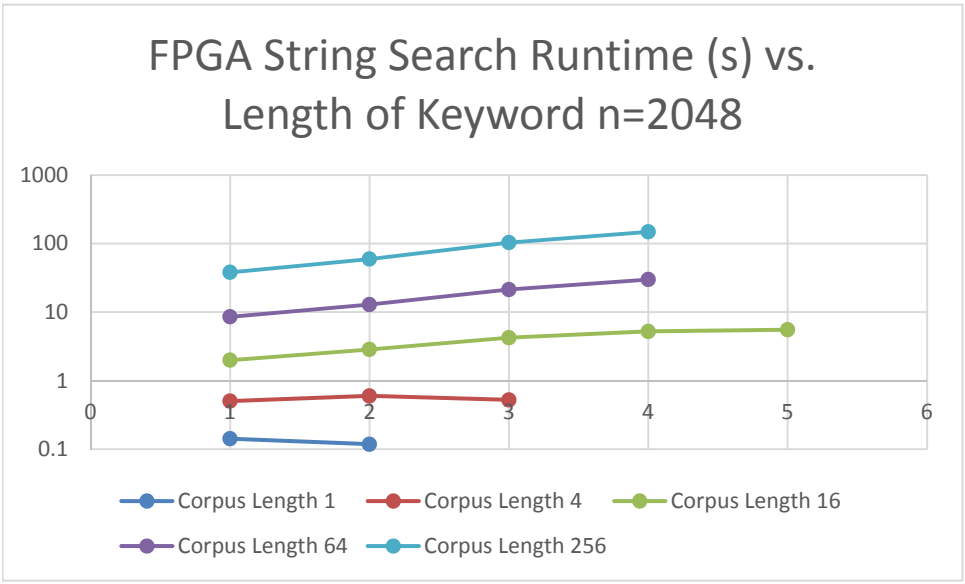


Figure 34: Runtime of the FPGA accellerated string search operation vs. length of corpus

Our intentions are to continue to evaluate the performance of the string search operations. For example, we plan on generating more data on the experiments we have run on larger keyword and corpus sets. We plan on generating data over more parameter settings to assess how ring dimension affects the FPGA co-processor in comparison to the compiled operations. For example, we will assess performance up to the n=16384 ring dimension and up to 10 character keywords and 1000 character text corpora. Early indications are that ring dimension has a lesser impact on the runtime of FPGA-based co-processor and we expect this to be much faster.

We plan on generating these results and preparing them for publication in an IEEE Transactions-type journal in the near future.

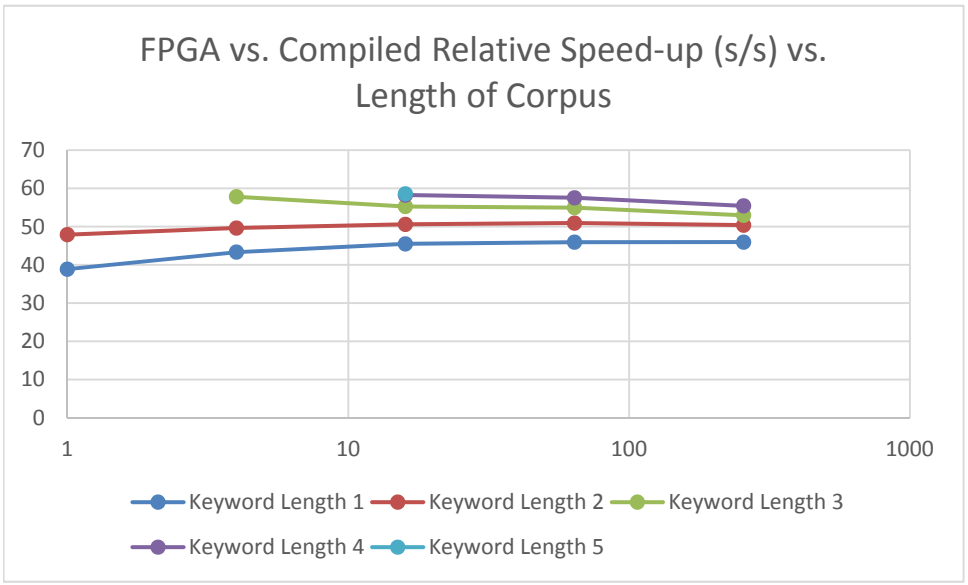


Figure 35: Relative speed-up of string search operation vs. length of keyword

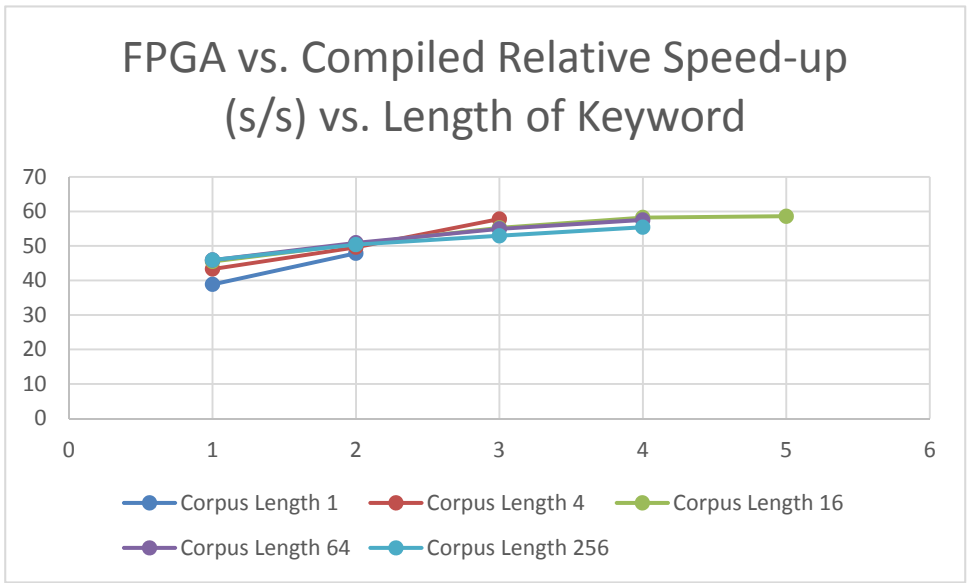


Figure 36: Relative Relative speed-up of string search operation vs. length of keyword

5 CONCLUSIONS

In conclusion we have taken FHE from an early theoretical possibility to an actual implementation on multiple platforms. Our three pronged approach of theory, software and hardware implementation has had many benefits we did not see at the outset. By working closely with newly developed FHE schemes, we were able to continuously improve the capabilities of our FHE scheme. At the onset we did not have a bootstrapping approach, yet eventually this

capability was not only developed but proved to be one of the most efficient approaches available. By using a rapid prototyping approach to software development, we were able to quickly implement working prototypes of our FHE scheme, and develop an understanding of where the strengths and weaknesses were in our implementations. We were able to apply software engineering to a relatively new set of algorithms, determining appropriate ways to parallelize our operations in order to achieve speedup on multicore computers. Our approach also allowed us to use state of art code generation approaches to generate embedded C code for use on iPhone and iPod systems from the same code we used for prototyping. Targeting hardware for our implementation drove our development in some fundamental ways. We knew from the onset that we needed efficient implementations of large integer modulus arithmetic. This helped drive our double CRT implementation, which keeps bit widths within reasonable limits. As a side result we found that such implementations improved the performance of our software as well.

Beyond our VoIP functionality, our implementation is part of a long-term community vision to support a general, practical and secure computing capability through a layered services architecture. Part of our vision is to provide software interfaces in our design for our highly optimized implementations of the basic FHE operations for both general and specific applications.

Although we only utilize limited-depth SHE for VOIP, that encryption system design is a scaled-down version of our FHE scheme design. Our design offers the possibility for a much more general VoIP teleconferencing capability that incorporates signal detection and noise filtering operations on the encrypted VoIP channels. This more general design would enable protection against some of the more practical attacks that could be made by an adversary such as noise injection attacks where an adversary inserts noise into a VoIP teleconferencing session to reduce the ability of participants to hear one another. Using more general FHE capabilities, we could enable the untrusted cloud host to securely filter the encrypted VoIP signals before or after mixing to reduce the impacts of insertion attacks.

A further aspect of our layered architecture vision is ability to mix-and-match a computing substrate at the server for much larger scalability and throughput. Our FHE design has been ported to an FPGA, but with further refinement could be amenable to GPUs [14] operating as FHE co-processors.

6 RECOMMENDATIONS

Recommendations for future work in this area are organized into three areas. The first area for future work is research into application specific uses of FHE. One noticeable area that did not get sufficient attention during our program was in the areas of applications. Initial attempts to identify applications of FHE had usually been selected by the Proceed researchers as being potential areas for both FHE and SMP computation. However, we often found that the security model for the application, while good for SMP did not make sense for the postulated cloud based FHE implementation. It is apparent to us, that as FHE becomes more usable through research and development that streamlines the implementation and executions costs, that more applications will become apparent. We also found from our research, that often times, a customized, application specific data representation and FHE parameter selection was more conducive to FHE manipulation (such as our VOIP application).

The second area for future research is in the area of FHE acceleration. Our approach of using Matlab and Simulink to rapidly prototype both VHDL and C code implementations of key FHE functions was vital to our being able to rapidly prototype and develop FHE accelerators of key FHE functions. However, we find now that as FHE technology matures, these key functions have been relatively stable. It is now time that native implementations of both C/C++ and VHDL of these functions will allow developers to achieve even more efficiencies in implementation. Further, the dramatic improvements in GPU technology over the last few years have brought wider bit width operations closer to reality. GPUs have a deeper penetration into the cloud computing commodity market than FPGAs currently have, and as such, may prove a valuable implementation platform for acceleration of FHE. As with many complex systems, there are numerous improvements that could be made to the FPGA based FHE Accelerator:

- We originally chose the largest tower size of 32 before we actually had a viable Bootstrap operation derived. We have since determined that we don't need that many tower entries for FHE at high security levels. Currently we are running the internal twiddle table block RAM at twice the frequency of the rest of the logic. Through a combination of decreasing the number of tower vectors down from 32, and using a larger size FPGA chip with more block RAM resources, we will no longer need to run the block RAM at twice the frequency of the other logic, and we will be able to increase the clock frequency of the other logic.
- Currently, the ring math logic is all internally gated by a clock enable signal. This allows the data to be fed though ring math logic at the rate at which it arrives to the FPGA from the PCIe interface. However, by updating the logic that drives the ring math logic so that it appropriately buffers data and sends it though the ring logic without any breaks, we would negate the need for the clock enable signals. This would further allow us to increase the ring math clock frequency since the large fan out of the clock enable signal complicates placement and routing.
- The system currently runs 8–lane x8 PCIe Gen1, which provides approximately 2 Gb/sec per lane for a total 16 Gb/sec throughput. If we move to PCIe Gen2, the throughput would double to 4 Gb/sec per lane. The current hardware supports Gen2 PCIe, however the PCIe IP block we used did not. However, it should be possible to move to Gen2 PCIe by either upgrading the FPGA tools, or by developing our own PCIe interface logic.

The third area for future work is in the development of a portable, extensible library of FHE operations that are easily adapted by users unfamiliar with FHE details. Such a library should be developed using state of the art programming practices, and be able to support a wide range of platforms: single and multi-core systems, hardware accelerators such as FPGAs and GPUs, and distributed cloud-type enterprise systems. This would enable developers to develop on one platform then scale their work easily to large compute environments.

7 REFERENCES

[1] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2009, aAI3382729.

[2] ——, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[3] D. Micciancio, "Lattice-based cryptography," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 713–715.

[4] C. Gentry and S. Halevi, “Implementing Gentry’s fully homomorphic encryption scheme,” in *Advances in Cryptology EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. Paterson, Ed. Springer Berlin / Heidelberg, 2011, vol. 6632, pp. 129–148.

[5] Z. Brakerski and V. Vaikuntanathan, “Efficient fully homomorphic encryption from (standard) lwe,” *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

[7] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical gapsvp,” in *Advances in Cryptology– CRYPTO 2012*. Springer, 2012, pp. 868–886.

[8] A. L´opez-Alt, E. Tromer, and V. Vaikuntanathan, “On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption,” in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 1219–1234.

[9] C. Gentry, A. Sahai, and B. Waters, “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based,” in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[10] C. Gentry, S. Halevi, and N. P. Smart, “Homomorphic evaluation of the aes circuit,” in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 850–867.

[11] M. Naehrig, K. Lauter, and V. Vaikuntanathan, “Can homomorphic encryption be practical?” in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, ser. CCSW ’11. New York, NY, USA: ACM, 2011, pp. 113–124. [Online]. Available: http://doi.acm.org/10.1145/2046660.2046682

[12] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, “Batch fully homomorphic encryption over the integers.” in *EUROCRYPT*, vol. 7881. Springer, 2013, pp. 315–335.

[13] Y. Dor¨oz, Y. Hu, and B. Sunar, “Homomorphic aes evaluation using ntru.” *IACR Cryptology ePrint Archive*, vol. 2014, p. 39, 2014.

[14] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, “Accelerating fully homomorphic encryption on GPUs,” in *Proceedings of the IEEE High Performance Extreme Computing Conference*, 2012.

[15] C. Gentry and S. Halevi, “HElib,” https://github.com/shaih/HElib, 2014.

[16] L. Ducas and D. Micciancio, “Fhew: Bootstrapping homomorphic encryption in less than a second,” in *Advances in Cryptology–EUROCRYPT 2015*. Springer, 2015, pp. 617–640.

[17] D. Wu and J. Haven, “Using homomorphic encryption for large scale statistical analysis,” 2012.

[18] J. Alperin-Sheriff and C. Peikert, “Practical bootstrapping in quasilinear time,” in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 1–20.

[19] J. Hoffstein, J. Pipher, and J. H. Silverman, “NTRU: A ring-based public key cryptosystem,” in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, J. P.

Buhler, Ed. Springer Berlin Heidelberg, 1998, vol. 1423, pp. 267–288. [Online]. Available: http://dx.doi.org/10.1007/BFb0054868

[20] C. Gentry, S. Halevi, and N. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology CRYPTO 2012*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds. Springer Berlin / Heidelberg, 2012, vol. 7417, pp. 850–867.

[21] H. Perl, M. Brenner, and M. Smith, "Poster: an implementation of the fully homomorphic Smart-Vercauteren cryptosystem," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 837–840. [Online]. Available: http://doi.acm.org/10.1145/2093476.2093506

[22] MAGMA, "V2.18-11," 2012, available at http://magma.maths.usyd.edu.au/magma/.

[23] V. Shoup, *NTL: A Library for doing Number Theory*, Courant Institute, New York University, New York, NY, 2012, available at http://shoup.net/ntl/.

[24] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (srtp)," 2004.

[25] P. Zimmermann, A. Johnston, and J. Callas, "Zrtp: Media path key agreement for secure rtp," *draft-zimmermann-avt-zrtp-04 (work in progress)*, 2007.

[26] R. Zhang, X. Wang, R. Farley, X. Yang, and X. Jiang, "On the feasibility of launching the man-in-the-middle attacks on voip from remote attackers," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 61–69. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533069

[27] A. Ornaghi and M. Valleri, "Man in the middle attacks," in *Blackhat Conference Europe*, 2003.

[28] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. N. Gansterer, "Security and usability aspects of man-in-the-middle attacks on zrtp." *J. UCS*, vol. 14, no. 5, pp. 673–692, 2008.

[29] S. Capkun, L. Butty´an, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 1, pp. 52–64, 2003.

[30] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Rfc 3261: Sip: session initiation protocol," 2003.

[31] K. Rohloff and D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on NTRU," in *Proceedings of the 2nd Workshop on Applied Homomorphic Cryptography (WAHC)*, 2014.

[32] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 7073. Springer, 2011, pp. 1–20.

[33] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *CT-RSA*, 2011, pp. 319–339.

[34] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post Quantum Cryptography*. Springer, February 2009, pp. 147–191.

[35] J. van de Pol, "Quantifying the security of lattice-based cryptosystems in practice," in *Mathematical and Statistical Aspects of Cryptography*, 2012.

[36] MATLAB, *R2014b*. Natick, Massachusetts: The MathWorks Inc., 2014.

[37] D. Butenhof, *Programming with POSIX (R) threads*. Addison-Wesley Professional, 1997.

[38] D. B. Cousins, K. Rohloff, C. Peikert, and R. Schantz, "SIPHER: Scalable implementation of primitives for homomorphic encryption - FPGA implementation using Simulink," in *Fifteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, ser. HPEC '11, 2011.

[39] ——, "An update on scalable implementation of primitives for homomorphic encryption - FPGA implementation using Simulink," in *Sixteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, ser. HPEC '12, 2012.

[40] *ibid* [18] [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40041-4 1

[41] A. L´opez-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC*, 2012, pp. 1219–1234.

[42] C. Gentry, S. Halevi, V. Lyubashevsky, C. Peikert, J. Silverman, and N. Smart, 2011, personal communication.

[43] C. Gentry, S. Halevi, and N. Smart. "Homomorphic evaluation of the AES circuit." In Reihaneh Safavi-Naini and Ran Canetti, editors, Advances in Cryptology CRYPTO 2012, volume 7417 of Lecture Notes in Computer Science, pages 850–867. Springer Berlin / Heidelberg, 2012.

[44] H, Cohen A Course in Computational Algebraic Number Theory. New York: Springer-Verlag, 1993.

[45] [Online] http://gmplib.org/ last accessed May 14, 2012.

[46] Ookla, 2014. [Online]. Available: https://www.ookla.com/

[47] Amazon AWS Instance Types, 2014. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[48] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, pp. 39–53, 2004.

[49] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in Advances in CryptologyCRYPTO93. Springer, 1994, pp. 232–249.

[50] A. D. Keromytis, "A survey of voice over ip security research," in *Information Systems Security*. Springer, 2009, pp. 1–17.

[51] ——, "Voice-over-ip security: Research and practice," *Security & Privacy, IEEE*, vol. 8, no. 2, pp. 76–78, 2010.

[52] V. Kumar, M. Korpi, and S. Sengodan, *IP Telephony with H.323: Architectures for Unified Networks and Integrated Services*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[53] Z. Anwar, W. Yurcik, R. E. Johnson, M. Hafiz, and R. H. Campbell, "Multiple design patterns for voice over ip (voip) security," in *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*. IEEE, 2006, pp. 8–pp.

[54] J. F. Ransome, C. CISM, J. Rittinghouse *et al.*, *Voice over Internet Protocol (VoIP) Security*. Digital Press, 2005.

[55] D. C. Sicker and T. Lookabaugh, "Voip security: Not an afterthought," *Queue*, vol. 2, no. 6, pp. 56–64, Sep. 2004. [Online]. Available: http://doi.acm.org/10.1145/1028893.1028898

[56] J. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Stam, Ed. Springer Berlin Heidelberg, 2013, vol. 8308, pp. 45–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45239-0 4

[57] J. Cheon, J.-S. Coron, J. Kim, M. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully homomorphic encryption over the integers," in *Advances in Cryptology EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, T. Johansson and P. Nguyen, Eds. Springer Berlin Heidelberg, 2013, vol. 7881, pp. 315–335. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38348-9 20

[58] Y. Doroz, Y. Hu, and B. Sunar, "Homomorphic aes evaluation using ntru," Cryptology ePrint Archive, Report 2014/039, 2014, http://eprint.iacr.org/.

[59] L. Liu, M. Fukumoto, and S. Saiki, "An improved mu-law proportionate nlms algorithm," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3797–3800.

[60] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*. Springer, 1999, pp. 223–238.

[61] J. Launchbury, D. Archer, T. DuBuisson, and E. Mertens, "Application-scale secure multiparty computation," in *Programming Languages and Systems*, ser. Lecture Notes in Computer Science, Z. Shao, Ed. Springer Berlin Heidelberg, 2014, vol. 8410, pp. 8–26. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54833-8 2

[62] S. Goldwasser, "Multi party computations: past and present," in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*. ACM, 1997, pp. 1–6.

[63] P. L. Montgomery "Modular Multiplication Without Trial Division", Mathematics of Computation Vol. 44, No. 170 (Apr., 1985), pp. 519-521, American Mathematical Society.

[64] P.D. Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," Advances in Cryptology — CRYPTO'86, Springer, 1986.

[65] V. Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings". In Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, chapter 1, pages 1–23. Springer Berlin / Heidelberg, Berlin.

[66] L. R. Rabiner and B. Gold. Theory and Application of Digital Signal Processing. Prentice-Hall, Inc., 1975.

[67] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster Interleaved Modular Multiplication Based on Barrett and Montgomery Reduction Methods", IEEE Transactions on Computers, Vol. 59, No. 12, Dec 2010.

[68] M.Arioua, S. Belkouch, M. Agdad, M. M. Hassani, “VHDL implementation of an optimiz ed 8-point FFT/IFFT processor in pipeline architecture for OFDM systems” 2011 International Conference on Multimedia Computing and Systems (ICMCS), 7-9 April 2011

8 BIBLIOGRAPHY

Publications from PROCEED SIPHER team from project start			DISTAR or CFR
Title	Authors	Venue	
SIPHER: Scalable Implementation of Primitives for Homomorphic EncRyption FPGA implementation using Simulink	David Bruce Cousins, Kurt Rohloff, Chris Peikert, Rick Schantz	HPEC 2011	17712
Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller	C. Peikert, D. Micciancio	Eurocrypt 2012	CFR
Pseudorandom Functions and Lattices,	Chris Peikert, A. Banerjee and A. Rosen	Eurocrypt 2012	CFR
Field Switching in BGV-Style Homomorphic Encryption	C. Gentry, S. Halevi, C. Peikert, N. Smart	SCN 2012	CFR
An update on Scalable Implementation of Primitives for Homomorphic EncRyption FPGA implementation using Simulink.	David Bruce Cousins, Kurt Rohloff, Chris Peikert, Rick Schantz	HPEC 2012	19495
Identity-Based (Lossy) Trapdoor Functions and Applications	Mihir Bellare and Eike Kiltz and Chris Peikert and Brent Waters	Eurocrypt 2012	CFR
A Toolkit for Ring-LWE Cryptography	Vadim Lyubashevsky, Chris Peikert, Oded Regev	Eurocrypt 2013	CFR
Classical Hardness of Learning with Errors	Zvika Brakersi, Adeline Langlois, Chris Peikert, Oded Regev, Damien Stehle	STOC 2013	CFR
How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE	Rikke Bendlin, Sara Krehbiel, Chris Peikert	ACNS 2013	CFR
Practical Bootstrapping in Quasilinear Time	Jacob Alperin-Sheriff, Chris Peikert	CRYPTO 2013	CFR
Hardness of SIS and LWE with Small Parameters	Daniele Micciancio, Chris Peikert	CRYPTO 2013	CFR
Accelerating Computations on Encrypted Data with an FPGA	David Bruce Cousins, Kurt Rohloff	Mathworks Technical Newsletter, Sep 2013	21346

An FPGA Coprocessor Implementation of Homomorphic EncRyption.	David Bruce Cousins, John Golusky, Daniel Sumorok, Kurt Rohloff	HPEC 2014	22818
Computing with Data Privacy: steps toward realization	David W. Archer Kurt Rohloff	IEEE Security & Privacy 1 (2015): 22-29.	23391
Scalable, Practical VOIP Teleconferencing with End-to-End Homomorphic Encryption	Kurt Rohloff, David Bruce Cousins, Daniel Sumorok	NDSS 2015	23281
A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU	Kurt Rohloff, David Bruce Cousins	WAHC 2014	20224

A.1 Appendix SIPHER SHE runtime data tables

Table 14: Encryption runtime (mSec) vs. depth of computation supported and Ring Dimension for p = 2

Depth Dim.	1	3	5	7	8	11	13	15	17	19
512	2.32	2.83	2.86	3.27	3.39	3.25	4.38	4.64	5.35	5.66
1024	3.87	5.33	5.17	5.98	5.68	5.63	6.94	8.4	9.04	9.2
2048	6.26	6.48	7.01	7.47	7.94	8.78	12.7	13.03	13.05	14.52
4096	12.08	12.27	13.04	14.87	17.38	17.65	20.73	17.46	21.57	22.13
8192	24.53	25.18	26.13	29.07	30.81	32.15	34.43	32.46	36.16	37.9
16384	52.3	55.02	58.05	59.71	60.29	61.98	63.44	64.99	69.96	72.89

Table 15: EvalAdd runtime (mSec) vs. depth of computation supported and Ring Dimension for p = 2

Depth Dim.	1	3	5	7	9	11	13	15	17	19
512 .	0.21	0.32	0.42	0.54	0.64	0.73	1.26	2.11	2.90	3.12
1024	0.3	1.04	0.47	0.57	0.72	0.74	1.4	2.72	2.85	2.93
2048	0.37	0.45	0.55	0.67	0.8	1	1.97	3	3.04	3.24
4096	0.56	0.65	0.74	0.91	1.92	2.07	2.25	2.43	3.73	3.54
8192	0.89	1.01	1.2	1.36	2.46	2.7	3.69	3.23	5.05	5.44
16384	1.58	1.82	2.12	2.39	3.99	4.19	4.27	4.77	7.16	7.29

Table 16: ComposedEvalMult runtime (mSec) vs. depth of computation and Ring Dimension for p = 2

Depth Dim..	1	3	5	7	9	11	13	15	17	19
512	16.03	22.73	23.32	22.65	22.87	22.96	24.35	25.24	25.37	25.78
1024	29.15	37.85	39.05	39.11	38.79	39.24	39.49	39.59	39.52	39.68
2048	49.17	66.31	66.77	67.41	67.15	68.38	68.22	69.27	69.45	71.09
4096	99.56	140.42	140.71	141.42	141.26	142.75	143.52	145.51	144.61	148.31
8192	196.83	279.37	280.42	284.4	283.98	285.69	289.59	286.55	292.69	295.69
16384	463.92	623.19	622.74	628.87	630.43	633.37	639.52	642.8	651.2	659.88

Table 17: Decryption runtime (mSec) vs. depth of computation supported and initial Ring Dimension for p = 2

Depth Dim	1	3	5	7	9	11	13	15	17	19

512	0.40	0.26	0.13	0.14	0.10	0.10	0.06	0.06	0.06	0.06
1024	0.87	0.38	0.18	0.11	0.11	0.11	0.11	0.11	0.05	0.05
2048	1.92	0.84	0.38	0.38	0.22	0.22	0.22	0.22	0.12	0.12
4096	3.36	1.7	0.84	0.86	0.37	0.39	0.38	0.22	0.22	0.21
8192	7.22	3.43	1.67	1.72	0.85	0.87	0.86	0.87	0.39	0.4
16384	15.36	7.18	3.37	3.37	1.67	1.67	1.67	1.73	0.87	0.85

A.2 Appendix SIPHER PUBLICATIONS

Following this page are full reprints of all SIPHER Publications listed is Section 8

SIPHER: Scalable Implementation of Primitives for Homomorphic EncRyption – FPGA implementation using Simulink

David Bruce Cousins, Kurt Rohloff, Chris Peikert, Rick Schantz
Raytheon BBN Technologies, Georgia Institute of Technology
{dcousins, krohloff, schantz}@bbn.com cpeikert@cc.gatech.edu

Abstract*

Practical Fully Homomorphic Encryption (FHE) would be a game-changing technology to enable secure, general computation on encrypted data, e.g., on untrusted off-site hardware. Recent theoretical breakthroughs demonstrated the existence of Fully Homomorphic Encryption schemes [1,4]. However, FHE remains impractical because current implementations are many orders of magnitude too slow for practical use, and do not scale well to the very large keys and ciphertexts needed to assure a sufficient level of security. A new DARPA program (PROCEED) has as its focus the acceleration of various aspects of the FHE concept toward practical implementation and use.

In this paper we present early work on our SIPHER project, an element of the PROCEED program, whose goal is to demonstrate FHE implementations that improve the state of the art by many orders of magnitude. As part of our activity we are developing a set of hardware primitives to accelerate FHE implementations based on lattice problems [3]. As an important aspect of our design methodology we use a state of the art tool-chain offered by the Mathworks to develop FPGA circuits from Simulink Models. We initially develop prototype descriptions in Matlab that we re-implement in a stream oriented, hardware implementable manner in Simulink. The operations of the implementations are compared to verify correctness. A conversion from Simulink to VHDL is done in a completely automated fashion using Mathwork's HDL coder. This tool chain provides us the means to develop our primitives, including cyclic VHDL based FPGA prototyping, much faster than traditional methods.

Fully and Somewhat Homomorphic Encryption

Fully Homomorphic Encryption (FHE) holds the promise to securely run arbitrary computations over encrypted data on untrusted computation hosts [4]. The general FHE concept of operations is that sensitive data is encrypted with a public key, then sent to an untrusted computation host, which can perform arbitrary computations on the encrypted data without first needing to decrypt it. It has been shown to be theoretically possible to evaluate arbitrary programs using just two special purpose FHE operations, EvalAdd and EvalMult, which roughly correspond to bitwise XOR and AND gates operating on encrypted bits. A sequence of these operations is run against the encrypted data, resulting in an encryption of the output of the original program run on the unencrypted data. This encrypted result can then be sent back to the original client, who decrypts the result using its secret key. The encrypted data is protected at all times with reasonable security guarantees based on computational hardness results.

FHE enables more secure and private computation, but to be effective there needs to be multiple orders of magnitude efficiency improvements before it can be practical. Known FHE schemes are highly inefficient partly because they are "noisy" - the encryption schemes' ciphertext is a function not only of the plaintext and encryption keys but also of a noise term. The amount of noise in a ciphertext rapidly increases as the EvalAdd and EvalMult operations are performed, and after too many such operations there is too much noise to correctly decrypt the ciphertext. To run larger numbers of EvalAdd and EvalMult operations, FHE schemes typically address the accumulation of noise with a very computationally expensive "recryption" operation that is periodically run on intermediate ciphertexts to keep the noise at a level that still permits decryption.

A Somewhat Homomorphic Encryption (SHE) scheme supports several (but not unlimited) EvalMult and EvalAdd operations while preserving the correctness of decryption. In other words, SHE can schemes support secure computation for only a small subset of programs. Our development approach is to select an efficient implementation of an SHE scheme which can be converted into a full FHE scheme with the addition of a recryption (noise reduction) operation and/or other supporting modifications. This enables us to incrementally develop SHE results using modest initial resources.

Although there have been some initial FHE implementations [1], there have been no practical implementations that can be used for effective general computation. Current designs of FHE schemes rely on operations (i.e. modular arithmetic with an enormous modulus) that are inefficient on standard CPU architectures and which are too memory intensive. For convenience all of these previous implementations have been limited by their focus on CPUs and do not take advantage of specialized parallel computation hardware like FPGAs.

* Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

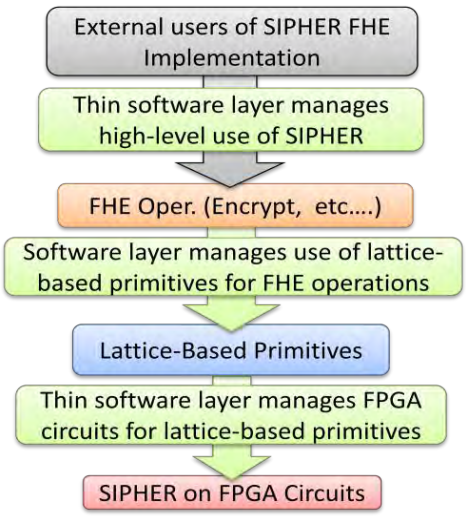


Figure 1: Conceptual diagram of system.

Figure 1 shows our vision for the layered services we provide in our FHE implementation. There are software interfaces for implementations of the basic FHE operations (KeyGen, Encrypt, EvalAdd, EvalMult, Recrypt, Decrypt) as a primitive basis for constructing more general applications on encrypted data. Our approach to the FHE primitives is based on the highly efficient lattice-based techniques developed by one of our investigators [3], which can be implemented with only a handful of core mathematical primitive operations (see Figure 2). Many of these operations are closely related to well-understood operations, such as Fast Fourier Transforms, which we are targeting for efficient implementations on FPGAs. The EvalAdd and EvalMult operations for example are simply element wise vector adds and multiplies taken modulo some particular prime integer q. These are trivial to express using Matlab: c = mod(a+b, q) and c = mod(a.*b, q).

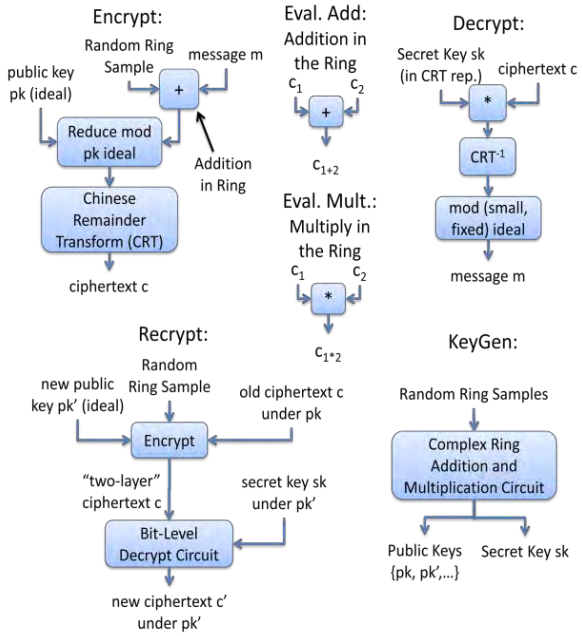


Figure 2: Primitives for a SHE scheme.

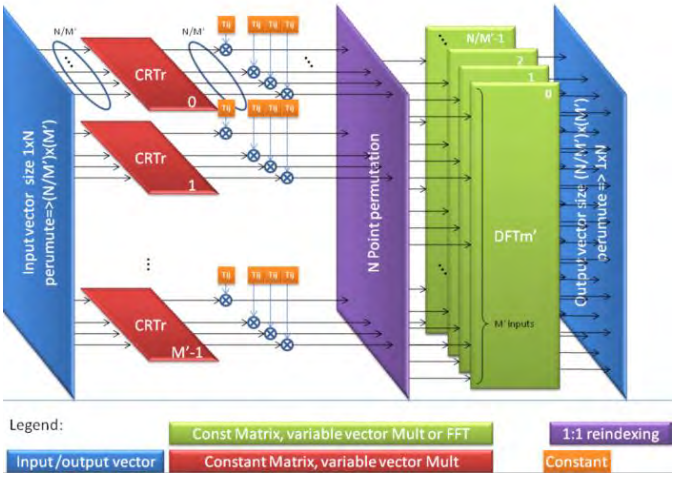


Figure 3: Internal Structure of CRT Primitive showing similarity to signal processing data flow.

We are leveraging previous work on signal processing implementations to implement the primitives (and consequently the FHE scheme) as circuits on FPGAs. The FPGAs provide highly cost-effective parallelism.

One of our primary primitive operations is the Chinese Remainder Transform (CRT). The CRT is mathematically similar to the Discrete Fourier Transform, but implemented using modular integer (instead of complex) arithmetic. Figure 3 shows the CRT implementation we are working with that is structurally very similar to the familiar processing of multi-dimensional signal data. The similarities of our primitives with well understood signal-processing operations that have been efficiently implemented in FPGAs give us confidence toward developing efficient and scalable FPGA implementations of the primitives.

The FFT operation in Figure 4 is similar to the standard FFT [2], except all operations are done in modulo q arithmetic. We were able to take Mathwork's example Simulink streaming FFT model (Figure 4), slightly modify the ordering of the output, and easily change from complex to integer arithmetic simply by altering the input and twiddle factor data types. Converting to modular arithmetic is also straightforward.

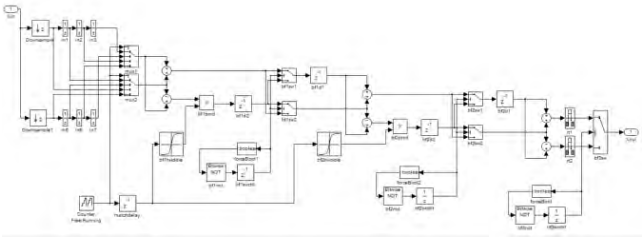


Figure 4: Simulink model for streaming FFT.

To implement the modular arithmetic efficiently in hardware we have taken advantage of the Montgomery Reduction method [5], which allows one to express mod q operations in a larger basis r, which can be a power of two. So while the bits required to represent the integers have grown, all arithmetic operations now are allowed to wrap around on overflow, eliminating the need to do a costly modular reduction operation in the hardware. We implement the Montgomery reduction method in Simulink using the fixed point tool box (Figure 5). The additional complexity this adds to the Simulink model for our ring operations is trivial. Figure 6 shows the Montgomery reduction steps in Red and Blue. Red steps convert to the Montgomery space, and are done once for each input. Any number of additions can be done without the need for a reduction step. Each multiply requires one reduction step. A final reduction step converts back into the original mod q representation. Our modified FFT implementation requires pre-computation of the twiddle factors in Montgomery representation (no real time impact), one reduction step for each input sample, one reduction for each output sample and one reduction at the output of each butterfly multiplication. Since all reduction is done using a pipelined approach, there is no additional computation time added (just latency).

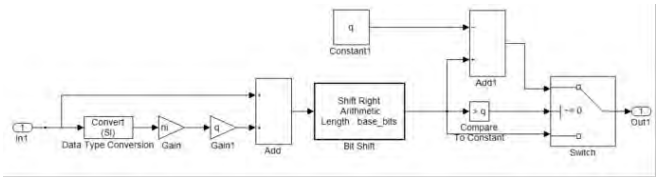


Figure 5: Simulink model for Montgomery Reduction

Interim Results

Our presentation will include examples of our primitives coded in Matlab and Simulink and examples of VHDL code generated by the HDL coder. We will also be able to show timing results from Modelsim based simulations of the resulting code.

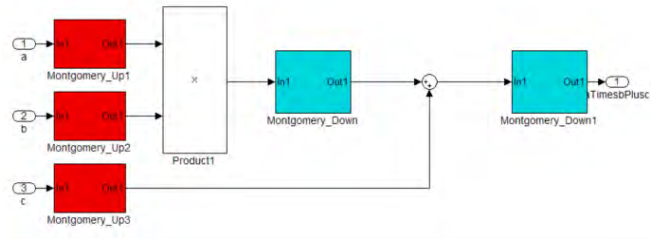


Figure 6: Simulink model for Ring Multiply-Add

References

[1] C. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic encryption scheme. In Kenneth Paterson, editor, Advances in Cryptology – EUROCRYPT 2011, volume 6632 of Lecture Notes in Computer Science, chapter 9, pages 129–148. Springer, 2011.

[2] W. M. Gentleman and G. Sande, "Fast Fourier transforms—for fun and profit," Proc. AFIPS 29, 563–578 (1966).

[3] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, *volume 6110 of Lecture Notes in Computer Science*, chapter 1, pages 1–23. Springer Berlin / Heidelberg, Berlin,

[4] D. Micciancio. A first glimpse of cryptography's Holy Grail. Comm. ACM 53, 3 (March 2010), 96-96.

[5] Peter L. Montgomery "Modular Multiplication Without Trial Division", *Mathematics of Computation* Vol. 44, No. 170 (Apr., 1985), pp. 519-521, American Mathematical Society.

Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller

Daniele Micciancio* Chris Peikert†

September 14, 2011

Abstract

We give new methods for generating and using "strong trapdoors" in cryptographic lattices, which are simultaneously simple, efficient, easy to implement (even in parallel), and asymptotically optimal with very small hidden constants. Our methods involve a new kind of trapdoor, and include specialized algorithms for inverting LWE, randomly sampling SIS preimages, and securely delegating trapdoors. These tasks were previously the main bottleneck for a wide range of cryptographic schemes, and our techniques substantially improve upon the prior ones, both in terms of practical performance and quality of the produced outputs. Moreover, the simple structure of the new trapdoor and associated algorithms can be exposed in applications, leading to further simplifications and efficiency improvements. We exemplify the applicability of our methods with new digital signature schemes and CCA-secure encryption schemes, which have better efficiency and security than the previously known lattice-based constructions.

1 Introduction

Cryptography based on lattices has several attractive and distinguishing features:

- On the *security* front, the best attacks on the underlying problems require exponential $2^{\Omega(n)}$ time in the main security parameter n, even for quantum adversaries. By constrast, for example, mainstream factoring-based cryptography can be broken in subexponential $2^{\tilde{O}(n^{1/3})}$ time classically, and even in polynomial $n^{O(1)}$ time using quantum algorithms. Moreover, lattice cryptography is supported by strong worst-case/average-case security reductions, which provide solid theoretical evidence that the random instances used in cryptography are indeed asymptotically hard, and do not suffer from any unforeseen "structural" weaknesses.

*University of California, San Diego. Email: textttdaniele@cs.ucsd.edu. This material is based on research sponsored by DARPA under agreement number FA8750-11-C-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

†School of Computer Science, College of Computing, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under Grant CNS-0716786 and CAREER Award CCF-1054495, by the Alfred P. Sloan Foundation, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Sloan Foundation, DARPA or the U.S. Government.

- On the *efficiency* and *implementation* fronts, lattice cryptography operations can be extremely simple, fast and parallelizable. Typical operations are the selection of uniformly random integer matrices $\mathbf{A}$ modulo some small $q = \mathrm{poly}(n)$, and the evaluation of simple linear functions like

$$f_{\mathbf{A}}(\mathbf{x}) := \mathbf{A}\mathbf{x} \bmod q \quad \text{and} \quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$$

  on short integer vectors $\mathbf{x}$, $\mathbf{e}$.[1] (For commonly used parameters, $f_{\mathbf{A}}$ is surjective while $g_{\mathbf{A}}$ is injective.) Often, the modulus q is small enough that all the basic operations can be directly implemented using machine-level arithmetic. By contrast, the analogous operations in number-theoretic cryptography (e.g., generating huge random primes, and exponentiating modulo such primes) are much more complex, admit only limited parallelism in practice, and require the use of "big number" arithmetic libraries.

In recent years lattice-based cryptography has also been shown to be extremely versatile, leading to a large number of theoretical applications ranging from (hierarchical) identity-based encryption [GPV08, CHKP10, ABB10a, ABB10b], to fully homomorphic encryption schemes [Gen09b, Gen09a, vGHV10, BV11b, BV11a, GH11, BGV11], and much more (e.g., [LM08, PW08, Lyu08, PV08, PVW08, Pei09b, ACPS09, Rüc10, Boy10, GHV10, GKV10]).

Not all lattice cryptography is as simple as selecting random matrices $\mathbf{A}$ and evaluating linear functions like $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$, however. In fact, such operations yield only collision-resistant hash functions, public-key encryption schemes that are secure under passive attacks, and little else. Richer and more advanced lattice-based cryptographic schemes, including chosen ciphertext-secure encryption, "hash-and-sign" digital signatures, and identity-based encryption also require generating a matrix $\mathbf{A}$ together with some *"strong" trapdoor*, typically in the form of a nonsingular square matrix (a basis) $\mathbf{S}$ of short integer vectors such that $\mathbf{A}\mathbf{S} = \mathbf{0} \bmod q$. (The matrix $\mathbf{S}$ is usually interpreted as a basis of a lattice defined by using $\mathbf{A}$ as a "parity check" matrix.) Applications of such strong trapdoors also require certain efficient inversion algorithms for the functions $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$, using $\mathbf{S}$. Appropriately inverting $f_{\mathbf{A}}$ can be particularly complex, as it typically requires sampling *random preimages* of $f_{\mathbf{A}}(\mathbf{x})$ according to a Gaussian-like probability distribution (see [GPV08]).

Theoretical solutions for all the above tasks (generating $\mathbf{A}$ with strong trapdoor $\mathbf{S}$ [Ajt99, AP09], trapdoor inversion of $g_{\mathbf{A}}$ and preimage sampling for $f_{\mathbf{A}}$ [GPV08]) are known, but they are rather complex and not very suitable for practice, in either runtime or the "quality" of their outputs. (The quality of a trapdoor $\mathbf{S}$ roughly corresponds to the Euclidean lengths of its vectors — shorter is better.) The current best method for trapdoor generation [AP09] is conceptually and algorithmically complex, and involves costly computations of Hermite normal forms and matrix inverses. And while the dimensions and quality of its output are *asymptotically* optimal (or nearly so, depending on the precise notion of quality), the hidden constant factors are rather large. Similarly, the standard methods for inverting $g_{\mathbf{A}}$ and sampling preimages of $f_{\mathbf{A}}$ [Bab85, Kle00, GPV08] are inherently sequential and time-consuming, as they are based on an orthogonalization process that uses high-precision real numbers. A more efficient and parallelizable method for preimage sampling (which uses only small-integer arithmetic) has recently been discovered [Pei10], but it is still more complex than is desirable for practice, and the quality of its output can be slightly worse than that of the sequential algorithm when using the same trapdoor $\mathbf{S}$.

More compact and efficient trapdoors appear necessary for bringing advanced lattice-based schemes to practice, not only because of the current unsatisfactory runtimes, but also because the concrete security of lattice cryptography can be quite sensitive to even small changes in the main parameters. As already

[1] Inverting these functions corresponds to solving the "short integer solution" (SIS) problem [Ajt96] for $f_{\mathbf{A}}$, and the "learning with errors" (LWE) problem [Reg05] for $g_{\mathbf{A}}$, both of which are widely used in lattice cryptography and enjoy provable worst-case hardness.

mentioned, two central objects are a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that serves as a public key, and an associated secret matrix $\mathbf{S} \in \mathbb{Z}^{m \times m}$ consisting of short integer vectors having "quality" s, where smaller is better. Here n is the main security parameter governing the hardness of breaking the functions, and m is the dimension of a lattice associated with $\mathbf{A}$, which is generated by the vectors in $\mathbf{S}$. Note that the security parameter n and lattice dimension m need not be the same; indeed, typically we have $m = \Theta(n \lg q)$, which for many applications is optimal up to constant factors. (For simplicity, throughout this introduction we use the base-2 logarithm; other choices are possible and yield tradeoffs among the parameters.) For the trapdoor quality, achieving $s = O(\sqrt{m})$ is asymptotically optimal, and random preimages of $f_{\mathbf{A}}$ generated using $\mathbf{S}$ have Euclidean length $\beta \approx s\sqrt{m}$. For security, it must be hard (*without* knowing the trapdoor) to find any preimage having length bounded by β. Interestingly, the computational resources needed to do so can increase dramatically with only a moderate decrease in the bound β (see, e.g., [GN08, MR09]). Therefore, improving the parameters m and s by even small constant factors can have a significant impact on concrete security. Moreover, this can lead to a "virtuous cycle" in which the increased security allows for the use of a smaller security parameter n, which leads to even smaller values of m, s, and β, etc. Note also that the schemes' key sizes and concrete runtimes are reduced as well, so improving the parameters yields a "win-win-win" scenario of simultaneously smaller keys, increased concrete security, and faster operations. (This phenomenon is borne out concretely; see Figure 2.)

1.1 Contributions

The first main contribution of this paper is a new method of trapdoor generation for cryptographic lattices, which is simultaneously simple, efficient, easy to implement (even in parallel), and asymptotically optimal with small hidden constants. The new trapdoor generator strictly subsumes the prior ones of [Ajt99, AP09], in that it proves the main theorems from those works, but with improved concrete bounds for all the relevant quantities (simultaneously), and via a conceptually simpler and more efficient algorithm. To accompany our trapdoor generator, we also give specialized algorithms for trapdoor inversion (for $g_{\mathbf{A}}$) and preimage sampling (for $f_{\mathbf{A}}$), which are simpler and more efficient in our setting than the prior general solutions [Bab85, Kle00, GPV08, Pei10].

Our methods yield large constant-factor improvements, and in some cases even small asymptotic improvements, in the lattice dimension m, trapdoor quality s, and storage size of the trapdoor. Because trapdoor generation and inversion algorithms are the main operations in many lattice cryptography schemes, our algorithms can be plugged in as 'black boxes' to deliver significant concrete improvements in all such applications. Moreover, it is often possible to expose the special (and very simple) structure of our trapdoor directly in cryptographic schemes, yielding additional improvements and potentially new applications. (Below we summarize a few improvements to existing applications, with full details in Section 6.)

We now give a detailed comparison of our results with the most relevant prior works [Ajt99, AP09, GPV08, Pei10]. The quantitative improvements are summarized in Figure 1.

Simpler, faster trapdoor generation and inversion algorithms. Our trapdoor generator is exceedingly simple, especially as compared with the prior constructions [Ajt99, AP09]. It essentially amounts to just one multiplication of two random matrices, whose entries are chosen independently from appropriate probability distributions. Surprisingly, this method is nearly identical to Ajtai's original method [Ajt96] of generating a random lattice together with a "weak" trapdoor of one or more short vectors (but *not* a full basis), with one added twist. And while there are no detailed runtime analyses or public implementations of [Ajt99, AP09], it is clear from inspection that our new method is significantly more efficient, since it does not involve any expensive Hermite normal form or matrix inversion computations.

Our specialized, parallel inversion algorithms for $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are also simpler and more practically efficient than the general solutions of [Bab85, Kle00, GPV08, Pei10] (though we note that our trapdoor generator is entirely compatible with those general algorithms as well). In particular, we give the first *parallel* algorithm for inverting $g_{\mathbf{A}}$ under asymptotically optimal error rates (previously, handling such large errors required the sequential "nearest-plane" algorithm of [Bab85]), and our preimage sampling algorithm for $f_{\mathbf{A}}$ works with smaller integers and requires much less offline storage than the one from [Pei10].

Tighter parameters. To generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that is within negligible statistical distance of uniform, our new trapdoor construction improves the lattice dimension from $m > 5n \lg q$ [AP09] down to $m \approx 2n \lg q$. (In both cases, the base of the logarithm is a tunable parameter that appears as a multiplicative factor in the quality of the trapdoor; here we fix upon base 2 for concreteness.) In addition, we give the first known *computationally pseudorandom* construction (under the LWE assumption), where the dimension can be as small as $m = n(1 + \lg q)$, although at the cost of an $\Omega(\sqrt{n})$ factor worse quality s.

Our construction also greatly improves the quality s of the trapdoor. The best prior construction [AP09] produces a basis whose Gram-Schmidt quality (i.e., the maximum length of its Gram-Schmidt orthogonalized vectors) was loosely bounded by $20\sqrt{n \lg q}$. However, the Gram-Schmidt notion of quality is useful only for less efficient, sequential inversion algorithms [Bab85, GPV08] that use high-precision real arithmetic. For the more efficient, parallel preimage sampling algorithm of [Pei10] that uses small-integer arithmetic, the parameters guaranteed by [AP09] are asymptotically worse, at $m > n \lg^2 q$ and $s \geq 16\sqrt{n \lg^2 q}$. By contrast, our (statistically secure) trapdoor construction achieves the "best of both worlds:" asymptotically optimal dimension $m \approx 2n \lg q$ and quality $s \approx 1.6\sqrt{n \lg q}$ or better, with a parallel preimage sampling algorithm that is slightly more efficient than the one of [Pei10].

Altogether, for any n and typical values of $q \geq 2^{16}$, we conservatively estimate that the new trapdoor generator and inversion algorithms collectively provide at least a $7 \lg q \geq 112$-*fold improvement* in the length bound $\beta \approx s\sqrt{m}$ for $f_{\mathbf{A}}$ preimages (generated using an efficient algorithm). We also obtain similar improvements in the size of the error terms that can be handled when efficiently inverting $g_{\mathbf{A}}$.

New, smaller trapdoors. As an additional benefit, our construction actually produces a *new kind of trapdoor* — not a basis — that is at least 4 times smaller in storage than a basis of corresponding quality, and is at least as powerful, i.e., a good basis can be efficiently derived from the new trapdoor. We stress that our specialized inversion algorithms using the new trapdoor provide almost exactly the same quality as the inefficient, sequential algorithms using a derived basis, so there is no trade-off between efficiency and quality. (This is in contrast with [Pei10] when using a basis generated according to [AP09].) Moreover, the storage size of the new trapdoor grows only linearly in the lattice dimension m, rather than quadratically as a basis does. This is most significant for applications like hierarchical ID-based encryption [CHKP10, ABB10a] that *delegate* trapdoors for increasing values of m. The new trapdoor also admits a very simple and efficient delegation mechanism, which unlike the prior method [CHKP10] does not require any costly operations like linear independence tests, or conversions from a full-rank set of lattice vectors into a basis. In summary, the new type of trapdoor and its associated algorithms are *strictly preferable* to a short basis in terms of algorithmic efficiency, output quality, and storage size (simultaneously).

Ring-based constructions. Finally, and most importantly for practice, all of the above-described constructions and algorithms extend immediately to the *ring* setting, where functions analogous to $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ require only quasi-linear $\tilde{O}(n)$ space and time to specify and evaluate (respectively), which is a factor of $\tilde{\Omega}(n)$ improvement over the matrix-based functions defined above. See the representative works [Mic02, PR06, LM06, LMPR08, LPR10] for more details on these functions and their security foundations.

	[Ajt99, AP09] constructions	This work (fast $f_{\mathbf{A}}^{-1}$)	Factor Improvement
Dimension m	slow $f_{\mathbf{A}}^{-1}$ [Kle00, GPV08]: $> 5n \lg q$ fast $f_{\mathbf{A}}^{-1}$ [Pei10]: $> n \lg^2 q$	$2n \lg q$ ($\overset{s}{\approx}$) $n(1 + \lg q)$ ($\overset{c}{\approx}$)	$2.5 - \lg q$
Quality s	slow $f_{\mathbf{A}}^{-1}$: $\approx 20\sqrt{n \lg q}$ fast $f_{\mathbf{A}}^{-1}$: $\approx 16\sqrt{n \lg^2 q}$	$\approx 1.6\sqrt{n \lg q}$ ($\overset{s}{\approx}$)	$12.5 - 10\sqrt{\lg q}$
Length $\beta \approx s\sqrt{m}$	slow $f_{\mathbf{A}}^{-1}$: $> 45n \lg q$ fast $f_{\mathbf{A}}^{-1}$: $> 16n \lg^2 q$	$\approx 2.3\, n \lg q$ ($\overset{s}{\approx}$)	$19 - 7 \lg q$

Figure 1: Summary of parameters for our constructions and algorithms versus prior ones. In the column labelled "this work," $\overset{s}{\approx}$ and $\overset{c}{\approx}$ denote constructions producing public keys $\mathbf{A}$ that are statistically close to uniform, and computationally pseudorandom, respectively. (All quality terms s and length bounds β omit the same statistical "smoothing" factor for $\mathbb{Z}$, which is about 4–5 in practice.)

To illustrate the kinds of concrete improvements that our methods provide, in Figure 2 we give representative parameters for the canonical application of GPV sigantures [GPV08], comparing the old and new trapdoor constructions for nearly equal levels of concrete security. We stress that these parameters are not highly optimized, and making adjustments to some of the tunable parameters in our constructions may provide better combinations of efficiency and concrete security. We leave this effort for future work.

1.2 Techniques

The main idea behind our new method of trapdoor generation is as follows. Instead of building a random matrix $\mathbf{A}$ through some specialized and complex process, we start from a carefully crafted *public* matrix $\mathbf{G}$ (and its associated lattice), for which the associated functions $f_{\mathbf{G}}$ and $g_{\mathbf{G}}$ admit very efficient (in both sequential and parallel complexity) and high-quality inversion algorithms. In particular, preimage sampling for $f_{\mathbf{G}}$ and inversion for $g_{\mathbf{G}}$ can be performed in essentially $O(n \log n)$ sequential time, and can even be performed by n parallel $O(\log n)$-time operations or table lookups. (This should be compared with the general algorithms for these tasks, which require at least quadratic $\Omega(n^2 \log^2 n)$ time, and are not always parallelizable for optimal noise parameters.) We emphasize that $\mathbf{G}$ is *not* a cryptographic key, but rather a fixed and public matrix that may be used by all parties, so the implementation of all its associated operations can be highly optimized, in both software and hardware. We also mention that the simplest and most practically efficient choices of $\mathbf{G}$ work for a modulus q that is a power of a small prime, such as $q = 2^k$, but a crucial search/decision reduction for LWE was not previously known for such q, despite its obvious practical utility. In Section 3 we provide a very general reduction that covers this case and others, and subsumes all of the known (and incomparable) search/decision reductions for LWE [BFKL93, Reg05, Pei09b, ACPS09].

To generate a *random* matrix $\mathbf{A}$ with a trapdoor, we take two additional steps: first, we extend $\mathbf{G}$ into a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{G}]$, for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ and sufficiently large $\bar{m}$. (As shown in [CHKP10], inversion of $g_{\mathbf{A}'}$ and preimage sampling for $f_{\mathbf{A}'}$ reduce very efficiently to the corresponding tasks for $g_{\mathbf{G}}$ and $f_{\mathbf{G}}$.) Finally, we simply apply to $\mathbf{A}'$ a certain random unimodular transformation defined by the matrix $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$, for a random "short" secret matrix $\mathbf{R}$ that will serve as the trapdoor, to obtain

$$\mathbf{A} = \mathbf{A}' \cdot \mathbf{T} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

	[AP09] with fast $f_{\mathbf{A}}^{-1}$	This work	Factor Improvement
Sec param n	436	284	1.5
Modulus q	2^{32}	2^{24}	256
Dimension m	446,644	13,812	32.3
Quality s	10.7×10^3	418	25.6
Length β	12.9×10^6	91.6×10^3	141
Key size (bits)	6.22×10^9	92.2×10^6	**67.5**
Key size (ring-based)	$\approx 16 \times 10^6$	$\approx 361 \times 10^3$	$\approx$ **44.3**

Figure 2: Representative parameters for GPV signatures (using fast inversion algorithms) for the old and new trapdoor generation methods. Using the methodology from [MR09], both sets of parameters have security level corresponding to a parameter δ of at most 1.007, which is estimated to require about 2^{46} core-years on a 64-bit 1.86GHz Xeon using the state-of-the-art in lattice basis reduction [GN08, CN11]. We use a smoothing parameter of $r = 4.5$ for $\mathbb{Z}$, which corresponds to statistical error of less than 2^{-90} for each randomized-rounding operation during signing. Key sizes are calculated using the Hermite normal form optimization. Key sizes for *ring-based* GPV signatures are approximated to be smaller by a factor of about $0.9n$.

The transformation given by $\mathbf{T}$ has the following properties:

- It is very easy to compute and invert, requiring essentially just one multiplication by $\mathbf{R}$ in both cases. (Note that $\mathbf{T}^{-1} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$.)
- It results in a matrix $\mathbf{A}$ that is distributed essentially uniformly at random, as required by the security reductions (and worst-case hardness proofs) for lattice-based cryptographic schemes.
- For the resulting functions $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$, preimage sampling and inversion very simply and efficiently reduce to the corresponding tasks for $f_{\mathbf{G}}$, $g_{\mathbf{G}}$. The overhead of the reduction is essentially just a single matrix-vector product with the secret matrix $\mathbf{R}$ (which, when inverting $f_{\mathbf{A}}$, can largely be precomputed even before the target value is known).

As a result, the cost of the inversion operations ends up being very close to that of computing $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ in the forward direction. Moreover, the fact that the running time is dominated by matrix-vector multiplications with the *fixed* trapdoor matrix $\mathbf{R}$ yields theoretical (but asymptotically significant) improvements in the context of batch execution of several operations relative to the same secret key $\mathbf{R}$: instead of evaluating several products $\mathbf{Rz}_1, \mathbf{Rz}_2, \ldots, \mathbf{Rz}_n$ individually at a total cost of $\Omega(n^3)$, one can employ fast matrix multiplication techniques to evaluate $\mathbf{R}[\mathbf{z}_1, \ldots, \mathbf{z}_n]$ as a whole is subcubic time. Batch operations can be exploited in applications like the multi-bit IBE of [GPV08] and its extensions to HIBE [CHKP10, ABB10a, ABB10b].

Related techniques. At the surface, our trapdoor generator appears similar to the original "GGH" approach of [GGH97] for generating a lattice together with a short basis. That technique works by choosing some random short vectors as the secret "good basis" of a lattice, and then transforms them into a public "bad basis" for the *same* lattice, via a unimodular matrix having large entries. (Note, though, that this does not produce a lattice from Ajtai's worst-case-hard family.) A closer look reveals, however, that (worst-case hardness aside) our method is actually *not* an instance of the GGH paradigm: here the initial short basis of the lattice

defined by $\mathbf{G}$ (or the semi-random matrix $[\bar{\mathbf{A}}|\mathbf{G}]$) is *fixed* and *public*, while the random unimodular matrix $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$ actually *produces a new lattice* by applying a (reversible) linear transformation to the original lattice. In other words, in contrast with GGH we multiply a (short) unimodular matrix on the "other side" of the original short basis, thus changing the lattice it generates.

A more appropriate comparison is to Ajtai's original method [Ajt96] for generating a random $\mathbf{A}$ together with a "weak" trapdoor of one or more short lattice vectors (but *not* a full basis). There, one simply chooses a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{0}]$ and outputs $\mathbf{A} = \mathbf{A}' \cdot \mathbf{T} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$, with short vectors $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$. Perhaps surprisingly, our strong trapdoor generator is just a simple twist on Ajtai's original weak generator, replacing $\mathbf{0}$ with the gadget $\mathbf{G}$.

Our constructions and inversion algorithms also draw upon several other techniques from throughout the literature. The trapdoor basis generator of [AP09] and the LWE-based "lossy" injective trapdoor function of [PW08] both use a fixed "gadget" matrix analogous to $\mathbf{G}$, whose entries grow geometrically in a structured way. In both cases, the gadget is concealed (either statistically or computationally) in the public key by a small combination of uniformly random vectors. Our method for adding tags to the trapdoor is very similar to a technique for doing the same with the lossy TDF of [PW08], and is identical to the method used in [ABB10a] for constructing compact (H)IBE. Finally, in our preimage sampling algorithm for $f_{\mathbf{A}}$, we use the "convolution" technique from [Pei10] to correct for some statistical skew that arises when converting preimages for $f_{\mathbf{G}}$ to preimages for $f_{\mathbf{A}}$, which would otherwise leak information about the trapdoor $\mathbf{R}$.

1.3 Applications

Our improved trapdoor generator and inversion algorithms can be plugged into any scheme that uses such tools as a "black box," and the resulting scheme will inherit all the efficiency improvements. (Every application we know of admits such a black-box replacement.) Moreover, the special properties of our methods allow for further improvements to the design, efficiency, and security reductions of existing schemes. Here we summarize some representative improvements that are possible to obtain; see Section 6 for complete details.

Hash-and-sign digital signatures. Our construction and supporting algorithms plug directly into the "full domain hash" signature scheme of [GPV08], which is strongly unforgeable in the random oracle model, with a tight security reduction. One can even use our computationally secure trapdoor generator to obtain a smaller public verification key, though at the cost of a hardness-of-LWE assumption, and a somewhat stronger SIS assumption (which affects concrete security). Determining the right balance between key size and security is left for later work.

In the standard model, there are two closely related types of hash-and-sign signature schemes:

- The one of [CHKP10], which has signatures of bit length $\tilde{O}(n^2)$, and is existentially unforgeable (later improved to be strongly unforgeable [Rüc10]) assuming the hardness of inverting $f_{\mathbf{A}}$ with solution length bounded by $\beta = \tilde{O}(n^{1.5})$.[2]
- The scheme of [Boy10], a lattice analogue of the pairing-based signature of [Wat05], which has signatures of bit length $\tilde{O}(n)$ and is existentially unforgeable assuming the hardness of inverting $f_{\mathbf{A}}$ with solution length bounded by $\beta = \tilde{O}(n^{3.5})$.

We improve the latter scheme in several ways, by: *(i)* improving the length bound to $\beta = \tilde{O}(n^{2.5})$; *(ii)* reducing the online runtime of the signing algorithm from $\tilde{O}(n^3)$ to $\tilde{O}(n^2)$ via chameleon hashing [KR00]; *(iii)* making the scheme strongly unforgeable *a la* [GPV08, Rüc10]; *(iv)* giving a tighter and simpler security reduction

[2] All parameters in this discussion assume a message length of $\tilde{\Theta}(n)$ bits.

(using a variant of the "prefix technique" [HW09] as in [CHKP10]), where the reduction's advantage degrades only linearly in the number of signature queries; and *(v)* removing all additional constraints on the parameters n and q (aside from those needed to ensure hardness of the SIS problem). We stress that the scheme itself is essentially the same (up to the improved and generalized parameters, and chameleon hashing) as that of [Boy10]; only the security proof and underlying assumption are improved. Note that in comparison with [CHKP10], there is still a trade-off between the bit length of the signatures and the bound β in the underlying SIS assumption; this appears to be inherent to the style of the security reduction. Note also that the public keys in all of these schemes are still rather large at $\tilde{O}(n^3)$ bits (or $\tilde{O}(n^2)$ bits using the ring analogue of SIS), so they are still mainly of theoretical interest. Improving the key sizes of standard-model signatures is an important open problem.

Chosen ciphertext-secure encryption. We give a new construction of CCA-secure public-key encryption (in the standard model) from the learning with errors (LWE) problem with error rate $\alpha = 1/\operatorname{poly}(n)$, where larger α corresponds to a harder concrete problem. Existing schemes exhibit various incomparable tradeoffs between key size and error rate. The first such scheme is due to [PW08]: it has public keys of size $\tilde{O}(n^2)$ bits (with somewhat large hidden factors) and relies on a quite small LWE error rate of $\alpha = \tilde{O}(1/n^4)$. The next scheme, from [Pei09b], has larger public keys of $\tilde{O}(n^3)$ bits, but uses a better error rate of $\alpha = \tilde{O}(1/n)$. Finally, using the generic conversion from selectively secure ID-based encryption to CCA-secure encryption [BCHK07], one can obtain from [ABB10a] a scheme having key size $\tilde{O}(n^2)$ bits and using error rate $\alpha = \tilde{O}(1/n^2)$. (Here decryption is randomized, since the IBE key-derivation algorithm is.) In particular, the public key of the scheme from [ABB10b] consists of 3 matrices in $\mathbb{Z}_q^{n \times m}$ where m is large enough to embed a (strong) trapdoor, plus essentially one vector in $\mathbb{Z}_q^n$ per message bit.

We give a CCA-secure system that enjoys the best of all prior constructions, which has $\tilde{O}(n^2)$-bit public keys, uses error rate $\alpha = \tilde{O}(1/n)$ (both with small hidden factors), and has deterministic decryption. To achieve this, we need to go beyond just plugging our improved trapdoor generator as a black box into prior constructions. Our scheme relies on the particular structure of the trapdoor instances; in effect, we directly construct a "tag-based adaptive trapdoor function" [KMO10]. The public key consists of only 1 matrix with an embedded (strong) trapdoor, rather than 3 as in the most compact scheme to date [ABB10a]; moreover, we can encrypt up to $n \log q$ message bits per ciphertext without needing any additional public key material. Combining these design changes with the improved dimension of our trapdoor generator, we obtain more than a 7.5-fold improvement in the public key size as compared with [ABB10a]. (This figure does not account for removing the extra public key material for the message bits, nor the other parameter improvements implied by our weaker concrete LWE assumption, which would shrink the keys even further.)

(Hierarchical) identity-based encryption. Just as with signatures, our constructions plug directly into the random-oracle IBE of [GPV08]. In the standard-model depth-d hierarchical IBEs of [CHKP10, ABB10a], our techniques can shrink the public parameters by an additional factor of about $\frac{2+4d}{1+d} \in [3, 4]$, relative to just plugging our improved trapdoor generator as a "black box" into the schemes. This is because for each level of the hierarchy, the public parameters only need to contain one matrix of the same dimension as $\mathbf{G}$ (i.e., about $n \lg q$), rather than two full trapdoor matrices (of dimension about $2n \lg q$ each).[3] Because the adaptation is straightforward given the tools developed in this work, we omit the details.

[3] We note that in [Pei09a] (an earlier version of [CHKP10]) the schemes are defined in a similar way using lower-dimensional extensions, rather than full trapdoor matrices at each level.

1.4 Other Related Work

Concrete parameter settings for a variety "strong" trapdoor applications are given in [RS10]. Those parameters are derived using the previous suboptimal generator of [AP09], and using the methods from this work would yield substantial improvements. The recent work of [LP11] also gives improved key sizes and concrete security for LWE-based cryptosystems; however, that work deals only with IND-CPA-secure encryption, and not at all with strong trapdoors or the further applications they enable (CCA security, digital signatures, (H)IBE, etc.).

2 Preliminaries

We denote the real numbers by $\mathbb{R}$ and the integers by $\mathbb{Z}$. For a nonnegative integer k, we let $[k] = \{1, \ldots, k\}$. Vectors are denoted by lower-case bold letters (e.g., $\mathbf{x}$) and are always in column form ($\mathbf{x}^t$ is a row vector). We denote matrices by upper-case bold letters, and treat a matrix $\mathbf{X}$ interchangeably with its ordered set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots\}$ of column vectors. For convenience, we sometimes use a scalar s to refer to the scaled identity matrix $s\mathbf{I}$, where the dimension will be clear from context.

The statistical distance between two distributions X, Y over a finite or countable domain D is $\Delta(X, Y) = \frac{1}{2}\sum_{w \in D} |X(w) - Y(w)|$. Statistical distance is a metric, and in particular obeys the triangle inequality. We say that a distribution over D is ϵ-uniform if its statistical distance from the uniform distribution is at most ϵ.

Throughout the paper, we use a "randomized-rounding parameter" r that we let be a fixed function $r(n) = \omega(\sqrt{\log n})$ growing asymptotically faster than $\sqrt{\log n}$. By "fixed function" we mean that $r = \omega(\sqrt{\log n})$ always refers to the very same function, and no other factors will be absorbed into the $\omega(\cdot)$ notation. This allows us to keep track of the precise multiplicative constants introduced by our constructions. Concretely, we take $r \approx \sqrt{\ln(2/\epsilon)/\pi}$ where ϵ is a desired bound on the statistical error introduced by each randomized-rounding operation for $\mathbb{Z}$, because the error is bounded by $\approx 2\exp(-\pi r^2)$ according to Lemma 2.3 below. For example, for $\epsilon = 2^{-54}$ we have $r \leq 3.5$, and for $\epsilon = 2^{-71}$ we have $r \leq 4$.

2.1 Linear Algebra

A unimodular matrix $\mathbf{U} \in \mathbb{Z}^{m \times m}$ is one for which $|\det(\mathbf{U})| = 1$; in particular, $\mathbf{U}^{-1} \in \mathbb{Z}^{m \times m}$ as well. The *Gram-Schmidt orthogonalization* of an ordered set of vectors $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \in \mathbb{R}^n$, is $\widetilde{\mathbf{V}} = \{\widetilde{\mathbf{v}}_1, \ldots, \widetilde{\mathbf{v}}_k\}$ where $\widetilde{\mathbf{v}}_i$ is the component of $\mathbf{v}_i$ orthogonal to $\mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{i-1})$ for all $i = 1, \ldots, k$. (In some cases we orthogonalize the vectors in a different order.) In matrix form, $\mathbf{V} = \mathbf{QDU}$ for some orthogonal $\mathbf{Q} \in \mathbb{R}^{n \times k}$, diagonal $\mathbf{D} \in \mathbb{R}^{k \times k}$ with nonnegative entries, and upper unitriangular $\mathbf{U} \in \mathbb{R}^{k \times k}$ (i.e., $\mathbf{U}$ is upper triangular with 1s on the diagonal). The decomposition is unique when the $\mathbf{v}_i$ are linearly independent, and we always have $\|\widetilde{\mathbf{v}}_i\| = d_{i,i}$, the ith diagonal entry of $\mathbf{D}$.

For any basis $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of $\mathbb{R}^n$, its origin-centered parallelepiped is defined as $\mathcal{P}_{1/2}(\mathbf{V}) = \mathbf{V} \cdot [-\frac{1}{2}, \frac{1}{2})^n$. Its dual basis is defined as $\mathbf{V}^* = \mathbf{V}^{-t} = (\mathbf{V}^{-1})^t$. If we orthogonalize $\mathbf{V}$ and $\mathbf{V}^*$ in forward and reverse order, respectively, then we have $\widetilde{\mathbf{v}_i^*} = \widetilde{\mathbf{v}}_i / \|\widetilde{\mathbf{v}}_i\|^2$ for all i. In particular, $\|\widetilde{\mathbf{v}_i^*}\| = 1/\|\widetilde{\mathbf{v}}_i\|$.

For any square real matrix $\mathbf{X}$, the *(Moore-Penrose) pseudoinverse*, denoted $\mathbf{X}^+$, is the unique matrix satisfying $(\mathbf{X}\mathbf{X}^+)\mathbf{X} = \mathbf{X}$, $\mathbf{X}^+(\mathbf{X}\mathbf{X}^+) = \mathbf{X}^+$, and such that both $\mathbf{X}\mathbf{X}^+$ and $\mathbf{X}^+\mathbf{X}$ are symmetric. We always have $\mathrm{span}(\mathbf{X}) = \mathrm{span}(\mathbf{X}^+)$, and when $\mathbf{X}$ is invertible, we have $\mathbf{X}^+ = \mathbf{X}^{-1}$.

A symmetric matrix $\Sigma \in \mathbb{R}^{n \times n}$ is *positive definite* (respectively, positive *semi*definite), written $\Sigma > \mathbf{0}$ (resp., $\Sigma \geq \mathbf{0}$), if $\mathbf{x}^t \Sigma \mathbf{x} > 0$ (resp., $\mathbf{x}^t \Sigma \mathbf{x} \geq 0$) for all nonzero $\mathbf{x} \in \mathbb{R}^n$. We have $\Sigma > \mathbf{0}$ if and only if Σ is invertible and $\Sigma^{-1} > \mathbf{0}$, and $\Sigma \geq \mathbf{0}$ if and only if $\Sigma^+ \geq \mathbf{0}$. Positive (semi)definiteness defines a partial

ordering on symmetric matrices: we say that $\Sigma_1 > \Sigma_2$ if $(\Sigma_1 - \Sigma_2) > \mathbf{0}$, and similarly for $\Sigma_1 \geq \Sigma_2$. We have $\Sigma_1 \geq \Sigma_2 \geq \mathbf{0}$ if and only if $\Sigma_2^+ \geq \Sigma_1^+ \geq \mathbf{0}$, and likewise for the analogous strict inequalities.

For any matrix $\mathbf{B}$, the symmetric matrix $\Sigma = \mathbf{B}\mathbf{B}^t$ is positive semidefinite, because

$$\mathbf{x}^t \Sigma \mathbf{x} = \langle \mathbf{B}^t\mathbf{x}, \mathbf{B}^t\mathbf{x} \rangle = \|\mathbf{B}^t\mathbf{x}\|^2 \geq 0$$

for any nonzero $\mathbf{x} \in \mathbb{R}^n$, where the inequality is always strict if and only if $\mathbf{B}$ is nonsingular. We say that $\mathbf{B}$ is a *square root* of $\Sigma > \mathbf{0}$, written $\mathbf{B} = \sqrt{\Sigma}$, if $\mathbf{B}\mathbf{B}^t = \Sigma$. Every $\Sigma \geq \mathbf{0}$ has a square root, which can be computed efficiently, e.g., via the Cholesky decomposition.

For any matrix $\mathbf{B} \in \mathbb{R}^{n \times k}$, there exists a *singular value decomposition* $\mathbf{B} = \mathbf{Q}\mathbf{D}\mathbf{P}^t$, where $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{P} \in \mathbb{R}^{k \times k}$ are orthogonal matrices, and $\mathbf{D} \in \mathbb{R}^{n \times k}$ is a diagonal matrix with nonnegative entries $s_i \geq 0$ on the diagonal, in non-increasing order. The s_i are called the *singular values* of $\mathbf{B}$. Under this convention, $\mathbf{D}$ is uniquely determined (though $\mathbf{Q}, \mathbf{P}$ may not be), and $s_1(\mathbf{B}) = \max_{\mathbf{u}} \|\mathbf{B}\mathbf{u}\| = \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\| \geq \|\mathbf{B}\|, \|\mathbf{B}^t\|$, where the maxima are taken over all unit vectors $\mathbf{u} \in \mathbb{R}^k$.

2.2 Lattices and Hard Problems

Generally defined, an m-dimensional *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^m$. For some $k \leq m$, called the *rank* of the lattice, Λ is generated as the set of all $\mathbb{Z}$-linear combinations of some k linearly independent *basis* vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$, i.e., $\Lambda = \{\mathbf{B}\mathbf{z} : \mathbf{z} \in \mathbb{Z}^k\}$. In this work, we are mostly concerned with full-rank integer lattices, i.e., $\Lambda \subseteq \mathbb{Z}^m$ with $k = m$. (We work with non-full-rank lattices only in the analysis of our Gaussian sampling algorithm in Section 5.4.) The dual lattice Λ^* is the set of all $\mathbf{v} \in \mathrm{span}(\Lambda)$ such that $\langle \mathbf{v}, \mathbf{x} \rangle \in \mathbb{Z}$ for every $\mathbf{x} \in \Lambda$. If $\mathbf{B}$ is a basis of Λ, then $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^t\mathbf{B})^{-1}$ is a basis of Λ^*. Note that when Λ is full-rank, $\mathbf{B}$ is invertible and hence $\mathbf{B}^* = \mathbf{B}^{-t}$.

Many cryptographic applications use a particular family of so-called q*-ary* integer lattices, which contain $q\mathbb{Z}^m$ as a sublattice for some (typically small) integer q. For positive integers n and q, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary and define the following full-rank m-dimensional q-ary lattices:

$$\begin{aligned} \Lambda^\perp(\mathbf{A}) &= \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \bmod q\} \\ \Lambda(\mathbf{A}^t) &= \{\mathbf{z} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{z} = \mathbf{A}^t\mathbf{s} \bmod q\}. \end{aligned}$$

It is easy to check that $\Lambda^\perp(\mathbf{A})$ and $\Lambda(\mathbf{A}^t)$ are dual lattices, up to a q scaling factor: $q \cdot \Lambda^\perp(\mathbf{A})^* = \Lambda(\mathbf{A}^t)$, and vice-versa. For this reason, it is sometimes more natural to consider the non-integral, "1-ary" lattice $\frac{1}{q}\Lambda(\mathbf{A}^t) = \Lambda^\perp(\mathbf{A})^* \supseteq \mathbb{Z}^m$. For any $\mathbf{u} \in \mathbb{Z}_q^n$ admitting an integral solution to $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q$, define the coset (or "shifted" lattice)

$$\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \bmod q\} = \Lambda^\perp(\mathbf{A}) + \mathbf{x}.$$

Here we recall some basic facts about these q-ary lattices.

Lemma 2.1. *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary and let $\mathbf{S} \in \mathbb{Z}^{m \times m}$ be any basis of $\Lambda^\perp(\mathbf{A})$.*

1. *For any unimodular $\mathbf{T} \in \mathbb{Z}^{m \times m}$, we have $\mathbf{T} \cdot \Lambda^\perp(\mathbf{A}) = \Lambda^\perp(\mathbf{A} \cdot \mathbf{T}^{-1})$, with $\mathbf{T} \cdot \mathbf{S}$ as a basis.*
2. *[ABB10a, implicit] For any invertible $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, we have $\Lambda^\perp(\mathbf{H} \cdot \mathbf{A}) = \Lambda^\perp(\mathbf{A})$.*
3. *[CHKP10, Lemma 3.2] Suppose that the columns of $\mathbf{A}$ generate all of $\mathbb{Z}_q^n$, let $\mathbf{A}' \in \mathbb{Z}_q^{n \times m'}$ be arbitrary, and let $\mathbf{W} \in \mathbb{Z}^{m \times m'}$ be an arbitrary solution to $\mathbf{A}\mathbf{W} = -\mathbf{A}' \bmod q$. Then $\mathbf{S}' = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{smallmatrix}\right]$ is a basis of $\Lambda^\perp([\mathbf{A}' \mid \mathbf{A}])$, and when orthogonalized in appropriate order, $\widetilde{\mathbf{S}'} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{S}} \end{smallmatrix}\right]$. In particular, $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$.*

Cryptographic problems. For $\beta > 0$, the *short integer solution* problem $\mathsf{SIS}_{q,\beta}$ is an average-case version of the approximate shortest vector problem on $\Lambda^{\perp}(\mathbf{A})$. The problem is: given uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for any desired $m = \mathrm{poly}(n)$, find a relatively short nonzero $\mathbf{z} \in \Lambda^{\perp}(\mathbf{A})$, i.e., output a nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$. When $q \geq \beta\sqrt{n} \cdot \omega(\sqrt{\log n})$, solving this problem (with any non-negligible probability over the random choice of $\mathbf{A}$) is at least as hard as (probabilistically) approximating the Shortest Independent Vectors Problem (SIVP, a classic problem in the computational study of point lattices [MG02]) on n-dimensional lattices to within $\tilde{O}(\beta\sqrt{n})$ factors in the *worst case* [Ajt96, MR04, GPV08].

For $\alpha > 0$, the *learning with errors* problem $\mathsf{LWE}_{q,\alpha}$ may be seen an average-case version of the bounded-distance decoding problem on the dual lattice $\frac{1}{q}\Lambda(\mathbf{A}^t)$. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, the additive group of reals modulo 1, and let D_α denote the Gaussian probability distribution over $\mathbb{R}$ with parameter α (see Section 2.3 below). For any fixed $\mathbf{s} \in \mathbb{Z}_q^n$, define $A_{\mathbf{s},\alpha}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ obtained by choosing $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow D_\alpha$, and outputting $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle / q + e \bmod 1)$. The search-$\mathsf{LWE}_{q,\alpha}$ problem is: given any desired number $m = \mathrm{poly}(n)$ of independent samples from $A_{\mathbf{s},\alpha}$ for some arbitrary $\mathbf{s}$, find $\mathbf{s}$. The decision-$\mathsf{LWE}_{q,\alpha}$ problem is to distinguish, with non-negligible advantage, between samples from $A_{\mathbf{s},\alpha}$ for uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$, and uniformly random samples from $\mathbb{Z}_q^n \times \mathbb{T}$. There are a variety of (incomparable) search/decision reductions for LWE under certain conditions on the parameters (e.g., [Reg05, Pei09b, ACPS09]); in Section 3 we give a reduction that essentially subsumes them all. When $q \geq 2\sqrt{n}/\alpha$, solving search-$\mathsf{LWE}_{q,\alpha}$ is at least as hard as *quantumly* approximating SIVP on n-dimensional lattices to within $\tilde{O}(n/\alpha)$ factors in the worst case [Reg05]. For a restricted range of parameters (e.g., when q is exponentially large) a *classical* (non-quantum) reduction is also known [Pei09b], but only from a potentially easier class of problems like the decisional Shortest Vector Problem (GapSVP) and the Bounded Distance Decoding Problem (BDD) (see [LM09]).

Note that the m samples $(\mathbf{a}_i, b_i)$ and underlying error terms e_i from $A_{\mathbf{s},\alpha}$ may be grouped into a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vectors $\mathbf{b} \in \mathbb{T}^m$, $\mathbf{e} \in \mathbb{R}^m$ in the natural way, so that $\mathbf{b} = (\mathbf{A}^t\mathbf{s})/q + \mathbf{e} \bmod 1$. In this way, $\mathbf{b}$ may be seen as an element of $\Lambda^{\perp}(\mathbf{A})^* = \frac{1}{q}\Lambda(\mathbf{A}^t)$, perturbed by Gaussian error. By scaling $\mathbf{b}$ and discretizing its entries using a form of randomized rounding (see [Pei10]), we can convert it into $\mathbf{b}' = \mathbf{A}^t\mathbf{s} + \mathbf{e}' \bmod q$ where $\mathbf{e}' \in \mathbb{Z}^m$ has *discrete* Gaussian distribution with parameter (say) $\sqrt{2}\alpha q$.

2.3 Gaussians and Lattices

The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle).$$

Applying a linear transformation given by a (not necessarily square) matrix $\mathbf{B}$ with linearly independent columns yields the (possibly degenerate) Gaussian function

$$\rho_{\mathbf{B}}(\mathbf{x}) \stackrel{\Delta}{=} \begin{cases} \rho(\mathbf{B}^+\mathbf{x}) = \exp\left(-\pi \cdot \mathbf{x}^t\Sigma^+\mathbf{x}\right) & \text{if } \mathbf{x} \in \mathrm{span}(\mathbf{B}) = \mathrm{span}(\Sigma) \\ 0 & \text{otherwise} \end{cases}$$

where $\Sigma = \mathbf{B}\mathbf{B}^t \geq \mathbf{0}$. Because $\rho_{\mathbf{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

Normalizing $\rho_{\sqrt{\Sigma}}$ by its total measure over $\mathrm{span}(\Sigma)$, we obtain the probability distribution function of the (continuous) *Gaussian distribution* $D_{\sqrt{\Sigma}}$. By linearity of expectation, this distribution has *covariance* $\mathbb{E}_{\mathbf{x} \leftarrow D_{\sqrt{\Sigma}}}[\mathbf{x} \cdot \mathbf{x}^t] = \frac{\Sigma}{2\pi}$. (The $\frac{1}{2\pi}$ factor is the variance of the Gaussian D_1, due to our choice of normalization.) For convenience, we implicitly ignore the $\frac{1}{2\pi}$ factor, and refer to Σ as the covariance matrix of $D_{\sqrt{\Sigma}}$.

Let $\Lambda \subset \mathbb{R}^n$ be a lattice, let $\mathbf{c} \in \mathbb{R}^n$, and let $\Sigma \geq \mathbf{0}$ be a positive semidefinite matrix such that $(\Lambda + \mathbf{c}) \cap \mathrm{span}(\Sigma)$ is nonempty. The *discrete Gaussian distribution* $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$ is simply the Gaussian distribution $D_{\sqrt{\Sigma}}$ restricted to have support $\Lambda + \mathbf{c}$. That is, for all $\mathbf{x} \in \Lambda + \mathbf{c}$,

$$D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x})}{\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c})} \propto \rho_{\sqrt{\Sigma}}(\mathbf{x}).$$

We recall the definition of the *smoothing parameter* from [MR04], generalized to non-spherical (and potentially degenerate) Gaussians. It is easy to see that the definition is consistent with the partial ordering of positive semidefinite matrices, i.e., if $\Sigma_1 \geq \Sigma_2 \geq \eta_\epsilon(\Lambda)$, then $\Sigma_1 \geq \eta_\epsilon(\Lambda)$.

Definition 2.2. Let $\Sigma \geq \mathbf{0}$ and $\Lambda \subset \mathrm{span}(\Sigma)$ be a lattice. We say that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ if $\rho_{\sqrt{\Sigma^+}}(\Lambda^*) \leq 1 + \epsilon$.

The following is a bound on the smoothing parameter in terms of any orthogonalized basis. Note that for practical choices like $n \leq 2^{14}$ and $\epsilon \geq 2^{-80}$, the multiplicative factor attached to $\|\widetilde{\mathbf{B}}\|$ is bounded by 4.6.

Lemma 2.3 ([GPV08, Theorem 3.1])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. We have*

$$\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n(1 + 1/\epsilon))/\pi}.$$

In particular, for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$.

For appropriate parameters, the smoothing parameter of a random lattice $\Lambda^\perp(\mathbf{A})$ is small, with very high probability. The following bound is a refinement and strengthening of one from [GPV08], which allows for a more precise analysis of the parameters and statistical errors involved in our constructions.

Lemma 2.4. *Let n, m, $q \geq 2$ be positive integers. For $\mathbf{s} \in \mathbb{Z}_q^n$, let the subgroup $\mathbb{G}_\mathbf{s} = \{\langle \mathbf{a}, \mathbf{s} \rangle : \mathbf{a} \in \mathbb{Z}_q^n\} \subseteq \mathbb{Z}_q$, and let $g_\mathbf{s} = |\mathbb{G}_\mathbf{s}| = q/\gcd(s_1, \ldots, s_n, q)$. Let $\epsilon > 0$, $\eta \geq \eta_\epsilon(\mathbb{Z}^m)$, and $s > \eta$ be reals. Then for uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$,*

$$\mathop{\mathbb{E}}_{\mathbf{A}} \left[\rho_{1/s}(\Lambda^\perp(\mathbf{A})^*) \right] \leq (1 + \epsilon) \sum_{\mathbf{s} \in \mathbb{Z}_q^n} \max\{1/g_\mathbf{s}, \eta/s\}^m. \tag{2.1}$$

In particular, if $q = p^e$ is a power of a prime p, and

$$m \geq \max\left\{ n + \frac{\log(3 + 2/\epsilon)}{\log p}, \frac{n \log q + \log(2 + 2/\epsilon)}{\log(s/\eta)} \right\}, \tag{2.2}$$

then $\mathbb{E}_\mathbf{A} \left[\rho_{1/s}(\Lambda^\perp(\mathbf{A})^*) \right] \leq 1 + 2\epsilon$, and so by Markov's inequality, $s \geq \eta_{2\epsilon/\delta}(\Lambda^\perp(\mathbf{A}))$ except with probability at most δ.

Proof. We will use the fact (which follows from the Poisson summation formula; see [MR04, Lemma 2.8]) that $\rho_t(\Lambda) \leq \rho_r(\Lambda) \leq (r/t)^m \cdot \rho_t(\Lambda)$ for any rank-m lattice Λ and $r \geq t > 0$.

For any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, one can check that $\Lambda^\perp(\mathbf{A})^* = \mathbb{Z}^m + \{\mathbf{A}^t\mathbf{s}/q : \mathbf{s} \in \mathbb{Z}_q^n\}$. Note that $\mathbf{A}^t\mathbf{s}$ is uniformly

random over $\mathbb{G}_{\mathbf{s}}^m$, for uniformly random $\mathbf{A}$. Then we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{A}}\left[\rho_{1/s}(\Lambda^{\perp}(\mathbf{A})^*)\right] &\le \sum_{\mathbf{s}\in\mathbb{Z}_q^n} \mathop{\mathbb{E}}_{\mathbf{A}}\left[\rho_{1/s}(\mathbb{Z}^m + \mathbf{A}^t\mathbf{s}/q)\right] && \text{(lin. of } \mathbb{E}\text{)} \\
&= \sum_{\mathbf{s}\in\mathbb{Z}_q^n} g_{\mathbf{s}}^{-m} \cdot \rho_{1/s}(g_{\mathbf{s}}^{-1}\cdot\mathbb{Z}^m) && \text{(avg. over } \mathbf{A}\text{)} \\
&\le \sum_{\mathbf{s}\in\mathbb{Z}_q^n} g_{\mathbf{s}}^{-m} \cdot \max\{1, g_{\mathbf{s}}\eta/s\}^m \cdot \rho_{1/\eta}(\mathbb{Z}^m), && \text{(above fact)} \\
&\le (1+\epsilon)\sum_{\mathbf{s}\in\mathbb{Z}_q^n} \max\{1/g_{\mathbf{s}}, \eta/s\}^m, && (\eta \ge \eta_\epsilon(\mathbb{Z}^m)).
\end{aligned}$$

To prove the second part of the claim, observe that $g_{\mathbf{s}} = p^i$ for some $i \ge 0$, and that there are at most g^n values of $\mathbf{s}$ for which $g_{\mathbf{s}} = g$, because each entry of $\mathbf{s}$ must be in $\mathbb{G}_{\mathbf{s}}$. Therefore,

$$\sum_{\mathbf{s}\in\mathbb{Z}_q^n} 1/g_{\mathbf{s}}^m \le \sum_{i\ge 0} p^{i(n-m)} = \frac{1}{1-p^{n-m}} \le 1 + \frac{\epsilon}{2(1+\epsilon)}.$$

(More generally, for arbitrary q we have $\sum_{\mathbf{s}} 1/g_{\mathbf{s}}^m \le \zeta(m-n)$, where $\zeta(\cdot)$ is the Riemann zeta function.) Similarly, $\sum_{\mathbf{s}}(\eta/s)^m = q^n(s/\eta)^{-m} \le \frac{\epsilon}{2(1+\epsilon)}$, and the claim follows. □

We need a number of standard facts about discrete Gaussians.

Lemma 2.5 ([MR04, Lemmas 2.9 and 4.1])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice. For any $\Sigma \ge \mathbf{0}$ and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \le \rho_{\sqrt{\Sigma}}(\Lambda)$. Moreover, if $\sqrt{\Sigma} \ge \eta_\epsilon(\Lambda)$ for some $\epsilon > 0$ and $\mathbf{c} \in \mathrm{span}(\Lambda)$, then $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \ge \frac{1-\epsilon}{1+\epsilon}\cdot\rho_{\sqrt{\Sigma}}(\Lambda)$.*

Combining the above lemma with a bound of Banaszczyk [Ban93], we have the following tail bound on discrete Gaussians.

Lemma 2.6 ([Ban93, Lemma 1.5])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $r \ge \eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$. For any $\mathbf{c} \in \mathrm{span}(\Lambda)$, we have*

$$\Pr\left[\|D_{\Lambda+\mathbf{c},r}\| \ge r\sqrt{n}\right] \le 2^{-n}\cdot\tfrac{1+\epsilon}{1-\epsilon}.$$

Moreover, if $\mathbf{c} = \mathbf{0}$ then the bound holds for any *$r > 0$, with $\epsilon = 0$.*

The next lemma bounds the predictability (i.e., probability of the most likely outcome or equivalently, min-entropy) of a discrete Gaussian.

Lemma 2.7 ([PR06, Lemma 2.11])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $r \ge 2\eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$. For any $\mathbf{c} \in \mathbb{R}^n$ and any $\mathbf{y} \in \Lambda + \mathbf{c}$, we have $\Pr[D_{\Lambda+\mathbf{c},r} = \mathbf{y}] \le 2^{-n}\cdot\frac{1+\epsilon}{1-\epsilon}$.*

2.4 Subgaussian Distributions and Random Matrices

For $\delta \ge 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is *δ-subgaussian* with parameter $s > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies

$$\mathbb{E}\left[\exp(2\pi t X)\right] \le \exp(\delta)\cdot\exp(\pi s^2 t^2).$$

Notice that the $\exp(\pi s^2 t^2)$ term on the right is precisely the (scaled) moment-generating function of the Gaussian distribution D_s. So, our definition differs from the usual definition of subgaussian only in the additional factor of $\exp(\delta)$; we need this relaxation when working with discrete Gaussians, usually taking $\delta = \ln(\frac{1+\epsilon}{1-\epsilon}) \approx 2\epsilon$ for the same small ϵ as in the smoothing parameter η_ϵ.

If X is δ-subgaussian, then its tails are dominated by a Gaussian of parameter s, i.e., $\Pr[|X| \geq t] \leq 2\exp(\delta)\exp(-\pi t^2/s^2)$ for all $t \geq 0$.[4] This follows by Markov's inequality: by scaling X we can assume $s = 1$, and we have

$$\Pr[X \geq t] = \Pr[\exp(2\pi t X) \geq \exp(2\pi t^2)] \leq \exp(\delta)\exp(\pi t^2)/\exp(2\pi t^2) = \exp(\delta)\exp(-\pi t^2).$$

The claim follows by repeating the argument with $-X$, and the union bound. Using the Taylor series expansion of $\exp(2\pi t X)$, it can be shown that any B-bounded symmetric random variable X (i.e., $|X| \leq B$ always) is 0-subgaussian with parameter $B\sqrt{2\pi}$.

More generally, we say that a random vector $\mathbf{x}$ or its distribution (respectively, a random matrix $\mathbf{X}$) is δ-subgaussian (of parameter s) if all its one-dimensional marginals $\langle \mathbf{u}, \mathbf{v} \rangle$ (respectively, $\mathbf{u}^t \mathbf{X} \mathbf{v}$) for unit vectors $\mathbf{u}, \mathbf{v}$ are δ-subgaussian (of parameter s). It follows immediately from the definition that the concatenation of independent δ_i-subgaussian vectors with common parameter s, interpreted as either a vector or matrix, is $(\sum \delta_i)$-subgaussian with parameter s.

Lemma 2.8. *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $s \geq \eta_\epsilon(\Lambda)$ for some $0 < \epsilon < 1$. For any $\mathbf{c} \in \mathrm{span}(\Lambda)$, $D_{\Lambda+\mathbf{c},s}$ is $\ln(\frac{1+\epsilon}{1-\epsilon})$-subgaussian with parameter s. Moreover, it is 0-subgaussian for* any *$s > 0$ when $\mathbf{c} = \mathbf{0}$.*

Proof. By scaling Λ we can assume that $s = 1$. Let $\mathbf{x}$ have distribution $D_{\Lambda+\mathbf{c}}$, and let $\mathbf{u} \in \mathbb{R}^n$ be any unit vector. We bound the scaled moment-generating function of the marginal $\langle \mathbf{x}, \mathbf{u} \rangle$ for any $t \in \mathbb{R}$:

$$\begin{aligned}
\rho(\Lambda + \mathbf{c}) \cdot \mathbb{E}\left[\exp(2\pi\langle \mathbf{x}, t\mathbf{u}\rangle)\right] &= \sum_{\mathbf{x} \in \Lambda + \mathbf{c}} \exp(-\pi(\langle \mathbf{x}, \mathbf{x}\rangle - 2\langle \mathbf{x}, t\mathbf{u}\rangle)) \\
&= \exp(\pi t^2) \cdot \sum_{\mathbf{x} \in \Lambda + \mathbf{c}} \exp(-\pi\langle \mathbf{x} - t\mathbf{u}, \mathbf{x} - t\mathbf{u}\rangle) \\
&= \exp(\pi t^2) \cdot \rho(\Lambda + \mathbf{c} - t\mathbf{u}).
\end{aligned}$$

Both claims then follow by Lemma 2.5. □

Here we recall a standard result from the non-asymptotic theory of random matrices; for further details, see [Ver11]. (The proof for δ-subgaussian distributions is a trivial adaptation of the 0-subgaussian case.)

Lemma 2.9. *Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a δ-subgaussian random matrix with parameter s. There exists a universal constant $C > 0$ such that for any $t \geq 0$, we have $s_1(\mathbf{X}) \leq C \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2\exp(\delta)\exp(-\pi t^2)$.*

Empirically, for discrete Gaussians the universal constant C in the above lemma is very close to $1/\sqrt{2\pi}$. In fact, it has been proved that $C \leq 1/\sqrt{2\pi}$ for matrices with independent identically distributed *continuous* Gaussian entries.

[4] The converse also holds (up to a small constant factor in the parameter s) when $\mathbb{E}[X] = 0$, but this will frequently not quite be the case in our applications, which is why we define subgaussian in terms of the moment-generating function.

3 Search to Decision Reduction

Here we give a new search-to-decision reduction for LWE that essentially subsumes all of the (incomparable) prior ones given in [BFKL93, Reg05, Pei09b, ACPS09].[5] Most notably, it handles moduli q that were not covered before, specifically, those like $q = 2^k$ that are divisible by powers of very small primes. The only known reduction that ours does not subsume is a different style of *sample-preserving* reduction recently given in [MM11], which works for a more limited class of moduli and error distributions; extending that reduction to the full range of parameters considered here is an interesting open problem. In what follows, $\omega(\sqrt{\log n})$ denotes some fixed function that grows faster than $\sqrt{\log n}$, asymptotically.

Theorem 3.1. *Let q have prime factorization $q = p_1^{e_1} \cdots p_k^{e_k}$ for pairwise distinct $\mathrm{poly}(n)$-bounded primes p_i with each $e_i \geq 1$, and let $0 < \alpha \leq 1/\omega(\sqrt{\log n})$. Let ℓ be the number of prime factors $p_i < \omega(\sqrt{\log n})/\alpha$. There is a probabilistic polynomial-time reduction from solving search-$\mathsf{LWE}_{q,\alpha}$ (in the worst case, with overwhelming probability) to solving decision-$\mathsf{LWE}_{q,\alpha'}$ (on the average, with non-negligible advantage) for any $\alpha' \geq \alpha$ such that $\alpha' \geq \omega(\sqrt{\log n})/p_i^{e_i}$ for every i, and $(\alpha')^\ell \geq \alpha \cdot \omega(\sqrt{\log n})^{1+\ell}$.*

For example, when every $p_i \geq \omega(\sqrt{\log n})/\alpha$ we have $\ell = 0$, and any $\alpha' \geq \alpha$ is acceptable. (This special case, with the additional constraint that every $e_i = 1$, is proved in [Pei09b].) As a qualitatively new example, when $q = p^e$ is a prime power for some (possibly small) prime p, then it suffices to let $\alpha' \geq \alpha \cdot \omega(\sqrt{\log n})^2$. (A similar special case where $q = p^e$ for sufficiently large p and $\alpha' = \alpha \ll 1/p$ is proved in [ACPS09].)

Proof. We show how to recover each entry of $\mathbf{s}$ modulo a large enough power of each p_i, given access to the distribution $A_{\mathbf{s},\alpha}$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and to an oracle $\mathcal{O}$ solving $\mathsf{DLWE}_{q,\alpha'}$. For the parameters in the theorem statement, we can then recover the remainder of $\mathbf{s}$ in polynomial time by rounding and standard Gaussian elimination.

First, observe that we can transform $A_{\mathbf{s},\alpha}$ into $A_{\mathbf{s},\alpha'}$ simply by adding (modulo 1) an independent sample from $D_{\sqrt{\alpha'^2 - \alpha^2}}$ to the second component of each $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle / q + D_\alpha \bmod 1) \in \mathbb{Z}_q^n \times \mathbb{T}$ drawn from $A_{\mathbf{s},\alpha}$.

We now show how to recover each entry of $\mathbf{s}$ modulo (powers of) any prime $p = p_i$ dividing q. Let $e = e_i$, and for $j = 0, 1, \ldots, e$ define $A^j_{\mathbf{s},\alpha'}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ obtained by drawing $(\mathbf{a}, b) \leftarrow A_{\mathbf{s},\alpha'}$ and outputting $(\mathbf{a}, b + r/p^j \bmod 1)$ for a fresh uniformly random $r \leftarrow \mathbb{Z}_q$. (Clearly, this distribution can be generated efficiently from $A_{\mathbf{s},\alpha'}$.) Note that when $\alpha' \geq \omega(\sqrt{\log n})/p^j \geq \eta_\epsilon((1/p^j)\mathbb{Z})$ for some $\epsilon = \mathrm{negl}(n)$, $A^j_{\mathbf{s},\alpha'}$ is negligibly far from $U = U(\mathbb{Z}_q^n \times \mathbb{T})$, and this holds at least for $j = e$ by hypothesis. Therefore, by a hybrid argument there exists some minimal $j \in [e]$ for which $\mathcal{O}$ has a non-negligible advantage in distinguishing between $A^{j-1}_{\mathbf{s},\alpha'}$ and $A^j_{\mathbf{s},\alpha'}$, over a random choice of $\mathbf{s}$ and all other randomness in the experiment. (This j can be found efficiently by measuring the behavior of $\mathcal{O}$.) Note that when $p_i \geq \omega(\sqrt{\log n})/\alpha \geq \omega(\sqrt{\log n})/\alpha'$, the minimal j must be 1; otherwise it may be larger, but there are at most ℓ of these by hypothesis. Now by a standard random self-reduction and amplification techniques (e.g., [Reg05, Lemma 4.1]), we can in fact assume that $\mathcal{O}$ accepts (respectively, rejects) with *overwhelming* probability given $A^{j-1}_{\mathbf{s},\alpha'}$ (resp., $A^j_{\mathbf{s},\alpha'}$), for *any* $\mathbf{s} \in \mathbb{Z}_q^n$.

Given access to $A^{j-1}_{\mathbf{s},\alpha'}$ and $\mathcal{O}$, we can test whether $s_1 = 0 \bmod p$ by invoking $\mathcal{O}$ on samples from $A^{j-1}_{\mathbf{s},\alpha'}$ that have been transformed as follows (all of what follows is analogous for $s_2, \ldots, s_n$): take each sample $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle / q + e + r/p^{j-1} \bmod 1) \leftarrow A^{j-1}_{\mathbf{s},\alpha'}$ to

$$(\mathbf{a}' = \mathbf{a} - r' \cdot (q/p^j) \cdot \mathbf{e}_1 \quad , \quad b' = b = \langle \mathbf{a}', \mathbf{s} \rangle / q + e + (pr + r's_1)/p^j \bmod 1) \tag{3.1}$$

[5] We say "essentially subsumes" because our reduction is not very meaningful when q is itself a very small prime, whereas those of [BFKL93, Reg05] are meaningful. This is only because our reduction deals with the *continuous* version of LWE. If we discretize the problem, then for very small prime q our reduction specializes to those of [BFKL93, Reg05].

for a fresh $r' \leftarrow \mathbb{Z}_q$ (where $\mathbf{e}_1 = (1, 0, \ldots, 0) \in \mathbb{Z}_q^n$). Observe that if $s_1 = 0 \bmod p$, the transformed samples are also drawn from $A^{j-1}_{\mathbf{s},\alpha'}$, otherwise they are drawn from $A^{j}_{\mathbf{s},\alpha'}$ because $r's_1$ is uniformly random modulo p. Therefore, $\mathcal{O}$ tells us which is the case.

Using the above test, we can efficiently recover $s_1 \bmod p$ by 'shifting' s_1 by each of $0, \ldots, p-1 \bmod p$ using the standard transformation that maps $A_{\mathbf{s},\alpha'}$ to $A_{\mathbf{s}+\mathbf{t},\alpha'}$ for any desired $\mathbf{t} \in \mathbb{Z}_q^n$, by taking $(\mathbf{a}, b)$ to $(\mathbf{a}, b + \langle \mathbf{a}, \mathbf{t} \rangle / q \bmod 1)$. (This enumeration step is where we use the fact that every p_i is $\mathrm{poly}(n)$-bounded.) Moreover, we can iteratively recover $s_1 \bmod p^2, \ldots, p^{e-j+1}$ as follows: having recovered $s_1 \bmod p^i$, first 'shift' $A_{\mathbf{s},\alpha'}$ to $A_{\mathbf{s}',\alpha'}$ where $s'_1 = 0 \bmod p^i$, then apply a similar procedure as above to recover $s'_1 \bmod p^{i+1}$: specifically, just modify the transformation in (3.1) to let $\mathbf{a}' = \mathbf{a} - r' \cdot (q/p^{j+i}) \cdot \mathbf{e}_1$, so that $b' = b = \langle \mathbf{a}', \mathbf{s} \rangle / q + e + (pr + r'(s'_1/p^i))/p^j$. This procedure works as long as p^{j+i} divides q, so we can recover $s_1 \bmod p^{e-j+1}$.

Using the above reductions and the Chinese remainder theorem, and letting j_i be the above minimal value of j for $p = p_i$ (of which at most ℓ of these are greater than 1), from $A_{\mathbf{s},\alpha}$ we can recover $\mathbf{s}$ modulo

$$P = \prod_i p_i^{e_i - (j_i - 1)} = q / \prod_i p_i^{j_i - 1} \geq q \cdot \left(\frac{\alpha'}{\omega(\sqrt{\log n})} \right)^{\ell} \geq q \cdot \alpha \cdot \omega(\sqrt{\log n}),$$

because $\alpha' < \omega(\sqrt{\log n})/p_i^{j_i - 1}$ for all i by definition of j_i and by hypothesis on α'. By applying the 'shift' transformation to $A_{\mathbf{s},\alpha}$ we can assume that $\mathbf{s} = 0 \bmod P$. Now every $\langle \mathbf{a}, \mathbf{s}' \rangle / q$ is an integer multiple of $P/q \geq \alpha \cdot \omega(\sqrt{\log n})$, and since every noise term $e \leftarrow D_\alpha$ has magnitude $< (\alpha/2) \cdot \omega(\sqrt{\log n})$ with overwhelming probability, we can round the second component of every $(\mathbf{a}, b) \leftarrow A_{\mathbf{s},\alpha}$ to the exact value of $\langle \mathbf{a}, \mathbf{s} \rangle / q \bmod 1$. From these we can solve for $\mathbf{s}$ by Gaussian elimination, and we are done. □

4 Primitive Lattices

At the heart of our new trapdoor generation algorithm (described in Section 5) is the construction of a very special family of lattices which have excellent geometric properties, and admit very fast and parallelizable decoding algorithms. The lattices are defined by means of what we call a *primitive matrix*. We say that a matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is primitive if its columns generate all of $\mathbb{Z}_q^n$, i.e., $\mathbf{G} \cdot \mathbb{Z}^m = \mathbb{Z}_q^n$.[6]

The main results of this section are summarized in the following theorem.

Theorem 4.1. *For any integers $q \geq 2$, $n \geq 1$, $k = \lceil \log_2 q \rceil$ and $m = nk$, there is a primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ such that*

- *The lattice $\Lambda^\perp(\mathbf{G})$ has a known basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ with $\|\widetilde{\mathbf{S}}\| \leq \sqrt{5}$ and $\|\mathbf{S}\| \leq \max\{\sqrt{5}, \sqrt{k}\}$. Moreover, when $q = 2^k$, we have $\widetilde{\mathbf{S}} = 2\mathbf{I}$ (so $\|\widetilde{\mathbf{S}}\| = 2$) and $\|\mathbf{S}\| = \sqrt{5}$.*
- *Both $\mathbf{G}$ and $\mathbf{S}$ require little storage. In particular, they are sparse (with only $O(m)$ nonzero entries) and highly structured.*
- *Inverting $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \bmod q$ can be performed in quasilinear $O(n \cdot \log^c n)$ time for any $\mathbf{s} \in \mathbb{Z}_q^n$ and any $\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$, where $\mathbf{B}$ can denote either $\mathbf{S}$ or $\widetilde{\mathbf{S}}$. Moreover, the algorithm is perfectly parallelizable, running in polylogarithmic $O(\log^c n)$ time using n processors. When $q = 2^k$, the polylogarithmic term $O(\log^c n)$ is essentially just the cost of k additions and shifts on k-bit integers.*

[6] We do not say that $\mathbf{G}$ is "full-rank," because $\mathbb{Z}_q$ is not a field when q is not prime, and the notion of rank for matrices over $\mathbb{Z}_q$ is not well defined.

- *Preimage sampling for $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\mathbf{x} \bmod q$ with Gaussian parameter $s \geq \|\widetilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$ can be performed in quasilinear $O(n \cdot \log^c n)$ time, or parallel polylogarithmic $O(\log^c n)$ time using n processors. When $q = 2^k$, the polylogarithmic term is essentially just the cost of k additions and shifts on k-bit integers, plus the (offline) generation of about m random integers drawn from $D_{\mathbb{Z},s}$.*

More generally, for any integer $b \geq 2$, all of the above statements hold with $k = \lceil \log_b q \rceil$, $\|\widetilde{\mathbf{S}}\| \leq \sqrt{b^2+1}$, and $\|\mathbf{S}\| \leq \max\{\sqrt{b^2+1}, (b-1)\sqrt{k}\}$; and when $q = b^k$, we have $\widetilde{\mathbf{S}} = b\mathbf{I}$ and $\|\mathbf{S}\| = \sqrt{b^2+1}$.

The rest of this section is dedicated to the proof of Theorem 4.1. In the process, we also make several important observations regarding the implementation of the inversion and sampling algorithms associated with $\mathbf{G}$, showing that our algorithms are not just asymptotically fast, but also quite practical.

Let $q \geq 2$ be an integer modulus and $k \geq 1$ be an integer dimension. Our construction starts with a *primitive vector* $\mathbf{g} \in \mathbb{Z}_q^k$, i.e., a vector such that $\gcd(g_1, \ldots, g_k, q) = 1$. The vector $\mathbf{g}$ defines a k-dimensional lattice $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$ having determinant $|\mathbb{Z}^k/\Lambda^\perp(\mathbf{g}^t)| = q$, because the residue classes of $\mathbb{Z}^k/\Lambda^\perp(\mathbf{g}^t)$ are in bijective correspondence with the possible values of $\langle \mathbf{g}, \mathbf{x} \rangle \bmod q$ for $\mathbf{x} \in \mathbb{Z}^k$, which cover all of $\mathbb{Z}_q$ since $\mathbf{g}$ is primitive. Concrete primitive vectors $\mathbf{g}$ will be described in the next subsections. Notice that when $q = \mathrm{poly}(n)$, we have $k = O(\log q) = O(\log n)$ and so $\Lambda^\perp(\mathbf{g}^t)$ is a very low-dimensional lattice. Let $\mathbf{S}_k \in \mathbb{Z}^{k \times k}$ be a basis of $\Lambda^\perp(\mathbf{g}^t)$, that is, $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \in \mathbb{Z}_q^{1 \times k}$ and $|\det(\mathbf{S}_k)| = q$.

The primitive vector $\mathbf{g}$ and associated basis $\mathbf{S}_k$ are used to define the parity-check matrix $\mathbf{G}$ and basis $\mathbf{S} \in \mathbb{Z}_q$ as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times nk}$ and $\mathbf{S} := \mathbf{I}_n \otimes \mathbf{S}_k \in \mathbb{Z}^{nk \times nk}$. That is,

$$\mathbf{G} := \begin{bmatrix} \cdots \mathbf{g}^t \cdots & & & \\ & \cdots \mathbf{g}^t \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^t \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}, \qquad \mathbf{S} := \begin{bmatrix} \mathbf{S}_k & & & \\ & \mathbf{S}_k & & \\ & & \ddots & \\ & & & \mathbf{S}_k \end{bmatrix} \in \mathbb{Z}^{nk \times nk}.$$

Equivalently, $\mathbf{G}$, $\Lambda^\perp(\mathbf{G})$, and $\mathbf{S}$ are the direct sums of n copies of $\mathbf{g}^t$, $\Lambda^\perp(\mathbf{g}^t)$, and $\mathbf{S}_k$, respectively. It follows that $\mathbf{G}$ is a primitive matrix, the lattice $\Lambda^\perp(\mathbf{G}) \subset \mathbb{Z}^{nk}$ has determinant q^n, and $\mathbf{S}$ is a basis for this lattice. It also follows (and is clear by inspection) that $\|\mathbf{S}\| = \|\mathbf{S}_k\|$ and $\|\widetilde{\mathbf{S}}\| = \|\widetilde{\mathbf{S}_k}\|$.

By this direct sum construction, it is immediate that inverting $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e})$ and sampling preimages of $f_{\mathbf{G}}(\mathbf{x})$ can be accomplished by performing the same operations n times in parallel for $g_{\mathbf{g}^t}$ and $f_{\mathbf{g}^t}$ on the corresponding portions of the input, and concatenating the results. For preimage sampling, if each of the $f_{\mathbf{g}^t}$ preimages has Gaussian parameter $\sqrt{\Sigma}$, then by independence, their concatenation has parameter $\mathbf{I}_n \otimes \sqrt{\Sigma}$. Likewise, inverting $g_{\mathbf{G}}$ will succeed whenever all the n independent $g_{\mathbf{g}^t}$-inversion subproblems are solved correctly.

In the next two subsections we study concrete instantiations of the primitive vector $\mathbf{g}$, and give optimized algorithms for inverting $g_{\mathbf{g}^t}$ and sampling preimages for $f_{\mathbf{g}^t}$. In both subsections, we consider primitive lattices $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$ defined by the vector

$$\mathbf{g}^t := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}, \qquad k = \lceil \log_2 q \rceil, \tag{4.1}$$

whose entries form a geometrically increasing sequence. (We focus on powers of 2, but all our results trivially extend to other integer powers, or even mixed-integer products.) The only difference between the two subsections is in the form of the modulus q. We first study the case when the modulus $q = 2^k$ is a power of 2, which leads to especially simple and fast algorithms. Then we discuss how the results can be generalized to arbitrary moduli q. Notice that in both cases, the syndrome $\langle \mathbf{g}, \mathbf{x} \rangle \in \mathbb{Z}_q$ of a binary

vector $\mathbf{x} = (x_0, \ldots, x_{k-1}) \in \{0,1\}^k$ is just the positive integer with binary expansion $\mathbf{x}$. In general, for arbitrary $\mathbf{x} \in \mathbb{Z}^k$ the syndrome $\langle \mathbf{g}, \mathbf{x} \rangle \in \mathbb{Z}_q$ can be computed very efficiently by a sequence of k additions and binary shifts, and a single reduction modulo q, which is also trivial when $q = 2^k$ is a power of 2. The syndrome computation is also easily parallelizable, leading to $O(\log k) = O(\log \log n)$ computation time using $O(k) = O(\log n)$ processors.

4.1 Power-of-Two Modulus

Let $q = 2^k$ be a power of 2, and let $\mathbf{g}$ be the geometric vector defined in Equation (4.1). Define the matrix

$$\mathbf{S}_k := \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}.$$

This is a basis for $\Lambda^\perp(\mathbf{g}^t)$, because $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \bmod q$ and $\det(\mathbf{S}_k) = 2^k = q$. Clearly, all the basis vectors are short. Moreover, by orthogonalizing $\mathbf{S}_k$ in reverse order, we have $\widetilde{\mathbf{S}_k} = 2 \cdot \mathbf{I}_k$. This construction is summarized in the following proposition. (It generalizes in the obvious way to any integer base, not just 2.)

Proposition 4.2. *For $q = 2^k$ and $\mathbf{g} = (1, 2, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$, the lattice $\Lambda^\perp(\mathbf{g}^t)$ has a basis $\mathbf{S}$ such that $\widetilde{\mathbf{S}} = 2\mathbf{I}$ and $\|\mathbf{S}\| \leq \sqrt{5}$. In particular, $\eta_\epsilon(\Lambda^\perp(\mathbf{g}^t)) \leq 2r = 2 \cdot \omega(\sqrt{\log n})$ for some $\epsilon(n) = \mathrm{negl}(n)$.*

Using Proposition 4.2 and known generic algorithms [Bab85, Kle00, GPV08], it is possible to invert $g_{\mathbf{g}^t}(\mathbf{s}, \mathbf{e})$ correctly whenever $\mathbf{e} \in \mathcal{P}_{1/2}((q/2) \cdot \mathbf{I})$, and sample preimages under $f_{\mathbf{g}^t}$ with Gaussian parameter $s \geq 2r = 2 \cdot \omega(\sqrt{\log n})$. In what follows we show how the special structure of the basis $\mathbf{S}$ leads to simpler, faster, and more practical solutions to these general lattice problems.

Inversion. Here we show how to efficiently find an unknown scalar $s \in \mathbb{Z}_q$ given $\mathbf{b}^t = [b_0, b_1, \ldots, b_{k-1}] = s \cdot \mathbf{g}^t + \mathbf{e}^t = [s + e_0, 2s + e_1, \ldots, 2^{k-1}s + e_{k-1}] \bmod q$, where $\mathbf{e} \in \mathbb{Z}^k$ is a short error vector.

An iterative algorithm works by recovering the binary digits $s_0, s_1, \ldots, s_{k-1} \in \{0,1\}$ of $s \in \mathbb{Z}_q$, from least to most significant, as follows: first, determine s_0 by testing whether

$$b_{k-1} = 2^{k-1}s + e_{k-1} = (q/2)s_0 + e_{k-1} \bmod q$$

is closer to 0 or to $q/2$ (modulo q). Then recover s_1 from $b_{k-2} = 2^{k-2}s + e_{k-2} = 2^{k-1}s_1 + 2^{k-2}s_0 + e_{k-2} \bmod q$, by subtracting $2^{k-2}s_0$ and testing proximity to 0 or $q/2$, etc. It is easy to see that the algorithm produces correct output if every $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right)$, i.e., if $\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{I}_k/2) = \mathcal{P}_{1/2}(q \cdot (\widetilde{\mathbf{S}_k})^{-t})$. It can also be seen that this algorithm is exactly Babai's "nearest-plane" algorithm [Bab85], specialized to the scaled dual $q(\mathbf{S}_k)^{-t}$ of the basis $\mathbf{S}_k$ of $\Lambda^\perp(\mathbf{g}^t)$, which is a basis for $\Lambda(\mathbf{g})$.

Formally, the iterative algorithm is: given a vector $\mathbf{b}^t = [b_0, \ldots, b_{k-1}] \in \mathbb{Z}_q^{1 \times k}$, initialize $s \leftarrow 0$.

1. For $i = k-1, \ldots, 0$: let $s \leftarrow s + 2^{k-1-i} \cdot \left[b_i - 2^i \cdot s \notin \left[-\frac{q}{4}, \frac{q}{4}\right) \bmod q\right]$, where $[E] = 1$ if expression E is true, and 0 otherwise. Also let $e_i \leftarrow b_i - 2^i \cdot s \in \left[-\frac{q}{4}, \frac{q}{4}\right)$.

2. Output $s \in \mathbb{Z}_q$ and $\mathbf{e} = (e_0, \ldots, e_{k-1}) \in \left[-\frac{q}{4}, \frac{q}{4}\right)^k \subset \mathbb{Z}^k$.

Note that for $x \in \{0, \ldots, q-1\}$ with binary representation $(x_{k-1}x_{k-2}\cdots x_0)_2$, we have

$$\left[x \notin \left[-\tfrac{q}{4}, \tfrac{q}{4}\right) \bmod q\right] = x_{k-1} \oplus x_{k-2}.$$

There is also a non-iterative approach to decoding using a lookup table, and a hybrid approach between the two extremes. Notice that rounding each entry b_i of $\mathbf{b}$ to the nearest multiple of 2^i (modulo q, breaking ties upward) before running the above algorithm does not change the value of s that is computed. This lets us precompute a lookup table that maps the $2^{k(k+1)/2} = q^{O(\lg q)}$ possible rounded values of $\mathbf{b}$ to the correct values of s. The size of this table grows very rapidly for $k > 3$, but in this case we can do better if we assume slightly smaller error terms $e_i \in \left[-\frac{q}{8}, \frac{q}{8}\right)$: simply round each b_i to the nearest multiple of $\max\{\frac{q}{8}, 2^i\}$, thus producing one of exactly $8^{k-1} = q^3/8$ possible results, whose solutions can be stored in a lookup table. Note that the result is correct, because in each coordinate the total error introduced by e_i and rounding to a multiple of $\frac{q}{8}$ is in the range $\left[-\frac{q}{4}, \frac{q}{4}\right)$. A hybrid approach combining the iterative algorithm with table lookups of ℓ bits of s at a time is potentially the most efficient option in practice, and is easy to devise from the above discussion.

Gaussian sampling. We now consider the preimage sampling problem for function $f_{\mathbf{g}^t}$, i.e., the task of Gaussian sampling over a desired coset of $\Lambda^\perp(\mathbf{g}^t)$. More specifically, we want to sample a vector from the set $\Lambda^\perp_u(\mathbf{g}^t) = \{\mathbf{x} \in \mathbb{Z}^k : \langle \mathbf{g}, \mathbf{x}\rangle = u \bmod q\}$ for a desired syndrome $u \in \mathbb{Z}_q$, with probability proportional to $\rho_s(\mathbf{x})$. We wish to do so for any fixed Gaussian parameter $s \geq \|\widetilde{\mathbf{S}_k}\| \cdot r = 2 \cdot \omega(\sqrt{\log n})$, which is an optimal bound on the smoothing parameter of $\Lambda^\perp(\mathbf{G})$.

As with inversion, there are two main approaches to Gaussian sampling, which are actually opposite extremes on a spectrum of storage/parallelism trade-offs. The first approach is essentially to precompute and store many independent samples $\mathbf{x} \leftarrow D_{\mathbb{Z}^k,s}$, 'bucketing' them based on the value of $\langle \mathbf{g}, \mathbf{x}\rangle \in \mathbb{Z}_q$ until there is at least one sample per bucket. Because each $\langle \mathbf{g}, \mathbf{x}\rangle$ is statistically close to uniform over $\mathbb{Z}_q$ (by the smoothing parameter bound for $\Lambda^\perp(\mathbf{g}^t)$), a coupon-collecting argument implies that we need to generate about $q \log q$ samples to occupy every bucket. The online part of the sampling algorithm for $\Lambda^\perp(\mathbf{g}^t)$ is trivial, merely taking a fresh $\mathbf{x}$ from the appropriate bucket. The downside is that the storage and precomputation requirements are rather high: in many applications, q (while polynomial in the security parameter) can be in the many thousands or more.

The second approach exploits the niceness of the orthogonalized basis $\widetilde{\mathbf{S}_k} = 2\mathbf{I}_k$. Using this basis, the randomized nearest-plane algorithm of [Kle00, GPV08] becomes very simple and efficient, and is equivalent to the following: given a syndrome $u \in \{0, \ldots, q-1\}$ (viewed as an integer),

1. For $i = 0, \ldots, k-1$: choose $x_i \leftarrow D_{2\mathbb{Z}+u,s}$ and let $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$.
2. Output $\mathbf{x} = (x_0, \ldots, x_{k-1})$.

Observe that every Gaussian x_i in the above algorithm is chosen from one of only two possible cosets of $2\mathbb{Z}$, determined by the least significant bit of u at that moment. Therefore, we may precompute and store several independent Gaussian samples from each of $2\mathbb{Z}$ and $2\mathbb{Z}+1$, and consume one per iteration when executing the algorithm. (As above, the individual samples may be generated by choosing several $x \leftarrow D_{\mathbb{Z},s}$ and bucketing each one according to its least-significant bit.) Such presampling makes the algorithm deterministic during its online phase, and because there are only two cosets, there is almost no wasted storage or precomputation. Notice, however, that this algorithm requires $k = \lg(q)$ sequential iterations.

Between the extremes of the two algorithms described above, there is a hybrid algorithm that chooses $\ell \geq 1$ entries of $\mathbf{x}$ at a time. (For simplicity, we assume that ℓ divides k exactly, though this is not

strictly necessary.) Let $\mathbf{h}^t = [1, 2, \ldots, 2^{\ell-1}] \in \mathbb{Z}_{2^\ell}^{1\times\ell}$ be a parity-check matrix defining the 2^ℓ-ary lattice $\Lambda^\perp(\mathbf{h}^t) \subseteq \mathbb{Z}^\ell$, and observe that $\mathbf{g}^t = [\mathbf{h}^t, 2^\ell \cdot \mathbf{h}^t, \ldots, 2^{k-\ell} \cdot \mathbf{h}^t]$. The hybrid algorithm then works as follows:

1. For $i = 0, \ldots, k/\ell - 1$, choose $(x_{i\ell}, \ldots, x_{(i+1)\ell-1}) \leftarrow D_{\Lambda^\perp_{u \bmod 2^\ell}(\mathbf{h}^t), s}$ and let $u \leftarrow (u - x)/2^\ell$, where $x = \sum_{j=0}^{\ell-1} x_{i\ell+j} \cdot 2^j \in \mathbb{Z}$.
2. Output $\mathbf{x} = (x_0, \ldots, x_{k-1})$.

As above, we can precompute samples $\mathbf{x} \leftarrow D_{\mathbb{Z}^\ell, s}$ and store them in a lookup table having 2^ℓ buckets, indexed by the value $\langle \mathbf{h}, \mathbf{x} \rangle \in \mathbb{Z}_{2^\ell}$, thereby making the algorithm deterministic in its online phase.

4.2 Arbitrary Modulus

For a modulus q that is not a power of 2, most of the above ideas still work, with slight adaptations. Let $k = \lceil \lg(q) \rceil$, so $q < 2^k$. As above, define $\mathbf{g}^t := [1, 2, \ldots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$, but now define the matrix

$$\mathbf{S}_k := \begin{bmatrix} 2 & & & & q_0 \\ -1 & 2 & & & q_1 \\ & -1 & & & q_2 \\ & & \ddots & & \vdots \\ & & & 2 & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix} \in \mathbb{Z}^{k\times k}$$

where $(q_0, \ldots, q_{k-1}) \in \{0,1\}^k$ is the binary expansion of $q = \sum_i 2^i \cdot q_i$. Again, $\mathbf{S}$ is a basis of $\Lambda^\perp(\mathbf{g}^t)$ because $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \bmod q$, and $\det(\mathbf{S}_k) = q$. Moreover, the basis vectors have squared length $\|\mathbf{s}_i\|^2 = 5$ for $i < k$ and $\|\mathbf{s}_k\|^2 = \sum_i q_i \le k$. The next lemma shows that $\mathbf{S}_k$ also has a good Gram-Schmidt orthogonalization.

Lemma 4.3. *With $\mathbf{S} = \mathbf{S}_k$ defined as above and orthogonalized in forward order, we have $\|\widetilde{\mathbf{s}_i}\|^2 = \frac{4-4^{-i}}{1-4^{-i}} \in (4, 5]$ for $1 \le i < k$, and $\|\widetilde{\mathbf{s}_k}\|^2 = \frac{3q^2}{4^k-1} < 3$.*

Proof. Notice that the the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_{k-1}$ are all orthogonal to $\mathbf{g}_k = (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}^k$. Thus, the orthogonal component of $\mathbf{s}_k$ has squared length

$$\|\widetilde{\mathbf{s}_k}\|^2 = \frac{\langle \mathbf{s}_k, \mathbf{g}_k \rangle^2}{\|\mathbf{g}_k\|^2} = \frac{q^2}{\sum_{j<k} 4^j} = \frac{3q^2}{4^k - 1}.$$

Similarly, the squared length of $\widetilde{\mathbf{s}_i}$ for $i < k$ can be computed as

$$\|\widetilde{\mathbf{s}_i}\|^2 = 1 + \frac{4^i}{\sum_{j<i} 4^j} = \frac{4 - 4^{-i}}{1 - 4^{-i}}.$$ □

This concludes the description and analysis of the primitive lattice $\Lambda^\perp(\mathbf{g}^t)$ when q is not a power of 2. Specialized inversion algorithms can also be adapted as well, but some care is needed. Of course, since the lattice dimension $k = O(\log n)$ is very small, one could simply use the general methods of [Bab85, Kle00, GPV08, Pei10] without worrying too much about optimizations, and satisfy all the claims made in Theorem 4.1. Below we briefly discuss alternatives for Gaussian sampling.

The offline 'bucketing' approach to Gaussian sampling works without any modification for arbitrary modulus, with just slighly larger Gaussian parameter $s \geq \sqrt{5} \cdot r$, because it relies only on the smoothing parameter bound of $\eta_\epsilon(\Lambda^\perp(\mathbf{g}^t)) \leq \|\widetilde{\mathbf{S}_k}\| \cdot \omega(\sqrt{\log n})$ and the fact that the number of buckets is q. The randomized nearest-plane approach to sampling does not admit a specialization as simple as the one we have described for $q = 2^k$. The reason is that while the basis $\mathbf{S}$ is sparse, its orthogonalization $\widetilde{\mathbf{S}}$ is not sparse in general. (This is in contrast to the case when $q = 2^k$, for which orthogonalizing in reverse order leads to the sparse matrix $\widetilde{\mathbf{S}} = 2\mathbf{I}$.) Still, $\widetilde{\mathbf{S}}$ is "almost triangular," in the sense that the off-diagonal entries decrease geometrically as one moves away from the diagonal. This may allow for optimizing the sampling algorithm by performig "truncated" scalar product computations, and still obtain an almost-Gaussian distribution on the resulting samples. An interesting alternative is to use a hybrid approach, where one first performs a single iteration of randomized nearest-plane algorithm to take care of the last basis vector $\mathbf{s}_k$, and then performs some variant of the convolution algorithm from [Pei10] to deal with the first $k-1$ basis vectors $[\mathbf{s}_1, \ldots, \mathbf{s}_{k-1}]$, which have very small lengths and singular values. Notice that the orthogonalized component of the last vector $\mathbf{s}_k$ is simply a scalar multiple of the primitive vector $\mathbf{g}$, so the scalar product $\langle \mathbf{s}_k, \mathbf{t} \rangle$ (for any vector $\mathbf{t}$ with syndrome $u = \langle \mathbf{g}, \mathbf{t} \rangle$) can be immediately computed from u as u/q (see Lemma 4.3).

4.3 The Ring Setting

The above constructions and algorithms all transfer easily to compact lattices defined over polynomial rings (i.e., number rings), as used in the representative works [Mic02, PR06, LM06, LPR10]. A commonly used example is the cyclomotic ring $R = \mathbb{Z}[x]/(\Phi_m(x))$ where $\Phi_m(x)$ denotes the mth cyclotomic polynomial, which is a monic, degree-$\varphi(m)$, irreducible polynomial whose zeros are all the *primitive* mth roots of unity in $\mathbb{C}$. The ring R is a $\mathbb{Z}$-module of rank n, i.e., it is generated as the additive integer combinations of the "power basis" elements $1, x, x^2, \ldots, x^{\varphi(m)-1}$. We let $R_q = R/qR$, the ring modulo the ideal generated by an integer q. For geometric concepts like error vectors and Gaussian distributions, it is usually nicest to work with the "canonical embedding" of R, which roughly (but not exactly) corresponds with the "coefficient embedding," which just considers the vector of coefficients relative to the power basis.

Let $\mathbf{g} \in R_q^k$ be a primitive vector modulo q, i.e., one for which the ideal generated by $q, g_1, \ldots, g_k$ is the full ring R. As above, the vector $\mathbf{g}$ defines functions $f_{\mathbf{g}^t} \colon R^k \to R_q$ and $g_{\mathbf{g}^t} \colon R_q \times R^k \to R_q^{1 \times k}$, defined as $f_{\mathbf{g}^t}(\mathbf{x}) = \langle \mathbf{g}, \mathbf{x} \rangle = \sum_{i=1}^k g_i \cdot x_i \bmod q$ and $g_{\mathbf{g}^t}(s, \mathbf{e}) = s \cdot \mathbf{g}^t + \mathbf{e}^t \bmod q$, and the related R-module

$$qR^k \quad \subseteq \quad \Lambda^\perp(\mathbf{g}^t) := \{\mathbf{x} \in R^k : f_{\mathbf{g}^t}(\mathbf{x}) = \langle \mathbf{g}, \mathbf{x} \rangle = 0 \bmod q\} \quad \subsetneq \quad R^k,$$

which has index (determinant) $q^n = |R_q|$ as an additive subgroup of R^k because $\mathbf{g}$ is primitive. Concretely, we can use the exact same primitive vector $\mathbf{g}^t = [1, 2, \ldots, 2^{k-1}] \in R_q^k$ as in Equation (4.1), interpreting its entries in the ring R_q rather than $\mathbb{Z}_q$.

Inversion and preimage sampling algorithms for $g_{\mathbf{g}^t}$ and $f_{\mathbf{g}^t}$ (respectively) are relatively straightforward to obtain, by adapting the basic approaches from the previous subsections. These algorithms are simplest when the power basis elements $1, x, x^2, \ldots, x^{\varphi(m)-1}$ are *orthogonal* under the canonical embedding (which is the case exactly when m is a power of 2, and hence $\Phi_m(x) = x^{m/2} + 1$), because the inversion operations reduce to parallel operations relative to each of the power basis elements. We defer the details to the full version.

5 Trapdoor Generation and Operations

In this section we describe our new trapdoor generation, inversion and sampling algorithms for hard random lattices. Recall that these are lattices $\Lambda^{\perp}(\mathbf{A})$ defined by an (almost) uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and that the standard notion of a "strong" trapdoor for these lattices (put forward in [GPV08] and used in a large number of subsequent applications) is a short lattice basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ for $\Lambda^{\perp}(\mathbf{A})$. There are several measures of quality for the trapdoor $\mathbf{S}$, the most common ones being (in nondecreasing order): the maximal Gram-Schmidt length $\|\widetilde{\mathbf{S}}\|$; the maximal Euclidean length $\|\mathbf{S}\|$; and the maximal singular value $s_1(\mathbf{S})$. Algorithms for generating random lattices together with high-quality trapdoor bases are given in [Ajt99, AP09]. In this section we give much simpler, faster and tighter algorithms to generate a hard random lattice with a trapdoor, and to use a trapdoor for performing standard tasks like inverting the LWE function $g_{\mathbf{A}}$ and sampling preimages for the SIS function $f_{\mathbf{A}}$. We also give a new, simple algorithm for delegating a trapdoor, i.e., using a trapdoor for $\mathbf{A}$ to obtain one for a matrix $[\mathbf{A} \mid \mathbf{A}']$ that extends $\mathbf{A}$, in a secure and non-reversible way.

The following theorem summarizes the main results of this section. Here we state just one typical instantiation with only asymptotic bounds. More general results and exact bounds are presented throughout the section.

Theorem 5.1. *There is an efficient randomized algorithm* $\mathsf{GenTrap}(1^n, 1^m, q)$ *that, given any integers* $n \geq 1$, $q \geq 2$, *and sufficiently large* $m = O(n \log q)$, *outputs a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and a 'trapdoor'* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\mathrm{negl}(n)$*-far from uniform. Moreover, there are efficient algorithms* Invert *and* SampleD *that with overwhelming probability over all random choices, do the following:*

- *For* $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, *where* $\mathbf{s} \in \mathbb{Z}_q^n$ *is arbitrary and either* $\|\mathbf{e}\| < q/O(\sqrt{n \log q})$ *or* $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega(\sqrt{\log n})$, *the deterministic algorithm* $\mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ *outputs* $\mathbf{s}$ *and* $\mathbf{e}$.
- *For any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and large enough* $s = O(\sqrt{n \log q})$, *the randomized algorithm* $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$ *samples from a distribution within* $\mathrm{negl}(n)$ *statistical distance of* $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}), s \cdot \omega(\sqrt{\log n})}$.

Throughout this section, we let $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ denote some fixed primitive matrix that admits efficient inversion and preimage sampling algorithms, as described in Theorem 4.1. (Recall that typically, $w = n\lceil \log q \rceil$ for some appropriate base of the logarithm.) All our algorithms and efficiency improvements are based on the primitive matrix $\mathbf{G}$ and associated algorithms described in Section 4, and a new notion of trapdoor that we define next.

5.1 A New Trapdoor Notion

We begin by defining the new notion of trapdoor, establish some of its most important properties, and give a simple and efficient algorithm for generating hard random lattices together with high-quality trapdoors.

Definition 5.2. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ be matrices with $m \geq w \geq n$. A $\mathbf{G}$*-trapdoor* for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{(m-w) \times w}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}\mathbf{G}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. We refer to $\mathbf{H}$ as the *tag* or *label* of the trapdoor. The *quality* of the trapdoor is measured by its largest singular value $s_1(\mathbf{R})$.

We remark that, by definition of $\mathbf{G}$-trapdoor, if $\mathbf{G}$ is a primitive matrix and $\mathbf{A}$ admits a $\mathbf{G}$ trapdoor, then $\mathbf{A}$ is primitive as well. In particular, $\det(\Lambda^{\perp}(\mathbf{A})) = q^n$. Since the primitive matrix $\mathbf{G}$ is typically fixed and public, we usually omit references to it, and refer to $\mathbf{G}$-trapdoors simply as trapdoors. We remark that since

$\mathbf{G}$ is primitive, the tag $\mathbf{H}$ in the above definition is uniquely determined by (and efficiently computable from) $\mathbf{A}$ and the trapdoor $\mathbf{R}$.

The following lemma says that a good basis for $\Lambda^\perp(\mathbf{A})$ may be obtained from knowledge of $\mathbf{R}$. We do not use the lemma anywhere in the rest of the paper, but include it here primarily to show that our new definition of trapdoor is at least as powerful as the traditional one of a short basis. Our algorithms for Gaussian sampling and LWE inversion do not need a full basis, and make direct (and more efficient) use of our new notion of trapdoor.

Lemma 5.3. *Let $\mathbf{S} \in \mathbb{Z}^{w \times w}$ be any basis for $\Lambda^\perp(\mathbf{G})$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ have trapdoor $\mathbf{R} \in \mathbb{Z}^{(m-w) \times w}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. Then the lattice $\Lambda^\perp(\mathbf{A})$ is generated by the basis*

$$\mathbf{S_A} = \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{bmatrix},$$

where $\mathbf{W} \in \mathbb{Z}^{w \times \bar{m}}$ is an arbitrary solution to $\mathbf{GW} = -\mathbf{H}^{-1}\mathbf{A}[\mathbf{I} \mid \mathbf{0}]^T \bmod q$. Moreover, the basis $\mathbf{S_A}$ satisfies $\|\widetilde{\mathbf{S_A}}\| \leq s_1(\left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]) \cdot \|\widetilde{\mathbf{S}}\| \leq (s_1(\mathbf{R}) + 1) \cdot \|\widetilde{\mathbf{S}}\|$, when $\mathbf{S_A}$ is orthogonalized in suitable order.

Proof. It is immediate to check that $\mathbf{A} \cdot \mathbf{S_A} = \mathbf{0} \bmod q$, so $\mathbf{S_A}$ generates a sublattice of $\Lambda^\perp(\mathbf{A})$. In fact, it generates the entire lattice because $\det(\mathbf{S_A}) = \det(\mathbf{S}) = q^n = \det(\Lambda^\perp(\mathbf{A}))$.

The bound on $\|\widetilde{\mathbf{S_A}}\|$ follows by simple linear algebra. Recall by Item 3 of Lemma 2.1 that $\|\widetilde{\mathbf{B}}\| = \|\widetilde{\mathbf{S}}\|$ when the columns of $\mathbf{B} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{smallmatrix}\right]$ are reordered appropriately. So it suffices to show that $\|\widetilde{\mathbf{TB}}\| \leq s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$ for any $\mathbf{T}$, $\mathbf{B}$. Let $\mathbf{B} = \mathbf{QDU}$ and $\mathbf{TB} = \mathbf{Q'D'U'}$ be Gram-Schmidt decompositions of $\mathbf{B}$ and $\mathbf{TB}$, respectively, with $\mathbf{Q}, \mathbf{Q'}$ orthogonal, $\mathbf{D}, \mathbf{D'}$ diagonal with nonnegative entries, and $\mathbf{U}, \mathbf{U'}$ upper unitriangular. We have

$$\mathbf{TQDU} = \mathbf{Q'D'U'} \Longrightarrow \mathbf{T'D} = \mathbf{D'U''},$$

where $\mathbf{T} = \mathbf{Q'T'Q}^{-1} \Rightarrow s_1(\mathbf{T'}) = s_1(\mathbf{T})$, and $\mathbf{U''}$ is upper unitriangular because such matrices form a multiplicative group. Now every row of $\mathbf{T'D}$ has Euclidean norm at most $s_1(\mathbf{T}) \cdot \|\mathbf{D}\| = s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$, while the ith row of $\mathbf{D'U''}$ has norm at least $d'_{i,i}$, the ith diagonal of $\mathbf{D'}$. We conclude that $\|\widetilde{\mathbf{TB}}\| = \|\mathbf{D}\| \leq s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$, as desired. □

We also make the following simple but useful observations:

- The rows of $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$ in Definition 5.2 can appear in any order, since this just induces a permutation of $\mathbf{A}$'s columns.
- If $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, then it can be made into an equally good trapdoor for any extension $[\mathbf{A} \mid \mathbf{B}]$, by padding $\mathbf{R}$ with zero rows; this leaves $s_1(\mathbf{R})$ unchanged.
- If $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H}$, then $\mathbf{R}$ is also a trapdoor for $\mathbf{A'} = \mathbf{A} - [\mathbf{0} \mid \mathbf{H'G}]$ with tag $(\mathbf{H} - \mathbf{H'})$ for any $\mathbf{H'} \in \mathbb{Z}_q^{n \times n}$, as long as $(\mathbf{H} - \mathbf{H'})$ is invertible modulo q. This is the main idea behind the compact IBE of [ABB10a], and can be used to give a family of "tag-based" trapdoor functions [KMO10]. In Section 6 we give explicit families of matrices $\mathbf{H}$ having suitable properties for applications.

5.2 Trapdoor Generation

We now give an algorithm to generate a (pseudo)random matrix $\mathbf{A}$ together with a $\mathbf{G}$-trapdoor. The algorithm is straightforward, and in fact it can be easily derived from the definition of $\mathbf{G}$-trapdoor itself. A random

lattice is built by first extending the primitive matrix $\mathbf{G}$ into a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{HG}]$ (where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ is chosen at random, and $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is the desired tag), and then applying a random transformation $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right] \in \mathbb{Z}^{m \times m}$ to the semi-random lattice $\Lambda^\perp(\mathbf{A}')$. Since $\mathbf{T}$ is unimodular with inverse $\mathbf{T}^{-1} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$, by Lemma 2.1 this yields the lattice $\mathbf{T} \cdot \Lambda^\perp(\mathbf{A}') = \Lambda^\perp(\mathbf{A}' \cdot \mathbf{T}^{-1})$ associated with the parity-check matrix $\mathbf{A} = \mathbf{A}' \cdot \mathbf{T}^{-1} = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}]$. Moreover, the distribution of $\mathbf{A}$ is close to uniform (either statistically, or computationally) as long as the distribution of $[\bar{\mathbf{A}} \mid \mathbf{0}]\mathbf{T}^{-1} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$ is. For details, see Algorithm 1, whose correctness is immediate.

Algorithm 1 Efficient algorithm $\mathsf{GenTrap}^{\mathcal{D}}(\bar{\mathbf{A}}, \mathbf{H})$ for generating a parity-check matrix $\mathbf{A}$ with trapdoor $\mathbf{R}$.

Input: Matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ for some $\bar{m} \geq 1$, invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, and distribution $\mathcal{D}$ over $\mathbb{Z}^{\bar{m} \times w}$.
(If no particular $\bar{\mathbf{A}}$, $\mathbf{H}$ are given as input, then the algorithm may choose them itself, e.g., picking $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random, and setting $\mathbf{H} = \mathbf{I}$.)

Output: A parity-check matrix $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n \times m}$, where $m = \bar{m} + w$, and trapdoor $\mathbf{R}$ with tag $\mathbf{H}$.

1: Choose a matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$ from distribution $\mathcal{D}$.
2: Output $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$.

We next describe two types of $\mathsf{GenTrap}$ instantiations. The first type generates a trapdoor $\mathbf{R}$ for a statistically near-uniform output matrix $\mathbf{A}$ using dimension $\bar{m} \approx n \log q$ or less (there is a trade-off between $\bar{m}$ and the trapdoor quality $s_1(\mathbf{R})$). The second types generates a computationally *pseudorandom* $\mathbf{A}$ (under the LWE assumption) using dimension $\bar{m} = 2n$; this pseudorandom construction is the first of its kind in the literature. Certain applications allow for an optimization that decreases $\bar{m}$ by an additive n term; this is most significant in the computationally secure construction because it yields $\bar{m} = n$.

Statistical instantiation. This instantiation works for any parameter $\bar{m}$ and distribution $\mathcal{D}$ over $\mathbb{Z}^{\bar{m} \times w}$ having the following two properties:

1. *Subgaussianity*: $\mathcal{D}$ is subgaussian with some parameter $s > 0$ (or δ-subgaussian for some small δ). This implies by Lemma 2.9 that $\mathbf{R} \leftarrow \mathcal{D}$ has $s_1(\mathbf{R}) = s \cdot O(\sqrt{\bar{m}} + \sqrt{w})$, except with probability $2^{-\Omega(\bar{m}+w)}$. (Recall that the constant factor hidden in the $O(\cdot)$ expression is $\approx 1/\sqrt{2\pi}$.)
2. *Regularity*: for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, $\mathbf{A} = [\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is δ-uniform for some $\delta = \mathrm{negl}(n)$.

   In fact, there is no loss in security if $\bar{\mathbf{A}}$ contains an identity matrix $\mathbf{I}$ as a submatrix and is otherwise uniform, since this corresponds with the Hermite normal form of the SIS and LWE problems. See, e.g., [MR09, Section 5] for further details.

For example, let $\mathcal{D} = \mathcal{P}^{\bar{m} \times w}$ where $\mathcal{P}$ is the distribution over $\mathbb{Z}$ that outputs 0 with probability $1/2$, and ± 1 each with probability $1/4$. Then $\mathcal{P}$ (and hence $\mathcal{D}$) is 0-subgaussian with parameter $\sqrt{2\pi}$, and satisfies the regularity condition (for any q) for $\delta \leq \frac{w}{2}\sqrt{q^n/2^{\bar{m}}}$, by a version of the leftover hash lemma (see, e.g., [AP09, Section 2.2.1]). Therefore, we can use any $\bar{m} \geq n \lg q + 2 \lg \frac{w}{2\delta}$.

As another important example, let $\mathcal{D} = D_{\mathbb{Z},s}^{\bar{m} \times w}$ be a discrete Gaussian distribution for some $s \geq \eta_\epsilon(\mathbb{Z})$ and $\epsilon = \mathrm{negl}(n)$. Then $\mathcal{D}$ is 0-subgaussian with parameter s by Lemma 2.8, and satisfies the regularity condition when $\bar{m}$ satisfies the bound (2.2) from Lemma 2.4. For example, letting $s = 2\eta_\epsilon(\mathbb{Z})$ we can use any $\bar{m} = n \lg q + \omega(\log n)$. (Other tradeoffs between s and $\bar{m}$ are possible, potentially using a different choice of $\mathbf{G}$, and more exact bounds on the error probabilities can be worked out from the lemma statements.) Moreover, by Lemmas 2.4 and 2.8 we have that with overwhelming probability over the choice of $\bar{\mathbf{A}}$, the

conditional distribution of $\mathbf{R}$ given $\mathbf{A} = [\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is $\mathrm{negl}(n)$-subgaussian with parameter s. We will use this fact in some of our applications in Section 6.

Computational instantiation. Let $\bar{\mathbf{A}} = [\mathbf{I} \mid \hat{\mathbf{A}}] \in \mathbb{Z}_q^{n \times \bar{m}}$ for $\bar{m} = 2n$, and let $\mathcal{D} = D_{\mathbb{Z},s}^{\bar{m} \times w}$ for some $s = \alpha q$, where $\alpha > 0$ is an LWE relative error rate (and typically $\alpha q > \sqrt{n}$). Clearly, $\mathcal{D}$ is 0-subgaussian with parameter αq. Also, $[\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R} = \hat{\mathbf{A}}\mathbf{R}_2 + \mathbf{R}_1]$ for $\mathbf{R} = \left[\begin{smallmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{smallmatrix}\right] \leftarrow \mathcal{D}$ is exactly an instance of decision-$\mathsf{LWE}_{n,q,\alpha}$ (in its normal form), and hence is pseudorandom (ignoring the identity submatrix) assuming that the problem is hard.

Further optimizations. If an application only uses a single tag $\mathbf{H} = \mathbf{I}$ (as is the case with, for example, GPV signatures [GPV08]), then we can save an additive n term in the dimension $\bar{m}$ (and hence in the total dimension m): instead of putting an identity submatrix in $\bar{\mathbf{A}}$, we can instead use the identity submatrix from $\mathbf{G}$ (which exists without loss of generality, since $\mathbf{G}$ is primitive) and conceal the remainder of $\mathbf{G}$ using either of the above methods.

All of the above ideas also translate immediately to the ring setting (see Section 4.3), using an appropriate regularity lemma (e.g., the one in [LPR10]) for a statistical instantiation, and the ring-LWE problem for a computationally secure instantiation.

5.3 LWE Inversion

Algorithm 2 below shows how to use a trapdoor to solve LWE relative to $\mathbf{A}$. Given a trapdoor $\mathbf{R}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and an LWE instance $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$ for some short error vector $\mathbf{e} \in \mathbb{Z}^m$, the algorithm recovers $\mathbf{s}$ (and $\mathbf{e}$). This naturally yields an inversion algorithm for the injective trapdoor function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$, which is hard to invert (and whose output is pseudorandom) if LWE is hard.

Algorithm 2 Efficient algorithm $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ for inverting the function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$.

Input: An oracle $\mathcal{O}$ for inverting the function $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ when $\hat{\mathbf{e}} \in \mathbb{Z}^w$ is suitably small.

- parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$;
- $\mathbf{G}$-trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times kn}$ for $\mathbf{A}$ with invertible tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$;
- vector $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$ for any $\mathbf{s} \in \mathbb{Z}_q^n$ and suitably small $\mathbf{e} \in \mathbb{Z}^m$.

Output: The vectors $\mathbf{s}$ and $\mathbf{e}$.

1: Compute $\hat{\mathbf{b}}^t = \mathbf{b}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$.
2: Get $(\hat{\mathbf{s}}, \hat{\mathbf{e}}) \leftarrow \mathcal{O}(\hat{\mathbf{b}})$.
3: **return** $\mathbf{s} = \mathbf{H}^{-t}\hat{\mathbf{s}}$ and $\mathbf{e} = \mathbf{b} - \mathbf{A}^t\mathbf{s}$ (interpreted as a vector in $\mathbb{Z}^m$ with entries in $[-\frac{q}{2}, \frac{q}{2})$).

Theorem 5.4. *Suppose that oracle $\mathcal{O}$ in Algorithm 2 correctly inverts $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ for any error vector $\hat{\mathbf{e}} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ for some $\mathbf{B}$. Then for any $\mathbf{s}$ and $\mathbf{e}$ of length $\|\mathbf{e}\| < q/(2\|\mathbf{B}\|s)$ where $s = \sqrt{s_1(\mathbf{R})^2 + 1}$, Algorithm 2 correctly inverts $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$. Moreover, for any $\mathbf{s}$ and random $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ where $1/\alpha \geq 2\|\mathbf{B}\|s \cdot \omega(\sqrt{\log n})$, the algorithm inverts successfully with overwhelming probability over the choice of $\mathbf{e}$.*

Note that using our constructions from Section 4, we can implement $\mathcal{O}$ so that either $\|\mathbf{B}\| = 2$ (for q a power of 2, where $\mathbf{B} = \widetilde{\mathbf{S}} = 2\mathbf{I}$) or $\|\mathbf{B}\| = \sqrt{5}$ (for arbitrary q).

Proof. Let $\bar{\mathbf{R}} = [\,\mathbf{R}^t\ \mathbf{I}\,]$, and note that $s = s_1(\bar{\mathbf{R}})$. By the above description, the algorithm works correctly when $\bar{\mathbf{R}}\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$; equivalently, when $(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q \in [-\frac{1}{2}, \frac{1}{2})$ for all i. By definition of s, we have $\|\mathbf{b}_i^t\bar{\mathbf{R}}\| \le s\|\mathbf{B}\|$. If $\|\mathbf{e}\| < q/(2\|\mathbf{B}\|s)$, then $|(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q| < 1/2$ by Cauchy-Schwarz. Moreover, if $\mathbf{e}$ is chosen at random from $D_{\mathbb{Z}^m,\alpha q}$, then by the fact that $\mathbf{e}$ is 0-subgaussian (Lemma 2.8) with parameter αq, the probability that $|(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q| \ge 1/2$ is negligible, and the second claim follows by the union bound. □

5.4 Gaussian Sampling

Here we show how to use a trapdoor for efficient Gaussian preimage sampling for the function $f_{\mathbf{A}}$, i.e., sampling from a discrete Gaussian over a desired coset of $\Lambda^\perp(\mathbf{A})$. Our precise goal is, given a $\mathbf{G}$-trapdoor $\mathbf{R}$ (with tag $\mathbf{H}$) for matrix $\mathbf{A}$ and a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$, to sample from the spherical discrete Gaussian $D_{\Lambda_\mathbf{u}^\perp(\mathbf{A}),s}$ for relatively small parameter s. As we show next, this task can be reduced, via some efficient pre- and post-processing, to sampling from any sufficiently narrow (not necessarily spherical) Gaussian over the primitive lattice $\Lambda^\perp(\mathbf{G})$.

The main ideas behind our algorithm, which is described formally in Algorithm 3, are as follows. For simplicity, suppose that $\mathbf{R}$ has tag $\mathbf{H} = \mathbf{I}$, so $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$, and suppose we have a subroutine for Gaussian sampling from any desired coset of $\Lambda^\perp(\mathbf{G})$ with some small, fixed parameter $\sqrt{\Sigma_\mathbf{G}} \ge \eta_\epsilon(\Lambda^\perp(\mathbf{G}))$. For example, Section 4 describes algorithms for which $\sqrt{\Sigma_\mathbf{G}}$ is either 2 or $\sqrt{5}$. (Throughout this summary we omit the small rounding factor $r = \omega(\sqrt{\log n})$ from all Gaussian parameters.) The algorithm for sampling from a coset $\Lambda_\mathbf{u}^\perp(\mathbf{A})$ follows from two main observations:

1. If we sample a Gaussian $\mathbf{z}$ with parameter $\sqrt{\Sigma_\mathbf{G}}$ from $\Lambda_\mathbf{u}^\perp(\mathbf{G})$ and produce $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$, then $\mathbf{y}$ is Gaussian over the (non-full-rank) set $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Lambda_\mathbf{u}^\perp(\mathbf{G}) \subsetneq \Lambda_\mathbf{u}^\perp(\mathbf{A})$ with parameter $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\sqrt{\Sigma_\mathbf{G}}$ (i.e., covariance $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Sigma_\mathbf{G}[\,\mathbf{R}^t\ \mathbf{I}\,]$). The (strict) inclusion holds because for any $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ where $\mathbf{z} \in \Lambda_\mathbf{u}^\perp(\mathbf{G})$, we have

$$\mathbf{A}\mathbf{y} = (\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])\mathbf{z} = \mathbf{G}\mathbf{z} = \mathbf{u}.$$

   Note that $s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_\mathbf{G}}) \le s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]) \cdot s_1(\sqrt{\Sigma_\mathbf{G}}) \le \sqrt{s_1(\mathbf{R})^2 + 1} \cdot s_1(\sqrt{\Sigma_\mathbf{G}})$, so $\mathbf{y}$'s distribution is only about an $s_1(\mathbf{R})$ factor wider than that of $\mathbf{z}$ over $\Lambda_\mathbf{u}^\perp(\mathbf{G})$. However, $\mathbf{y}$ lies in a non-full-rank subset of $\Lambda_\mathbf{u}^\perp(\mathbf{A})$, and its distribution is 'skewed' (non-spherical). This leaks information about the trapdoor $\mathbf{R}$, so we cannot just output $\mathbf{y}$.

2. To sample from a *spherical* Gaussian over all of $\Lambda_\mathbf{u}^\perp(\mathbf{A})$, we use the 'convolution' technique from [Pei10] to correct for the above-described problems with the distribution of $\mathbf{y}$. Specifically, we first choose a Gaussian perturbation $\mathbf{p} \in \mathbb{Z}^m$ having covariance $s^2 - \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Sigma_\mathbf{G}\left[\,\mathbf{R}^t\ \mathbf{I}\,\right]$, which is well-defined as long as $s \ge s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_\mathbf{G}})$. We then sample $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ as above for an adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A}\mathbf{p}$, and output $\mathbf{x} = \mathbf{p} + \mathbf{y}$. Now the support of $\mathbf{x}$ is all of $\Lambda_\mathbf{u}^\perp(\mathbf{A})$, and because the covariances of $\mathbf{p}$ and $\mathbf{y}$ are additive (subject to some mild hypotheses), the overall distribution of $\mathbf{x}$ is spherical with Gaussian parameter s that can be as small as $s \approx s_1(\mathbf{R}) \cdot s_1(\sqrt{\Sigma_\mathbf{G}})$.

Quality analysis. Algorithm 3 can sample from a discrete Gaussian with parameter $s \cdot \omega(\sqrt{\log n})$ where s can be as small as $\sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\Sigma_\mathbf{G}) + 2}$. We stress that this is only very slightly larger — a factor of at most $\sqrt{6/4} \le 1.23$ — than the bound $(s_1(\mathbf{R}) + 1) \cdot \|\widetilde{\mathbf{S}}\|$ from Lemma 5.3 on the largest Gram-Schmidt norm of a lattice basis derived from the trapdoor $\mathbf{R}$. (Recall that our constructions from Section 4 give $s_1(\Sigma_\mathbf{G}) = \|\widetilde{\mathbf{S}}\|^2 = 4$ or 5.) In the iterative "randomized nearest-plane" sampling algorithm of [Kle00, GPV08], the Gaussian parameter s is lower-bounded by the largest Gram-Schmidt norm of the orthogonalized input basis (times the same $\omega(\sqrt{\log n})$ factor used in our algorithm). Therefore, the efficiency

and parallelism of Algorithm 3 comes at almost no cost in quality versus slower, iterative algorithms that use high-precision arithmetic. (It seems very likely that the corresponding small loss in security can easily be mitigated with slightly larger parameters, while still yielding a significant net gain in performance.)

Runtime analysis. We now analyze the computational cost of Algorithm 3, with a focus on optimizing the online runtime and parallelism (sometimes at the expense of the offline phase, which we do not attempt to optimize).

The offline phase is dominated by sampling from $D_{\mathbb{Z}^m, r\sqrt{\Sigma}}$ for some fixed (typically non-spherical) covariance matrix $\Sigma > \mathbf{I}$. By [Pei10, Theorem 3.1], this can be accomplished (up to any desired statistical distance) simply by sampling a continuous Gaussian $D_{r\sqrt{\Sigma - \mathbf{I}}}$ with sufficient precision, then independently randomized-rounding each entry of the sampled vector to $\mathbb{Z}$ using Gaussian parameter $r \geq \eta_\epsilon(\mathbb{Z})$.

Naively, the online work is dominated by the computation of $\mathbf{H}^{-1}(\mathbf{u} - \bar{\mathbf{w}})$ and $\mathbf{Rz}$ (plus the call to $\mathcal{O}(\mathbf{v})$, which as described in Section 4 requires only $O(\log^c n)$ work, or one table lookup, by each of n processors in parallel). In general, the first computation takes $O(n^2)$ scalar multiplications and additions in $\mathbb{Z}_q$, while the latter takes $O(\bar{m} \cdot w)$, which is typically $\Theta(n^2 \log^2 q)$. (Obviously, both computations are perfectly parallelizable.) However, the special form of $\mathbf{z}$, and often of $\mathbf{H}$, allow for some further asymptotic and practical optimizations: since $\mathbf{z}$ is typically produced by concatenating n independent dimension-k subvectors that are sampled offline, we can precompute much of $\mathbf{Rz}$ by pre-multiplying each subvector by each of the n blocks of k columns in $\mathbf{R}$. This reduces the online computation of $\mathbf{Rz}$ to the summation of n dimension-$\bar{m}$ vectors, or $O(n^2 \log q)$ scalar additions (and no multiplications) in $\mathbb{Z}_q$. As for multiplication by $\mathbf{H}^{-1}$, in some applications (like GPV signatures) $\mathbf{H}$ is always the identity $\mathbf{I}$, in which case multiplication is unnecessary; in all other applications we know of, $\mathbf{H}$ actually represents multiplication in a certain extension field/ring of $\mathbb{Z}_q$, which can be computed in $O(n \log n)$ scalar operations and depth $O(\log n)$. In conclusion, the asymptotic cost of the online phase is still dominated by computing $\mathbf{Rz}$, which takes $\tilde{O}(n^2)$ work, but the hidden constants are small and many practical speedups are possible.

Theorem 5.5. *Algorithm 3 is correct.*

To prove the theorem we need the following fact about products of Gaussian functions.

Fact 5.6 (Product of degenerate Gaussians)**.** *Let $\Sigma_1, \Sigma_2 \in \mathbb{R}^{m \times m}$ be symmetric positive semidefinite matrices, let $V_i = \mathrm{span}(\Sigma_i)$ for $i = 1, 2$ and $V_3 = V_1 \cap V_2$, let $\mathbf{P} = \mathbf{P}^t \in \mathbb{R}^{m \times m}$ be the symmetric matrix that projects orthogonally onto V_3, and let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^m$ be arbitrary. Supposing it exists, let $\mathbf{v}$ be the unique point in $(V_1 + \mathbf{c}_1) \cap (V_2 + \mathbf{c}_2) \cap V_3^\perp$. Then*

$$\rho_{\sqrt{\Sigma_1}}(\mathbf{x} - \mathbf{c}_1) \cdot \rho_{\sqrt{\Sigma_2}}(\mathbf{x} - \mathbf{c}_2) = \rho_{\sqrt{\Sigma_1 + \Sigma_2}}(\mathbf{c}_1 - \mathbf{c}_2) \cdot \rho_{\sqrt{\Sigma_3}}(\mathbf{x} - \mathbf{c}_3),$$

where Σ_3 and $\mathbf{c}_3 \in \mathbf{v} + V_3$ are such that

$$\Sigma_3^+ = \mathbf{P}(\Sigma_1^+ + \Sigma_2^+)\mathbf{P}$$
$$\Sigma_3^+(\mathbf{c}_3 - \mathbf{v}) = \Sigma_1^+(\mathbf{c}_1 - \mathbf{v}) + \Sigma_2^+(\mathbf{c}_2 - \mathbf{v}).$$

Proof of Theorem 5.5. We adopt the notation from the algorithm, let $V = \mathrm{span}(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]) \subset \mathbb{R}^m$, let $\mathbf{P}$ be the matrix that projects orthogonally onto V, and define the lattice $\Lambda = \mathbb{Z}^m \cap V = \mathcal{L}(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])$, which spans V. We analyze the output distribution of SampleD. Clearly, it always outputs an element of $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$, so let $\bar{\mathbf{x}} \in \Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ be arbitrary. Now SampleD outputs $\bar{\mathbf{x}}$ exactly when it chooses in Step 1 some $\bar{\mathbf{p}} \in V + \bar{\mathbf{x}}$, followed in

Algorithm 3 Efficient algorithm $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R}, \bar{\mathbf{A}}, \mathbf{H}, \mathbf{u}, s)$ for sampling a discrete Gaussian over $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$.

Input: An oracle $\mathcal{O}(\mathbf{v})$ for Gaussian sampling over a desired coset $\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})$ with fixed parameter $r\sqrt{\Sigma_{\mathbf{G}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}))$, for some $\Sigma_{\mathbf{G}} \geq 2$ and $\epsilon \leq 1/2$.

Offline phase:

- partial parity-check matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$;
- trapdoor matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$;
- positive definite $\Sigma \geq \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right](2 + \Sigma_{\mathbf{G}})[\,\mathbf{R}^t \ \mathbf{I}\,]$, e.g., any $\Sigma = s^2 \geq (s_1(\mathbf{R})^2 + 1)(s_1(\Sigma_{\mathbf{G}}) + 2)$.

Online phase:

- invertible tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ defining $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$, for $m = \bar{m} + w$ ($\mathbf{H}$ may instead be provided in the offline phase, if it is known then);
- syndrome $\mathbf{u} \in \mathbb{Z}_q^n$.

Output: A vector $\mathbf{x}$ drawn from a distribution within $O(\epsilon)$ statistical distance of $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}), r \cdot \sqrt{\Sigma}}$.

Offline phase:

1: Choose a fresh perturbation $\mathbf{p} \leftarrow D_{\mathbb{Z}^m, r\sqrt{\Sigma_{\mathbf{p}}}}$, where $\Sigma_{\mathbf{p}} = \Sigma - \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \Sigma_{\mathbf{G}} \,[\,\mathbf{R}^t \ \mathbf{I}\,] \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\,\mathbf{R}^t \ \mathbf{I}\,]$.

2: Let $\mathbf{p} = [\,\mathbf{p}_1\,]$ for $\mathbf{p}_1 \in \mathbb{Z}^{\bar{m}}$, $\mathbf{p}_2 \in \mathbb{Z}^w$, and compute $\bar{\mathbf{w}} = \bar{\mathbf{A}}(\mathbf{p}_1 - \mathbf{R}\mathbf{p}_2) \in \mathbb{Z}_q^n$ and $\mathbf{w} = \mathbf{G}\mathbf{p}_2 \in \mathbb{Z}_q^n$.

Online phase:

3: Let $\mathbf{v} \leftarrow \mathbf{H}^{-1}(\mathbf{u} - \bar{\mathbf{w}}) - \mathbf{w} = \mathbf{H}^{-1}(\mathbf{u} - \mathbf{A}\mathbf{p}) \in \mathbb{Z}_q^n$, and choose $\mathbf{z} \leftarrow D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}), r\sqrt{\Sigma_{\mathbf{G}}}}$ by calling $\mathcal{O}(\mathbf{v})$.

4: **return** $\mathbf{x} \leftarrow \mathbf{p} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$.

Step 3 by the unique $\bar{\mathbf{z}} \in \Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})$ such that $\bar{\mathbf{x}} - \bar{\mathbf{p}} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\bar{\mathbf{z}}$. It is easy to check that $\rho_{\sqrt{\Sigma_{\mathbf{G}}}}(\bar{\mathbf{z}}) = \rho_{\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{x}} - \bar{\mathbf{p}})$, where

$$\Sigma_{\mathbf{y}} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \Sigma_{\mathbf{G}} \,[\,\mathbf{R}^t \ \mathbf{I}\,] \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\,\mathbf{R}^t \ \mathbf{I}\,]$$

is the covariance matrix with $\mathrm{span}(\Sigma_{\mathbf{y}}) = V$. Note that $\Sigma_{\mathbf{p}} + \Sigma_{\mathbf{y}} = \Sigma$ by definition of $\Sigma_{\mathbf{p}}$, and that $\mathrm{span}(\Sigma_{\mathbf{p}}) = \mathbb{R}^m$ because $\Sigma_{\mathbf{p}} > \mathbf{0}$. Therefore, we have (where C denotes a normalizing constant that may vary from line to line, but does not depend on $\bar{\mathbf{x}}$):

$$\begin{aligned}
p_{\bar{\mathbf{x}}} &= \Pr[\mathsf{SampleD} \text{ outputs } \bar{\mathbf{x}}] \\
&= \sum_{\bar{\mathbf{p}} \in \mathbb{Z}^m \cap (V + \bar{\mathbf{x}})} D_{\mathbb{Z}^m, r\sqrt{\Sigma_{\mathbf{p}}}}(\bar{\mathbf{p}}) \cdot D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}), r\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{z}}) && \text{(def. of SampleD)} \\
&= C \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_{\mathbf{p}}}}(\bar{\mathbf{p}}) \cdot \rho_{r\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{p}} - \bar{\mathbf{x}}) / \rho_{r\sqrt{\Sigma_{\mathbf{G}}}}(\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})) && \text{(def. of } D) \\
&= C \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_3}}(\bar{\mathbf{p}} - \mathbf{c}_3) / \rho_{r\sqrt{\Sigma_{\mathbf{G}}}}(\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})) && \text{(Fact 5.6)} \\
&\in C[1, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_3}}(\bar{\mathbf{p}} - \mathbf{c}_3) && \text{(Lemma 2.5 and } r\sqrt{\Sigma_{\mathbf{G}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}))) \\
&= C[1, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \rho_{r\sqrt{\Sigma_3}}(\mathbb{Z}^m \cap (V + \bar{\mathbf{x}}) - \mathbf{c}_3), && (5.1)
\end{aligned}$$

where $\Sigma_3^+ = \mathbf{P}(\Sigma_{\mathbf{p}}^+ + \Sigma_{\mathbf{y}}^+)\mathbf{P}$ and $\mathbf{c}_3 \in \mathbf{v} + V = \bar{\mathbf{x}} + V$, because the component of $\bar{\mathbf{x}}$ orthogonal to V is the unique point $\mathbf{v} \in (V + \bar{\mathbf{x}}) \cap V^{\perp}$. Therefore,

$$\mathbb{Z}^m \cap (V + \bar{\mathbf{x}}) - \mathbf{c}_3 = (\mathbb{Z}^m \cap V) + (\bar{\mathbf{x}} - \mathbf{c}_3) \subset V$$

is a coset of the lattice $\Lambda = \mathcal{L}(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])$. It remains to show that $r\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda)$, so that the rightmost term in (5.1) above is essentially a constant (up to some factor in $[\frac{1-\epsilon}{1+\epsilon}, 1]$) independent of $\bar{\mathbf{x}}$, by Lemma 2.5. Then we can conclude that $p_{\bar{\mathbf{x}}} \in [\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}})$, from which the theorem follows.

To show that $r\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda)$, note that since $\Lambda^* \subset V$, for any covariance Π we have $\rho_{\mathbf{P}\sqrt{\Pi}}(\Lambda^*) = \rho_{\sqrt{\Pi}}(\Lambda^*)$, and so $\mathbf{P}\sqrt{\Pi} \geq \eta_\epsilon(\Lambda)$ if and only if $\sqrt{\Pi} \geq \eta_\epsilon(\Lambda)$. Now because both $\Sigma_{\mathbf{p}}, \Sigma_{\mathbf{y}} \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\left[\mathbf{R}^t \ \mathbf{I}\right]$, we have

$$\Sigma_{\mathbf{p}}^+ + \Sigma_{\mathbf{y}}^+ \leq \left(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\left[\mathbf{R}^t \ \mathbf{I}\right]\right)^+.$$

Because $r\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \geq \eta_\epsilon(\Lambda)$ for $\epsilon = \mathrm{negl}(n)$ by Lemma 2.3, we have $r\sqrt{\Sigma_3} = r\sqrt{(\Sigma_{\mathbf{p}}^+ + \Sigma_{\mathbf{y}}^+)^+} \geq \eta_\epsilon(\Lambda)$, as desired. □

5.5 Trapdoor Delegation

Here we describe very simple and efficient mechanism for securely delegating a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ to a trapdoor for an extension $\mathbf{A}' \in \mathbb{Z}_q^{n\times m'}$ of $\mathbf{A}$. Our method has several advantages over the previous basis delegation algorithm of [CHKP10]: first and most importantly, the size of the delegated trapdoor grows only linearly with the dimension m' of $\Lambda^\perp(\mathbf{A}')$, rather than quadratically. Second, the algorithm is much more efficient, because it does not require testing linear independence of Gaussian samples, nor computing the expensive ToBasis and Hermite normal form operations. Third, the resulting trapdoor $\mathbf{R}$ has a 'nice' Gaussian distribution that is easy to analyze and may be useful in applications. We do note that while the delegation algorithm from [CHKP10] works for *any* extension $\mathbf{A}'$ of $\mathbf{A}$ (including $\mathbf{A}$ itself), ours requires $m' \geq m + w$. Fortunately, this is frequently the case in applications such as HIBE and others that use delegation.

Algorithm 4 Efficient algorithm $\mathsf{DelTrap}^{\mathcal{O}}(\mathbf{A}' = [\mathbf{A} \mid \mathbf{A}_1], \mathbf{H}', s')$ for delegating a trapdoor.

Input: an oracle $\mathcal{O}$ for discrete Gaussian sampling over cosets of $\Lambda = \Lambda^\perp(\mathbf{A})$ with parameter $s' \geq \eta_\epsilon(\Lambda)$.
- parity-check matrix $\mathbf{A}' = [\mathbf{A} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times w}$;
- invertible matrix $\mathbf{H}' \in \mathbb{Z}_q^{n\times n}$;

Output: a trapdoor $\mathbf{R}' \in \mathbb{Z}^{m\times w}$ for $\mathbf{A}'$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$.

1: Using $\mathcal{O}$, sample each column of $\mathbf{R}'$ independently from a discrete Gaussian with parameter s' over the appropriate coset of $\Lambda^\perp(\mathbf{A})$, so that $\mathbf{A}\mathbf{R}' = \mathbf{H}'\mathbf{G} - \mathbf{A}_1$.

Usually, the oracle $\mathcal{O}$ needed by Algorithm 4 would be implemented (up to $\mathrm{negl}(n)$ statistical distance) by Algorithm 3 above, using a trapdoor $\mathbf{R}$ for $\mathbf{A}$ where $s_1(\mathbf{R})$ is sufficiently small relative to s'. The following is immediate from Lemma 2.9 and the fact that the columns of $\mathbf{R}'$ are independent and $\mathrm{negl}(n)$-subgaussian. A relatively tight bound on the hidden constant factor can also be derived from Lemma 2.9.

Lemma 5.7. *For any valid inputs $\mathbf{A}'$ and $\mathbf{H}'$, Algorithm 4 outputs a trapdoor $\mathbf{R}'$ for $\mathbf{A}'$ with tag $\mathbf{H}'$, whose distribution is the same for any valid implementation of $\mathcal{O}$, and $s_1(\mathbf{R}') \leq s' \cdot O(\sqrt{m} + \sqrt{w})$ except with negligible probability.*

6 Applications

The main applications of "strong" trapdoors have included digital signature schemes in both the random-oracle and standard models, encryption secure under chosen-ciphertext attack (CCA), and (hierarchical) identity-based encryption. Here we focus on signature schemes and CCA-secure encryption, where our techniques lead to significant new improvements (beyond what is obtained by plugging in our trapdoor generator as a "black box"). Where appropriate, we also briefly mention the improvements that are possible in the remaining applications.

6.1 Algebraic Background

In our applications we need a special collection of elements from a certain ring $\mathcal{R}$, which induce invertible matrices $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ as required by our trapdoor construction. We construct such a ring using ideas from the literature on secret sharing over groups and modules, e.g., [DF94, Feh98]. Define the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ for some monic degree-n polynomial $f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in \mathbb{Z}[x]$ that is irreducible modulo every prime p dividing q. (Such an $f(x)$ can be constructed by finding monic irreducible degree-n polynomials in $\mathbb{Z}_p[x]$ for each prime p dividing q, and using the Chinese remainder theorem on their coefficients to get $f(x)$.) Recall that $\mathcal{R}$ is a free $\mathbb{Z}_q$-module of rank n, i.e., the elements of $\mathcal{R}$ can be represented as vectors in $\mathbb{Z}_q^n$ relative to the standard basis of monomials $1, x, \ldots, x^{n-1}$. Multiplication by any fixed element of $\mathcal{R}$ then acts as a linear transformation on $\mathbb{Z}_q^n$ according to the rule $x \cdot (a_0, \ldots, a_{n-1})^t = (0, a_0, \ldots, a_{n-2})^t - a_{n-1}(f_0, f_1, \ldots, f_{n-1})^t$, and so can be represented by an (efficiently computable) matrix in $\mathbb{Z}_q^{n \times n}$ relative to the standard basis. In other words, there is an injective ring homomorphism $h\colon \mathcal{R} \to \mathbb{Z}_q^{n \times n}$ that maps any $a \in \mathcal{R}$ to the matrix $\mathbf{H} = h(a)$ representing multiplication by a. In particular, $\mathbf{H}$ is invertible if and only if $a \in \mathcal{R}^*$, the set of units in $\mathcal{R}$. By the Chinese remainder theorem, and because $\mathbb{Z}_p[x]/(f(x))$ is a field by construction of $f(x)$, an element $a \in \mathcal{R}$ is a unit exactly when it is nonzero (as a polynomial residue) modulo every prime p dividing q. We use this fact quite essentially in the constructions that follow.

6.2 Signature Schemes

6.2.1 Definitions

A signature scheme SIG for a message space $\mathcal{M}$ (which may depend on the security parameter n) is a tuple of PPT algorithms as follows:

- $\mathsf{Gen}(1^n)$ outputs a verification key vk and a signing key sk.
- $\mathsf{Sign}(sk, \mu)$, given a signing key sk and a message $\mu \in \mathcal{M}$, outputs a signature $\sigma \in \{0,1\}^*$.
- $\mathsf{Ver}(vk, \mu, \sigma)$, given a verification key vk, a message μ, and a signature σ, either accepts or rejects.

The correctness requirement is: for any $\mu \in \mathcal{M}$, generate $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $\sigma \leftarrow \mathsf{Sign}(sk, \mu)$. Then $\mathsf{Ver}(vk, \mu, \sigma)$ should accept with overwhelming probability (over all the randomness in the experiment).

We recall two standard notions of security for signatures. An intermediate notion is strong unforgeability under *static* chosen-message attack, or su-scma security, is defined as follows: first, the forger $\mathcal{F}$ outputs a list of *distinct* query messages $\mu^{(1)}, \ldots, \mu^{(Q)}$ for some Q. (The distinctness condition simplifies our construction, and does not affect the notion's usefulness.) Next, we generate $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $\sigma^{(i)} \leftarrow \mathsf{Sign}(sk, \mu^{(i)})$ for each $i \in [Q]$, then give vk and each $\sigma^{(i)}$ to $\mathcal{F}$. Finally, $\mathcal{F}$ outputs an attempted forgery (μ^*, σ^*). The forger's advantage $\mathbf{Adv}_{\mathrm{SIG}}^{\text{su-scma}}(\mathcal{F})$ is the probability that $\mathsf{Ver}(vk, \mu^*, \sigma^*)$ accepts and $(\mu^*, \sigma^*) \neq (\mu^{(i)}, \sigma^{(i)})$ for all $i \in [Q]$, taken over all the randomness of the experiment. The

scheme is su-scma-secure if $\mathbf{Adv}^{\text{su-scma}}_{\text{SIG}}(\mathcal{F}) = \text{negl}(n)$ for every nonuniform probabilistic polynomial-time algorithm $\mathcal{F}$.

Another notion, called strong existential unforgeability under *adaptive* chosen-message attack, or su-acma security, is defined similarly, except that $\mathcal{F}$ is first given vk and may adaptively choose the messages $\mu^{(i)}$ to be signed, which need not be distinct.

Using a family of *chameleon* hash functions, there is a generic transformation from eu-scma- to eu-acma-security; see, e.g., [KR00]. Furthermore, the transformation results in an *offline/online* scheme in which the Sign algorithm can be precomputed before the message to be signed is known; see [ST01]. The basic idea is that the signer chameleon hashes the true message, then signs the hash value using the eu-scma-secure scheme (and includes the randomness used in the chameleon hash with the final signature). A suitable type of chameleon hash function has been constructed under a weak hardness-of-SIS assumption; see [CHKP10].

6.2.2 Standard Model Scheme

Here we give a signature scheme that is statically secure in the standard model. The scheme itself is essentially identical (up to the improved and generalized parameters) to the one of [Boy10], which is a lattice analogue of the pairing-based signature of [Wat05]. We give a new proof with an improved security reduction that relies on a weaker assumption. The proof uses a variant of the "prefix technique" [HW09] also used in [CHKP10].

Our scheme involves a number of parameters. For simplicity, we give some exemplary asymptotic bounds here. (Other slight trade-offs among the parameters are possible, and more precise values can be obtained using the more exact bounds from earlier in the paper and the material below.) In what follows, $\omega(\sqrt{\log n})$ represents a fixed function that asymptotically grows faster than $\sqrt{\log n}$.

- $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for large enough $q = \text{poly}(n)$ and $k = \lceil \log q \rceil = O(\log n)$, with the ability to sample from cosets of $\Lambda^\perp(\mathbf{G})$ with Gaussian parameter $O(1) \cdot \omega(\sqrt{\log n}) \geq \eta_\epsilon(\Lambda^\perp(\mathbf{G}))$. (See for example the constructions from Section 4.)
- $\bar{m} = O(nk)$ and $\mathcal{D} = D^{\bar{m} \times nk}_{\mathbb{Z}, \omega(\sqrt{\log n})}$ so that $(\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{R})$ is $\text{negl}(n)$-far from uniform for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, and $m = \bar{m} + 2nk$ is the total dimension of the signatures.
- ℓ is a suitable message length (see below), and $s = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ is a sufficiently large Gaussian parameter.

The legal values of ℓ are influenced by the choice of q and n. Our security proof requires a special collection of units in the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ as constructed in Section 6.1 above. We need a sequence of ℓ units $u_1, \ldots, u_\ell \in \mathcal{R}^*$, not necessarily distinct, such that any nontrivial subset-sum is also a unit, i.e., for any nonempty $S \subseteq [\ell]$, $\sum_{i \in S} u_i \in \mathcal{R}^*$. By the characterization of units in $\mathcal{R}$ described in Section 6.1, letting p be the smallest prime dividing q, we can allow any $\ell \leq (p-1) \cdot n$ by taking $p-1$ copies of each of the monomials $x^i \in \mathcal{R}^*$ for $i = 0, \ldots, n-1$.

The signature scheme has message space $\{0,1\}^\ell$, and is defined as follows.

- $\mathsf{Gen}(1^n)$: choose $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$, choose $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times nk}$ from distribution $\mathcal{D}$, and let $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

  For $i = 0, 1, \ldots, \ell$, choose $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times nk}$. Also choose a syndrome $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

  The public verification key is $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$. The secret signing key is $sk = \mathbf{R}$.

- $\mathsf{Sign}(sk, \mu \in \{0,1\}^\ell)$: let $\mathbf{A}_\mu = \left[\mathbf{A} \mid \mathbf{A}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{A}_i\right] \in \mathbb{Z}_q^{n \times m}$, where $\mu_i \in \{0,1\}$ is the ith bit of μ, interpreted as an integer. Output $\mathbf{v} \in \mathbb{Z}^m$ sampled from $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu), s}$, using SampleD with trapdoor $\mathbf{R}$ for $\mathbf{A}$ (which is also a trapdoor for its extension $\mathbf{A}_\mu$).

- $\mathsf{Ver}(vk, \mu, \mathbf{v})$: let $\mathbf{A}_\mu$ be as above. Accept if $\|\mathbf{v}\| \le s \cdot \sqrt{m}$ and $\mathbf{A}_\mu \cdot \mathbf{v} = \mathbf{u}$; otherwise, reject.

Notice that the signing process takes $O(\ell n^2 k)$ scalar operations (to add up the $\mathbf{A}_i$s), but after transforming the scheme to a fully secure one using chameleon hashing, these computations can be performed offline before the message is known.

Theorem 6.1. *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking the $\mathsf{SIS}_{q,\beta}$ problem for large enough $\beta = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$ such that, for any adversary $\mathcal{F}$ mounting an su-scma attack on SIG and making at most Q queries,*

$$\mathbf{Adv}_{\mathsf{SIS}_{q,\beta}}(\mathcal{S}^{\mathcal{F}}) \ge \mathbf{Adv}^{\textit{su-scma}}_{\textit{SIG}}(\mathcal{F})/(2(\ell-1)Q+2) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{F}$ be an adversary mounting an su-scma attack on SIG, having advantage $\delta = \mathbf{Adv}^{\text{su-scma}}_{\text{SIG}}(\mathcal{F})$. We construct a reduction $\mathcal{S}$ attacking $\mathsf{SIS}_{q,\beta}$. The reduction $\mathcal{S}$ takes as input $\bar{m} + nk + 1$ uniformly random and independent samples from $\mathbb{Z}_q^n$, parsing them as a matrix $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{B}] \in \mathbb{Z}_q^{n \times (\bar{m}+nk)}$ and syndrome $\mathbf{u}' \in \mathbb{Z}_q^n$. It will use $\mathcal{F}$ either to find some $\mathbf{z} \in \mathbb{Z}^m$ of length $\|\mathbf{z}\| \le \beta - 1$ such that $\mathbf{A}\mathbf{z} = \mathbf{u}'$ (from which it follows that $[\mathbf{A} \mid \mathbf{u}'] \cdot \mathbf{z}' = \mathbf{0}$, where $\mathbf{z}' = \left[\begin{smallmatrix}\mathbf{z}\\ -1\end{smallmatrix}\right]$ is nonzero and of length at most β), or a nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0}$ (from which is follows that $[\mathbf{A} \mid \mathbf{u}'] \cdot \left[\begin{smallmatrix}\mathbf{z}\\ 0\end{smallmatrix}\right] = \mathbf{0}$).

We distinguish between two types of forger $\mathcal{F}$: one that produces a forgery on an unqueried message (a violation of standard existential unforgeability), and one that produces a new signature on a queried message (a violation of strong unforgeability). Clearly any $\mathcal{F}$ with advantage δ has probability at least $\delta/2$ of succeeding in at least one of these two tasks.

First we consider $\mathcal{F}$ that forges on an unqueried message (with probability at least $\delta/2$). Our reduction $\mathcal{S}$ simulates the static chosen-message attack to $\mathcal{F}$ as follows:

- Invoke $\mathcal{F}$ to receive up to Q messages $\mu^{(1)}, \mu^{(2)}, \ldots \in \{0,1\}^\ell$. Compute the set P of all strings $p \in \{0,1\}^{\le \ell}$ having the property that p is a shortest string for which no $\mu^{(j)}$ has p as a prefix. Equivalently, P represents the set of maximal subtrees of $\{0,1\}^{\le \ell}$ (viewed as a tree) that do not contain any of the queried messages. The set P has size at most $(\ell-1) \cdot Q + 1$, and may be computed efficiently. (See, e.g., [CHKP10] for a precise description of an algorithm.) Choose some p from P uniformly at random, letting $t = |p| \le \ell$.
- Construct a verification key $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u} = \mathbf{u}')$: for $i = 0, \ldots, \ell$, choose $\mathbf{R}_i \leftarrow \mathcal{D}$, and let

$$\mathbf{A}_i = \mathbf{H}_i\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}_i, \text{ where } \mathbf{H}_i = \begin{cases} h(0) = \mathbf{0} & i > t \\ (-1)^{p_i} \cdot h(u_i) & i \in [t] \\ -\sum_{j \in [t]} p_j \cdot \mathbf{H}_j & i = 0 \end{cases}.$$

  (Recall that $u_1, \ldots, u_\ell \in \mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ are units whose nontrivial subset-sums are also units.)

  Note that by hypothesis on $\bar{m}$ and $\mathcal{D}$, for any choice of p the key vk is only $\mathrm{negl}(n)$-far from uniform in statistical distance. Note also that by our choice of the $\mathbf{H}_i$, for any message $\mu \in \{0,1\}^\ell$ having p as a prefix, we have $\mathbf{H}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{H}_i = \mathbf{0}$. Whereas for any $\mu \in \{0,1\}^\ell$ having $p' \ne p$ as its t-bit prefix, we have

$$\mathbf{H}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{H}_i = \sum_{i \in [t]} (p'_i - p_i) \cdot \mathbf{H}_i = \sum_{i \in [t], p'_i \ne p_i} (-1)^{p_i} \cdot \mathbf{H}_i = h\Big(\sum_{i \in [t], p'_i \ne p_i} u_i\Big),$$

  which is invertible by hypothesis on the u_is. Finally, observe that with overwhelming probability over any fixed choice of vk and the $\mathbf{H}_i$, each column of each $\mathbf{R}_i$ is still independently distributed as

a discrete Gaussian with parameter $\omega(\sqrt{\log n}) \geq \eta_\epsilon(\bar{\mathbf{A}})$ over some fixed coset of $\Lambda^\perp(\bar{\mathbf{A}})$, for some negligible $\epsilon = \epsilon(n)$.

- Generate signatures for the queried messages: for each message $\mu = \mu^{(i)}$, compute

$$\mathbf{A}_\mu = \big[\mathbf{A} \mid \mathbf{A}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{A}_i\big] = \big[\bar{\mathbf{A}} \mid \mathbf{B} \mid \mathbf{HG} - \bar{\mathbf{A}}\big(\mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{R}_i\big)\big],$$

  where $\mathbf{H}$ is invertible because the t-bit prefix of μ is not p. Therefore, $\mathbf{R} = (\mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{R}_i)$ is a trapdoor for $\mathbf{A}_\mu$. By the conditional distribution on the $\mathbf{R}_i$s, concatenation of subgaussian random variables, and Lemma 2.9, we have

$$s_1(\mathbf{R}) = \sqrt{\ell+1} \cdot O(\sqrt{\bar{m}} + \sqrt{nk}) \cdot \omega(\sqrt{\log n}) = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})$$

  with overwhelming probability. Since $s = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ is sufficiently large, we can generate a properly distributed signature $\mathbf{v}_\mu \leftarrow D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu),s}$ using $\mathsf{SampleD}$ with trapdoor $\mathbf{R}$.

Next, $\mathcal{S}$ gives vk and the generated signatures to $\mathcal{F}$. Because vk and the signatures are distributed within $\mathrm{negl}(n)$ statistical distance of those in the real attack (for any choice of the prefix p), with probability at least $\delta/2 - \mathrm{negl}(n)$, $\mathcal{F}$ outputs a forgery $(\mu^*, \mathbf{v}^*)$ where μ^* is different from all the queried messages, $\mathbf{A}_{\mu^*}\mathbf{v}^* = \mathbf{u}$, and $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m}$. Furthermore, conditioned on this event, μ^* has p as a prefix with probability at least $1/((\ell-1)Q+1) - \mathrm{negl}(n)$, because p is still essentially uniform in P conditioned on the view of $\mathcal{F}$. Therefore, all of these events occur with probability at least $\delta/(2(\ell-1)Q+2) - \mathrm{negl}(n)$.

In such a case, $\mathcal{S}$ extracts a solution to its SIS challenge instance from the forgery $(\mu^*, \mathbf{v}^*)$ as follows. Because μ^* starts with p, we have $\mathbf{A}_{\mu^*} = \big[\bar{\mathbf{A}} \mid \mathbf{B} \mid -\bar{\mathbf{A}}\mathbf{R}^*\big]$ for $\mathbf{R}^* = \mathbf{R}_0 + \sum_{i\in[\ell]} \mu^*_i \mathbf{R}_i$, and so

$$\underbrace{[\bar{\mathbf{A}} \mid \mathbf{B}]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{I}_{\bar{m}} & & -\mathbf{R}^* \\ & \mathbf{I}_{nk} & \end{bmatrix} \mathbf{v}^*}_{\mathbf{z}} = \mathbf{u} \bmod q,$$

as desired. Because $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m} = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ and $s_1(\mathbf{R}^*) = \sqrt{\ell+1} \cdot O(\sqrt{\bar{m}} + \sqrt{nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability (conditioned on the view of $\mathcal{F}$ and any fixed $\mathbf{H}_i$), we have $\|\mathbf{z}\| = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$, which is at most $\beta - 1$, as desired.

Now we consider an $\mathcal{F}$ that forges on one of its queried messages (with probability at least $\delta/2$). Our reduction $\mathcal{S}$ simulates the attack to $\mathcal{F}$ as follows:

- Invoke $\mathcal{F}$ to receive up to Q *distinct* messages $\mu^{(1)}, \mu^{(2)}, \ldots \in \{0,1\}^\ell$. Choose one of these messages $\mu = \mu^{(i)}$ uniformly at random, "guessing" that the eventual forgery will be on μ.
- Construct a verification key $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$: generate $\mathbf{A}_i$ exactly as above, using $p = \mu$. Then choose $\mathbf{v} \leftarrow D_{\mathbb{Z}^m,s}$ and let $\mathbf{u} = \mathbf{A}_\mu \mathbf{v}$, where $\mathbf{A}_\mu$ is defined in the usual way.
- Generate signatures for the queried messages: for all the queries except μ, proceed exactly as above (which is possible because all the queries are distinct and hence do not have $p = \mu$ as a prefix). For μ, use $\mathbf{v}$ as the signature, which has the required distribution $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu),s}$ by construction.

When $\mathcal{S}$ gives vk and the signatures to $\mathcal{F}$, with probability at least $\delta/2 - \mathrm{negl}(n)$ the forger must output a forgery $(\mu^*, \mathbf{v}^*)$ where μ^* is one of its queries, $\mathbf{v}^*$ is different from the corresponding signature it received, $\mathbf{A}_{\mu^*}\mathbf{v}^* = \mathbf{u}$, and $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m}$. Because vk and the signatures are appropriately distributed for any

choice μ that $\mathcal{S}$ made, conditioned on the above event the probability that $\mu^* = \mu$ is at least $1/Q - \mathrm{negl}(n)$. Therefore, all of these events occur with probability at least $\delta/(2Q) - \mathrm{negl}(n)$.

In such a case, $\mathcal{S}$ extracts a solution to its SIS challenge from the forgery as follows. Because $\mu^* = \mu$, we have $\mathbf{A}_{\mu^*} = \left[\bar{\mathbf{A}} \mid \mathbf{B} \mid -\bar{\mathbf{A}}\mathbf{R}^*\right]$ for $\mathbf{R}^* = \mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i^* \mathbf{R}_i$, and so

$$\underbrace{[\bar{\mathbf{A}} \mid \mathbf{B}]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{I}_{\bar{m}} & & -\mathbf{R}^* \\ & \mathbf{I}_{nk} & \end{bmatrix} (\mathbf{v}^* - \mathbf{v})}_{\mathbf{z}} = \mathbf{0} \bmod q.$$

Because both $\|\mathbf{v}^*\|, \|\mathbf{v}\| \le s \cdot \sqrt{m} = O(\sqrt{\ell}nk) \cdot \omega(\sqrt{\log n})^2$ and $s_1(\mathbf{R}^*) = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability (conditioned on the view of $\mathcal{F}$ and any fixed $\mathbf{H}_i$), we have $\|\mathbf{z}\| = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$ with overwhelming probability, as needed. It just remains to show that $\mathbf{z} \neq \mathbf{0}$ with overwhelming probability. To see this, write $\mathbf{w} = \mathbf{v}^* - \mathbf{v} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \in \mathbb{Z}^{\bar{m}} \times \mathbb{Z}^{nk} \times \mathbb{Z}^{nk}$, with $\mathbf{w} \neq \mathbf{0}$. If $\mathbf{w}_2 \neq \mathbf{0}$ or $\mathbf{w}_3 = \mathbf{0}$, then $\mathbf{z} \neq \mathbf{0}$ and we are done. Otherwise, choose some entry of $\mathbf{w}_3$ that is nonzero; without loss of generality say it is w_m. Let $\mathbf{r} = (\mathbf{R}_0)_{nk}$. Now for any fixed values of $\mathbf{R}_i$ for $i \in [\ell]$ and fixed first $nk - 1$ columns of $\mathbf{R}_0$, we have $\mathbf{z} = \mathbf{0}$ only if $\mathbf{r} \cdot w_m = \mathbf{y} \in \mathbb{R}^{\bar{m}}$ for some fixed $\mathbf{y}$. Conditioned on the adversary's view (specifically, $(\mathbf{A}_0)_{nk} = \bar{\mathbf{A}}\mathbf{r}$), $\mathbf{r}$ is distributed as a discrete Gaussian of parameter $\ge 2\eta_\epsilon(\Lambda^\perp(\bar{\mathbf{A}}))$ for some $\epsilon = \mathrm{negl}(n)$ over a coset of $\Lambda^\perp(\bar{\mathbf{A}})$. Then by Lemma 2.7, we have $\mathbf{r} = \mathbf{y}/w_m$ with only $2^{-\Omega(n)}$ probability, and we are done. □

6.3 Chosen Ciphertext-Secure Encryption

Definitions. A public-key cryptosystem for a message space $\mathcal{M}$ (which may depend on the security parameter) is a tuple of algorithms as follows:

- $\mathsf{Gen}(1^n)$ outputs a public encryption key pk and a secret decryption key sk.
- $\mathsf{Enc}(pk, m)$, given a public key pk and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \{0,1\}^*$.
- $\mathsf{Dec}(sk, c)$, given a decryption key sk and a ciphertext c, outputs some $m \in \mathcal{M} \cup \{\perp\}$.

The correctness requirement is: for any $m \in \mathcal{M}$, generate $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $c \leftarrow \mathsf{Enc}(pk, m)$. Then $\mathsf{Dec}(sk, c)$ should output m with overwhelming probability (over all the randomness in the experiment).

We recall the two notions of security under chosen-ciphertext attacks. We start with the weaker notion of CCA1 (or "lunchtime") security. Let $\mathcal{A}$ be any nonuniform probabilistic polynomial-time algorithm. First, we generate $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and give pk to $\mathcal{A}$. Next, we give $\mathcal{A}$ oracle access to the decryption procedure $\mathsf{Dec}(sk, \cdot)$. Next, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$ and is given a challenge ciphertext $c \leftarrow \mathsf{Enc}(pk, m_b)$ for either $b = 0$ or $b = 1$. The scheme is CCA1-secure if the views of $\mathcal{A}$ (i.e., the public key pk, the answers to its oracle queries, and the ciphertext c) for $b = 0$ versus $b = 1$ are computationally indistinguishable (i.e., $\mathcal{A}$'s acceptance probabilities for $b = 0$ versus $b = 1$ differ by only $\mathrm{negl}(n)$). In the stronger CCA2 notion, after receiving the challenge ciphertext, $\mathcal{A}$ continues to have access to the decryption oracle $\mathsf{Dec}(sk, \cdot)$ for any query not equal to the challenge ciphertext c; security it defined similarly.

Construction. To highlight the main new ideas, here we present a public-key encryption scheme that is CCA1-secure. Full CCA2 security can be obtained via relatively generic transformations using either strongly unforgeable one-time signatures [DDN00], or a message authentication code and weak form of commitment [BCHK07]; we omit these details.

Our scheme involves a number of parameters, for which we give some exemplary asymptotic bounds. In what follows, $\omega(\sqrt{\log n})$ represents a fixed function that asymptotically grows faster than $\sqrt{\log n}$.

- $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for large enough prime power $q = p^e = \mathrm{poly}(n)$ and $k = O(\log q) = O(\log n)$. We require an oracle $\mathcal{O}$ that solves LWE with respect to $\Lambda(\mathbf{G}^t)$ for any error vector in some $\mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ where $\|\mathbf{B}\| = O(1)$. (See for example the constructions from Section 4.)
- $\bar{m} = O(nk)$ and $\mathcal{D} = D_{\mathbb{Z},\omega(\sqrt{\log n})}^{\bar{m} \times nk}$ so that $(\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{R})$ is $\mathrm{negl}(n)$-far from uniform for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, and $m = \bar{m} + nk$ is the total dimension of the public key and ciphertext.
- α is an error rate for LWE, for sufficiently large $1/\alpha = O(nk) \cdot \omega(\sqrt{\log n})$.

Our scheme requires a special collection of elements in the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ as constructed in Section 6.1 (recall that here $q = p^e$). We need a very large set $\mathcal{U} = \{u_1, \ldots, u_\ell\} \subset \mathcal{R}$ with the "unit differences" property: for any $i \neq j$, the difference $u_i - u_j \in \mathcal{R}^*$, and hence $h(u_i - u_j) = h(u_i) - h(u_j) \in \mathbb{Z}_q^{n \times n}$ is invertible. (Note that the u_is need not all be units themselves.) Concretely, by the characterization of units in $\mathcal{R}$ given above, we take $\mathcal{U}$ to be all linear combinations of the monomials $1, x, \ldots, x^{n-1}$ with coefficients in $\{0, \ldots, p-1\}$, of which there are exactly p^n. Since the difference between any two such distinct elements is nonzero modulo p, it is a unit.

The system has message space $\{0,1\}^{nk}$, which we map bijectively to the cosets of $\Lambda/2\Lambda$ for $\Lambda = \Lambda(\mathbf{G}^t)$ via some function encode that is efficient to evaluate and invert. Concretely, letting $\mathbf{S} \in \mathbb{Z}^{nk \times nk}$ be any basis of Λ, we can map $\mathbf{m} \in \{0,1\}^{nk}$ to $\mathsf{encode}(\mathbf{m}) = \mathbf{S}\mathbf{m} \in \mathbb{Z}^{nk}$.

- $\mathsf{Gen}(1^n)$: choose $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, letting $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R} \bmod q$. The public key is $pk = \mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n \times m}$ and the secret key is $sk = \mathbf{R}$.
- $\mathsf{Enc}(pk = [\bar{\mathbf{A}} \mid \mathbf{A}_1], \mathbf{m} \in \{0,1\}^{nk})$: choose nonzero $u \leftarrow \mathcal{U}$ and let $\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}]$. Choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\bar{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$, and $\mathbf{e}_1 \leftarrow D_{\mathbb{Z},s}^{nk}$ where $s^2 = (\|\bar{\mathbf{e}}\|^2 + \bar{m}(\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$.

  Let
  $$\mathbf{b}^t = 2(\mathbf{s}^t \mathbf{A}_u \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod 2q,$$
  where $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1) \in \mathbb{Z}^m$ and $\mathbf{0}$ has dimension $\bar{m}$. (Note the use of mod-$2q$ arithmetic: $2(\mathbf{s}^t \mathbf{A}_u \bmod q)$ is an element of the lattice $2\Lambda(\mathbf{A}_u^t) \supseteq 2q\mathbb{Z}^m$.) Output the ciphertext $c = (u, \mathbf{b}) \in \mathcal{U} \times \mathbb{Z}_{2q}^m$.
- $\mathsf{Dec}(sk = \mathbf{R}, c = (u, \mathbf{b}) \in \mathcal{U} \times \mathbb{Z}_{2q}^m)$: Let $\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}] = [\bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

  1. If c does not parse or $u = 0$, output $\perp$. Otherwise, call $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}_u, \mathbf{b} \bmod q)$ to get values $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1) \in \mathbb{Z}^{\bar{m}} \times \mathbb{Z}^{nk}$ for which $\mathbf{b}^t = \mathbf{z}^t \mathbf{A}_u + \mathbf{e}^t \bmod q$. (Note that $h(u) \in \mathbb{Z}_q^{n \times n}$ is invertible, as required by Invert.) If the call to Invert fails for any reason, output $\perp$.
  2. If $\|\bar{\mathbf{e}}\| \geq \alpha q \sqrt{\bar{m}}$ or $\|\mathbf{e}_1\| \geq \alpha q \sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$, output $\perp$.
  3. Let $\mathbf{v} = \mathbf{b} - \mathbf{e} \bmod 2q$, parsed as $\mathbf{v} = (\bar{\mathbf{v}}, \mathbf{v}_1) \in \mathbb{Z}_{2q}^{\bar{m}} \times \mathbb{Z}_{2q}^{nk}$. If $\bar{\mathbf{v}} \notin 2\Lambda(\bar{\mathbf{A}}^t)$, output $\perp$. Finally, output $\mathsf{encode}^{-1}(\mathbf{v}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \bmod 2q) \in \{0,1\}^{nk}$ if it exists, otherwise output $\perp$.

  (In practice, to avoid timing attacks one would perform all of the Dec operations first, and only then finally output $\perp$ if any of the validity tests failed.)

Lemma 6.2. *The above scheme has only $2^{-\Omega(n)}$ probability of decryption error.*

The error probability can be made zero by changing Gen and Enc so that they resample $\mathbf{R}$, $\bar{\mathbf{e}}$, and/or $\mathbf{e}_1$ in the rare event that they violate the corresponding bounds given in the proof below.

Proof. Let $(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{Gen}(1^n)$. By Lemma 2.9, we have $s_1(\mathbf{R}) \le O(\sqrt{nk}) \cdot \omega(\sqrt{\log n})$ except with probability $2^{-\Omega(n)}$. Now consider the random choices made by $\mathsf{Enc}(\mathbf{A}, \mathbf{m})$ for arbitrary $\mathbf{m} \in \{0,1\}^{nk}$. By Lemma 2.6, we have both $\|\bar{\mathbf{e}}\| < \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_1\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$, except with probability $2^{-\Omega(n)}$. Letting $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1)$, we have

$$\left\|\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\right\| \le \|\bar{\mathbf{e}}^t\mathbf{R}\| + \|\mathbf{e}_1\| < \alpha q \cdot O(nk) \cdot \omega(\sqrt{\log n}).$$

In particular, for large enough $1/\alpha = O(nk) \cdot \omega(\sqrt{\log n})$ we have $\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$. Therefore, the call to Invert made by $\mathsf{Dec}(\mathbf{R}, (u, \mathbf{b}))$ returns $\mathbf{e}$. It follows that for $\mathbf{v} = (\bar{\mathbf{v}}, \mathbf{v}_1) = \mathbf{b} - \mathbf{e} \bmod 2q$, we have $\bar{\mathbf{v}} \in 2\Lambda(\bar{\mathbf{A}}^t)$ as needed. Finally,

$$\mathbf{v}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = 2(\mathbf{s}^t h(u)\mathbf{G} \bmod q) + \mathsf{encode}(\mathbf{m}) \bmod 2q,$$

which is in the coset $\mathsf{encode}(\mathbf{m}) \in \Lambda(\mathbf{G}^t)/2\Lambda(\mathbf{G}^t)$, and so Dec outputs $\mathbf{m}$ as desired. □

Theorem 6.3. *The above scheme is CCA1-secure assuming the hardness of decision-*$\mathsf{LWE}_{q,\alpha'}$ *for* $\alpha' = \alpha/3 \ge 2\sqrt{n}/q$.

Proof. We start by giving a particular form of discretized LWE that we will need below. Given access to an LWE distribution $A_{\mathbf{s},\alpha'}$ over $\mathbb{Z}_q^n \times \mathbb{T}$ for any $\mathbf{s} \in \mathbb{Z}_q^n$ (where recall that $\mathbb{T} = \mathbb{R}/\mathbb{Z}$), by [Pei10, Theorem 3.1] we can transform its samples $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a}\rangle/q + e \bmod 1)$ to have the form $(\mathbf{a}, 2(\langle \mathbf{s}, \mathbf{a}\rangle \bmod q) + \mathbf{e}' \bmod 2q)$ for $e' \leftarrow D_{\mathbb{Z},\alpha q}$, by mapping $b \mapsto 2qb + D_{\mathbb{Z}-2qb,s} \bmod 2q$ where $s^2 = (\alpha q)^2 - (2\alpha' q)^2 \ge 4n \ge \eta_\epsilon(\mathbb{Z})^2$. This transformation maps the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_{2q}$, so the discretized distribution is pseudorandom under the hypothesis of the theorem.

We proceed via a sequence of hybrid games. The game H_0 is exactly the CCA1 attack with the system described above.

In game H_1, we change how the public key $\mathbf{A}$ and challenge ciphertext $c^* = (u^*, \mathbf{b}^*)$ are constructed, and the way that decryption queries are answered (slightly), but in a way that introduces only $\mathrm{negl}(n)$ statistical difference with H_0. At the start of the experiment we choose nonzero $u^* \leftarrow \mathcal{U}$ and let the public key be $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] = [\bar{\mathbf{A}} \mid -h(u^*)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\bar{\mathbf{A}}$ and $\mathbf{R}$ are chosen in the same way as in H_0. (In particular, we still have $s_1(\mathbf{R}) \le O(\sqrt{nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability.) Note that $\mathbf{A}$ is still $\mathrm{negl}(n)$-uniform for any choice of u^*, so conditioned on any fixed choice of $\mathbf{A}$, the value of u^* is statistically hidden from the attacker. To aid with decryption queries, we also choose an arbitrary (not necessarily short) $\hat{\mathbf{R}} \in \mathbb{Z}^{\bar{m} \times nk}$ such that $\mathbf{A}_1 = -\bar{\mathbf{A}}\hat{\mathbf{R}} \bmod q$.

To answer a decryption query on a ciphertext $(u, \mathbf{b})$, we use an algorithm very similar to Dec with trapdoor $\mathbf{R}$. After testing whether $u = 0$ (and outputting $\perp$ if so), we call $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}_u, \mathbf{b} \bmod q)$ to get some $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}^m$, where

$$\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}] = [\bar{\mathbf{A}} \mid h(u - u^*)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

(If Invert fails, we output $\perp$.) We then perform steps 2 and 3 on $\mathbf{e} \in \mathbb{Z}^m$ and $\mathbf{v} = \mathbf{b} - \mathbf{e} \bmod 2q$ exactly as in Dec, except that we use $\hat{\mathbf{R}}$ in place of $\mathbf{R}$ when decoding the message in step 3.

We now analyze the behavior of this decryption routine. Whenever $u \ne u^*$, which is the case with overwhelming probability because u^* is statistically hidden, by the "unit differences" property on $\mathcal{U}$ we have that $h(u - u^*) \in \mathbb{Z}_q^{n \times n}$ is invertible, as required by the call to Invert. Now, either there exists an $\mathbf{e}$ that satisfies the validity tests in step 2 and such that $\mathbf{b}^t = \mathbf{z}^t\mathbf{A}_u + \mathbf{e}^t \bmod q$ for some $\mathbf{z} \in \mathbb{Z}_q^n$, or there does not. In the latter case, no matter what Invert does in H_0 and H_1, step 2 will return $\perp$ in both games. Now consider the former case: by the constraints on $\mathbf{e}$, we have $\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ in both games, so the call to Invert

must return this $\mathbf{e}$ (but possibly different $\mathbf{z}$) in both games. Finally, the result of decryption is the same in both games: if $\bar{\mathbf{v}} \in 2\Lambda(\bar{\mathbf{A}}^t)$ (otherwise, both games return $\perp$), then we can express $\mathbf{v}$ as

$$\mathbf{v}^t = 2(\mathbf{s}^t \mathbf{A}_u \bmod q) + (\mathbf{0}, \mathbf{v}')^t \bmod 2q$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{v}' \in \mathbb{Z}_{2q}^{nk}$. Then for *any* solution $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times nk}$ to $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R} \bmod q$, we have

$$\mathbf{v}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = 2(\mathbf{s}^t h(u)\mathbf{G} \bmod q) + (\mathbf{v}')^t \bmod 2q.$$

In particular, this holds for the $\mathbf{R}$ in H_0 and the $\hat{\mathbf{R}}$ in H_1 that are used for decryption. It follows that both games output $\mathsf{encode}^{-1}(\mathbf{v}')$, if it exists (and $\perp$ otherwise).

Finally, in H_1 we produce the challenge ciphertext $(u, \mathbf{b})$ on a message $\mathbf{m} \in \{0,1\}^{nk}$ as follows. Let $u = u^*$, and choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\bar{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$ as usual, but do not choose $\mathbf{e}_1$. Note that $\mathbf{A}_u = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $\bar{\mathbf{b}}^t = 2(\mathbf{s}^t \bar{\mathbf{A}} \bmod q) + \bar{\mathbf{e}}^t \bmod 2q$. Let

$$\mathbf{b}_1^t = -\bar{\mathbf{b}}^t \mathbf{R} + \hat{\mathbf{e}}^t + \mathsf{encode}(\mathbf{m}) \bmod 2q,$$

where $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q \sqrt{m} \cdot \omega(\sqrt{\log n})}^{nk}$, and output $(u, \mathbf{b} = (\bar{\mathbf{b}}, \mathbf{b}_1))$. We now show that the distribution of $(u, \mathbf{b})$ is within $\mathrm{negl}(n)$ statistical distance of that in H_0, given the attacker's view (i.e., pk and the results of the decryption queries). Clearly, u and $\bar{\mathbf{b}}$ have essentially the same distribution as in H_0, because u is $\mathrm{negl}(n)$-uniform given pk, and by construction of $\bar{\mathbf{b}}$. By substitution, we have

$$\mathbf{b}_1^t = 2(\mathbf{s}^t(-\bar{\mathbf{A}}\mathbf{R}) \bmod q) + (\bar{\mathbf{e}}^t \mathbf{R} + \hat{\mathbf{e}}^t) + \mathsf{encode}(m).$$

Therefore, it suffices to show that for fixed $\bar{\mathbf{e}}$, each $\langle \bar{\mathbf{e}}, \mathbf{r}_i \rangle + \hat{e}_i$ has distribution $\mathrm{negl}(n)$-far from $D_{\mathbb{Z},s}$, where $s^2 = (\|\bar{\mathbf{e}}\|^2 + m(\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$, over the random choice of $\mathbf{r}_i$ (conditioned on the value of $\bar{\mathbf{A}}\mathbf{r}_i$ from the public key) and of $\hat{e}_i$. Because each $\mathbf{r}_i$ is an independent discrete Gaussian over a coset of $\Lambda^{\perp}(\bar{\mathbf{A}})$, the claim follows essentially by [Reg05, Corollary 3.10], but adapted to discrete random variables using [Pei10, Theorem 3.1] in place of [Reg05, Claim 3.9].

In game H_2, we only change how the $\bar{\mathbf{b}}$ component of the challenge ciphertext is created, letting it be uniformly random in $\mathbb{Z}_{2q}^{\bar{m}}$. We construct pk, answer decryption queries, and construct $\mathbf{b}_1$ in exactly the same way as in H_1. First observe that under our (discretized) LWE hardness assumption, games H_1 and H_2 are computationally indistinguishable by an elementary reduction: given $(\bar{\mathbf{A}}, \bar{\mathbf{b}}) \in \mathbb{Z}_q^{n \times \bar{m}} \times \mathbb{Z}_{2q}^{\bar{m}}$ where $\bar{\mathbf{A}}$ is uniformly random and either $\bar{\mathbf{b}}^t = 2(\mathbf{s}^t \bar{\mathbf{A}} \bmod q) + \mathbf{e}^t \bmod 2q$ (for $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$) or $\bar{\mathbf{b}}$ is uniformly random, we can efficiently emulate either game H_1 or H_2 (respectively) by doing everything exactly as in the two games, except using the given $\bar{\mathbf{A}}$ and $\bar{\mathbf{b}}$ when constructing the public key and challenge ciphertext.

Now by the leftover hash lemma, $(\bar{\mathbf{A}}, \bar{\mathbf{b}}^t, \bar{\mathbf{A}}\mathbf{R}, -\bar{\mathbf{b}}^t\mathbf{R})$ is $\mathrm{negl}(n)$-uniform when $\mathbf{R}$ is chosen as in H_2. Therefore, the challenge ciphertext has the same distribution (up to $\mathrm{negl}(n)$ statistical distance) for any encrypted message, and so the adversary's advantage is negligible. This completes the proof. □

References

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115. 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9. 1999.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, April 2011. Preliminary version in STACS 2009.

[Bab85] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS 1985.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BCHK07] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BFKL93] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO*, pages 278–291. 1993.

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/`.

[Boy10] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517. 2010.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*. 2011. To appear.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010.

[CN11] Y. Chen and P. Q. Nguyen. BKZ 2.0: Simulation and better lattice security estimates. In *ASIACRYPT*. 2011. To appear.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DF94] Y. Desmedt and Y. Frankel. Perfect homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM J. Discrete Math.*, 7(4):667–679, 1994.

[Feh98] S. Fehr. *Span Programs over Rings and How to Share a Secret from a Module*. Master's thesis, ETH Zurich, Institute for Theoretical Computer Science, 1998.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131. 1997.

[GH11] C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS*. 2011. To appear.

[GHV10] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522. 2010.

[GKV10] S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT*, pages 395–412. 2010.

[GN08] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51. 2008.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[HW09] S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO*, pages 654–670. 2009.

[Kle00] P. N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. 2000.

[KMO10] E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT*, pages 673–692. 2010.

[KR00] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*. 2000.

[LM06] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155. 2006.

[LM08] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54. 2008.

[LM09] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, pages 577–594. 2009.

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72. 2008.

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339. 2011.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23. 2010.

[Lyu08] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, pages 162–179. 2008.

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[Mic02] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

[MM11] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484. 2011.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[MR09] D. Micciancio and O. Regev. Lattice-based cryptography. In *Post Quantum Cryptography*, pages 147–191. Springer, February 2009.

[Pei09a] C. Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359, July 2009. `http://eprint.iacr.org/`.

[Pei09b] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[PR06] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166. 2006.

[PV08] C. Peikert and V. Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553. 2008.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571. 2008.

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196. 2008.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[RS10] M. Rückert and M. Schneider. Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.

[Rüc10] M. Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In *PQCrypto*, pages 182–200. 2010.

[ST01] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *CRYPTO*, pages 355–367. 2001.

[Ver11] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, January 2011. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`, last accessed 4 Feb 2011.

[vGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43. 2010.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127. 2005.

Pseudorandom Functions and Lattices

Abhishek Banerjee* Chris Peikert† Alon Rosen‡

September 29, 2011

Abstract

We give direct constructions of pseudorandom function (PRF) families based on conjectured hard lattice problems and learning problems. Our constructions are asymptotically efficient and highly parallelizable in a practical sense, i.e., they can be computed by simple, relatively *small* low-depth arithmetic or boolean circuits (e.g., in NC^1 or even TC^0). In addition, they are the first low-depth PRFs that have no known attack by efficient quantum algorithms. Central to our results is a new "derandomization" technique for the learning with errors (LWE) problem which, in effect, generates the error terms deterministically.

1 Introduction and Main Results

The past few years have seen significant progress in constructing public-key, identity-based, and homomorphic cryptographic schemes using lattices, e.g., [Reg05, PW08, GPV08, Gen09, CHKP10, ABB10a] and many more. Part of their appeal stems from provable worst-case hardness guarantees (starting with the seminal work of Ajtai [Ajt96]), good asymptotic efficiency and parallelism, and apparent resistance to quantum attacks (unlike the classical problems of factoring integers or computing discrete logarithms).

Perhaps surprisingly, there has been comparatively less progress in using lattices for *symmetric* cryptography, e.g., message authentication codes, block ciphers, and the like, which are widely used in practice. While in principle most symmetric objects of interest can be obtained generically from any one-way function, and hence from lattices, these generic constructions are usually very inefficient, which puts them at odds with the high performance demands of most applications. In addition, generic constructions often use their underlying primitives (e.g., one-way functions) in an inherently inefficient and *sequential* manner. While most lattice-based primitives are relatively efficient and highly parallelizable in a practical sense (i.e., they can be evaluated by small, low-depth circuits), those advantages are completely lost when plugging them into generic sequential constructions. This motivates the search for specialized constructions of symmetric objects that have comparable efficiency and parallelism to their lower-level counterparts.

Our focus in this work is on *pseudorandom function* (PRF) families, a central object in symmetric cryptography first rigorously defined and constructed by Goldreich, Goldwasser, and Micali ("GGM") [GGM84].

*School of Computer Science, Georgia Institute of Technology. Email: abhishek.banerjee@cc.gatech.edu. Research supported in part by an ARC Fellowship.

†School of Computer Science, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under Grant CNS-0716786 and CAREER Award CCF-1054495, and by the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

‡Efi Arazi School of Computer Science, IDC Herzliya. Email: alon.rosen@idc.ac.il. Research supported in part by BSF grant 2010296.

Given a PRF family, most central goals of symmetric cryptography (e.g., encryption, authentication, identification) have simple solutions that make efficient use of the PRF. Informally, a family of deterministic functions is pseudorandom if no efficient adversary, given adaptive oracle access to a randomly chosen function from the family, can distinguish it from a uniformly random function. The seminal GGM construction is based generically on any length-doubling pseudorandom generator (and hence on any one-way function), but it requires k *sequential* invocations of the generator when operating on k-bit inputs.

In contrast, by relying on a generic object called a "pseudorandom *synthesizer*," or directly on concrete number-theoretic problems (such as decision Diffie-Hellman, RSA, and factoring), Naor and Reingold [NR95, NR97] and Naor, Reingold, and Rosen [NRR00] (see also [LW09, BMR10]) constructed very elegant and more efficient PRFs, which can in principle be computed in parallel by low-depth circuits (e.g., in NC^2 or TC^0). However, achieving such low depth for their number-theoretic constructions requires extensive preprocessing and enormous circuits, so their results serve mainly as a proof of theoretical feasibility rather than practical utility.

In summary, thus far all parallelizable PRFs from commonly accepted cryptographic assumptions rely on exponentiation in large multiplicative groups, and the functions (or at least their underlying hard problems) can be broken by polynomial-time quantum algorithms. While lattices appear to be a natural candidate for avoiding these drawbacks, and there has been some partial progress in the form of *randomized* weak PRFs [ACPS09] and randomized MACs [Pie10, KPC+11], constructing an efficient, parallelizable (deterministic) PRF under lattice assumptions has, frustratingly, remained open for some time now.

1.1 Results and Techniques

In this work we give the first direct constructions of PRF families based on lattices, via the *learning with errors* (LWE) [Reg05] and *ring*-LWE [LPR10] problems, and some new variants. Our constructions are highly parallelizable in a *practical* sense, i.e., they can be computed by relatively *small* low-depth circuits, and the runtimes are also potentially practical. (However, their performance and key sizes are still far from those of heuristically designed functions like AES.) In addition, (at least) one of our constructions can be evaluated in the circuit class TC^0 (i.e., constant-depth, poly-sized circuits with unbounded fan-in and threshold gates), which asymptotically matches the shallowest known PRF constructions based on the decision Diffie-Hellman and factoring problems [NR97, NRR00].

As a starting point, we recall that in their work introducing *synthesizers* as a foundation for PRFs [NR95], Naor and Reingold described a synthesizer based on a simple, conjectured hard-to-learn function. At first glance, this route seems very promising for obtaining PRFs from lattices, using LWE as the hard learning problem (which is known to be as hard as worst-case lattice problems [Reg05, Pei09]). However, a crucial point is that Naor and Reingold's synthesizer uses a *deterministic* hard-to-learn function, whereas LWE's hardness depends essentially on adding *random, independent* errors to every output of a mod-q "parity" function. (Indeed, without any error, parity functions are trivially easy to learn.) Probably the main obstacle so far in constructing efficient lattice/LWE-based PRFs has been in finding a way to introduce (sufficiently independent) error terms into each of the exponentially many function outputs, while still keeping the function deterministic and its key size a fixed polynomial. As evidence, consider that recent constructions of weaker primitives such as symmetric authentication protocols [HB01, JW05, KSS06], randomized weak PRFs [ACPS09], and message-authentication codes [Pie10, KPC+11] from noisy-learning problems are all inherently *randomized* functions, where security relies on introducing fresh noise at every invocation. Unfortunately, this is not an option for deterministic primitives like PRFs.

Derandomizing LWE**.** To resolve the above-described issues, our first main insight is a way of partially "derandomizing" the LWE problem, i.e., generating the *errors* efficiently and deterministically, while preserving hardness. This technique immediately yields a deterministic synthesizer and hence a simple and parallelizable PRF, though with a few subtleties specific to our technique that we elaborate upon below.

Before we explain the derandomization idea, first recall the learning with errors problem $\mathsf{LWE}_{n,q,\alpha}$ in dimension n (the main security parameter) with modulus q and error rate α. We are given many independent pairs $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where each $\mathbf{a}_i$ is uniformly random, and the b_i are all either "noisy inner products" of the form $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$ for a random secret $\mathbf{s} \in \mathbb{Z}_q^n$ and "small" random error terms $e_i \in \mathbb{Z}$ of magnitude $\approx \alpha q$, or are uniformly random and independent of the $\mathbf{a}_i$. The goal of the (decision) LWE problem is to distinguish between these two cases, with any non-negligible advantage. In the *ring*-LWE problem [LPR10], we are instead given noisy ring products $b_i \approx a_i \cdot s$, where s and the a_i are random elements of a certain polynomial ring R_q (the canonical example being $R_q = \mathbb{Z}_q[z]/(z^n + 1)$ for n a power of 2), and the error terms are "small" in a certain basis of the ring; the goal again is to distinguish these from uniformly random pairs. While the dimension n is the main hardness parameter, the error rate α also plays a very important role in both theory and practice: as long as the "absolute" error αq exceeds $\sqrt{n}$ or so, (ring-)LWE is provably as hard as approximating conjectured hard problems on (ideal) lattices to within $\tilde{O}(n/\alpha)$ factors in the worst case [Reg05, Pei09, LPR10]. Moreover, known attacks using lattice basis reduction (e.g., [LLL82, Sch87]) or combinatorial/algebraic methods [BKW03, AG11] require time $2^{\tilde{\Omega}(n/\log(1/\alpha))}$, where the $\tilde{\Omega}(\cdot)$ notation hides polylogarithmic factors in n. We emphasize that without the error terms, (ring-)LWE would become trivially easy, and that all prior hardness results for LWE and its many variants (e.g., [Reg05, Pei09, GKPV10, LPR10, Pie10]) require random, independent errors.

Our derandomization technique for LWE is very simple: instead of adding a small random error term to each inner product $\langle \mathbf{a}_i, \mathbf{s} \rangle \in \mathbb{Z}_q$, we just deterministically *round* it to the nearest element of a sufficiently "coarse" public subset of $p \ll q$ well-separated values in $\mathbb{Z}_q$ (e.g., a subgroup). In other words, the "error term" comes solely from deterministically rounding $\langle \mathbf{a}_i, \mathbf{s} \rangle$ to a relatively nearby value. Since there are only p possible rounded outputs in $\mathbb{Z}_q$, it is usually easier to view them as elements of $\mathbb{Z}_p$ and denote the rounded value by $\lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p \in \mathbb{Z}_p$. We call the problem of distinguishing such rounded inner products from uniform samples the *learning with rounding* ($\mathsf{LWR}_{n,q,p}$) problem. Note that the problem can be hard only if $q > p$ (otherwise no error is introduced), that the "absolute" error is roughly q/p, and that the "error rate" relative to q (i.e., the analogue of α in the LWE problem) is on the order of $1/p$.

We show that for appropriate parameters, $\mathsf{LWR}_{n,q,p}$ is at least as hard as $\mathsf{LWE}_{n,q,\alpha}$ for an error rate α proportional to $1/p$, giving us a worst-case hardness guarantee for LWR. In essence, the reduction relies on the fact that with high probability, we have $\lfloor \langle \mathbf{a}, \mathbf{s} \rangle + e \rceil_p = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p$ when e is small relative to q/p, while $\lfloor U(\mathbb{Z}_q) \rceil_p \approx U(\mathbb{Z}_p)$ where U denotes the uniform distribution. Therefore, given samples $(\mathbf{a}_i, b_i)$ of an unknown type (either LWE or uniform), we can simply round the b_i terms to generate samples of a corresponding type (LWR or uniform, respectively). (The formal proof is somewhat more involved, because it has to deal with the rare event that the error term changes the rounded value.) In the ring setting, the derandomization technique and hardness proof based on ring-LWE all go through without difficulty as well. While our proof needs both the ratio q/p and the inverse LWE error rate $1/\alpha$ to be slightly super-polynomial in n, the state of the art in attack algorithms indicates that as long as q/p is an integer (so that $\lfloor U(\mathbb{Z}_q) \rceil_p = U(\mathbb{Z}_p)$) and is at least $\Omega(\sqrt{n})$, LWR may be exponentially hard (even for quantum algorithms) for any $p = \mathrm{poly}(n)$, and superpolynomially hard when $p = 2^{n^\epsilon}$ for any $\epsilon < 1$.

We point out that in LWE-based cryptosystems, rounding to a fixed, coarse subset is a common method of removing noise and recovering the plaintext when decrypting a "noisy" ciphertext; here we instead use it to avoid having to introduce any random noise in the first place. We believe that this technique should be useful

in many other settings, especially in symmetric cryptography. For example, the LWR problem immediately yields a simple and practical pseudorandom generator that does not require extracting biased (e.g., Gaussian) random values from its input seed, unlike the standard pseudorandom generators based on the LWE or LPN (learning parity with noise) problems. In addition, the rounding technique and its implications for PRFs are closely related to the "modulus reduction" technique from a concurrent and independent work of Brakerski and Vaikuntanathan [BV11a] on fully homomorphic encryption from LWE, and a very recent follow-up work of Brakerski, Gentry, and Vaikuntanathan [BGV11]; see Section 1.3 below for a discussion and comparison.

LWR-based synthesizers and PRFs. Recall from [NR95] that a pseudorandom *synthesizer* is a two-argument function $S(\cdot,\cdot)$ such that, for random and independent sequences $x_1,\ldots,x_m$ and $y_1,\ldots,y_m$ of inputs (for any $m = \mathrm{poly}(n)$), the matrix of all m^2 values $z_{i,j} = S(x_i, y_j)$ is pseudorandom (i.e., computationally indistinguishable from uniform). A synthesizer can be seen as an (almost) length-*squaring* pseudorandom generator with good locality properties, in that it maps $2m$ random "seed" elements (the x_i and y_j) to m^2 pseudorandom elements, and any component of its output depends on only two components of the input seed.

Using synthesizers in a recursive tree-like construction, Naor and Reingold gave PRFs on k-bit inputs, which can be computed using a total of about k synthesizer evaluations, arranged nicely in only $\lg k$ levels (depth). Essentially, the main idea is that given a synthesizer $S(\cdot,\cdot)$ and two independent PRF instances F_0 and F_1 on t input bits each, one gets a PRF on $2t$ input bits, defined as

$$F(x_1 \cdots x_{2t}) = S\big(\, F_0(x_1 \cdots x_t)\,,\, F_1(x_{t+1} \cdots x_{2t})\,\big). \tag{1.1}$$

The base case of a 1-bit PRF can trivially be implemented by returning one of two random strings in the function's secret key. Using particular NC^1 synthesizers based on a variety of both concrete and general assumptions, Naor and Reingold therefore obtain k-bit PRFs in NC^2, i.e., having circuit depth $O(\log^2 k)$.

We give a very simple and computationally efficient $\mathsf{LWR}_{n,q,p}$-based synthesizer $S_{n,q,p}\colon \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathbb{Z}_p$, defined as

$$S_{n,q,p}(\mathbf{a}, \mathbf{s}) = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p. \tag{1.2}$$

(In this and what follows, products of vectors or matrices over $\mathbb{Z}_q$ are always performed modulo q.) Pseudorandomness of this synthesizer under LWR follows by a standard hybrid argument, using the fact that the $\mathbf{a}_i$ vectors given in the LWR problem are public. (In fact, the synthesizer outputs $S(\mathbf{a}_i, \mathbf{s}_j)$ are pseudorandom even given the $\mathbf{a}_i$.) To obtain a PRF using the tree construction of [NR95], we need the synthesizer output length to roughly match its input length, so we actually use the synthesizer $T_{n,q,p}(\mathbf{S}_1, \mathbf{S}_2) = \lfloor \mathbf{S}_1 \cdot \mathbf{S}_2 \rceil_p \in \mathbb{Z}_p^{n \times n}$ for $\mathbf{S}_i \in \mathbb{Z}_q^{n \times n}$. Note that the matrix multiplication can be done with a constant-depth, size-$O(n^2)$ arithmetic circuit over $\mathbb{Z}_q$. Or for better space and time complexity, we can instead use the ring-LWR synthesizer $S_{R,q,p}(s_1, s_2) = \lfloor s_1 \cdot s_2 \rceil_p$, since the ring product $s_1 \cdot s_2 \in R_q$ is the same size as $s_1, s_2 \in R_q$. The ring product can also be computed with a constant depth, size-$O(n^2)$ circuit over $\mathbb{Z}_q$, or in $O(\log n)$ depth and only $O(n \log n)$ scalar operations using Fast Fourier Transform-like techniques [LMPR08, LPR10].

Using the recursive input-doubling construction from Equation (1.1) above, we get the following concrete PRF with input length $k = 2^d$. Let $q_d > q_{d-1} > \cdots > q_0 \geq 2$ be a chain of moduli where each q_j/q_{j-1} is a sufficiently large integer, e.g., $q_j = q^{j+1}$ for some $q \geq \sqrt{n}$. The secret key is a set of $2k$ matrices $\mathbf{S}_{i,b} \in \mathbb{Z}_{q_d}^{n \times n}$ for each $i \in \{1,\ldots,k\}$ and $b \in \{0,1\}$. Each pair $(\mathbf{S}_{i,0}, \mathbf{S}_{i,1})$ defines a 1-bit PRF $F_i(b) = \mathbf{S}_{i,b}$, and these are combined in a tree-like fashion according to Equation (1.1) using the appropriate synthesizers

$T_{n,q_j,q_{j-1}}$ for $j = d, \ldots, 1$. As a concrete example, when $k = 8$ (so $x = x_1 \cdots x_8$ and $d = 3$), we have

$$F_{\{\mathbf{S}_{i,b}\}}(x) \;=\; \left\lfloor \left\lfloor \lfloor \mathbf{S}_{1,x_1} \cdot \mathbf{S}_{2,x_2} \rceil_{q_2} \cdot \lfloor \mathbf{S}_{3,x_3} \cdot \mathbf{S}_{4,x_4} \rceil_{q_2} \right\rceil_{q_1} \cdot \left\lfloor \lfloor \mathbf{S}_{5,x_5} \cdot \mathbf{S}_{6,x_6} \rceil_{q_2} \cdot \lfloor \mathbf{S}_{7,x_7} \cdot \mathbf{S}_{8,x_8} \rceil_{q_2} \right\rceil_{q_1} \right\rceil_{q_0} . \quad (1.3)$$

(In the ring setting, we just use random elements $s_{i,b} \in R_{q_d}$ in place of the matrices $\mathbf{S}_{i,b}$.) Notice that the function involves $d = \lg k$ levels of matrix (or ring) products, each followed by a rounding operation. In the exemplary case where $q_j = q^{j+1}$, the rounding operations essentially drop the "least-significant" base-q digit, so they can be implemented very easily in practice, especially if every q_j is a power of 2. The function is also amenable to all of the nice time/space trade-offs, seed-compression techniques, and incremental computation ideas described in [NR95].

In the security proof, we rely on the conjectured hardness of $\mathsf{LWR}_{q_j,q_{j-1}}$ for $j = d, \ldots, 1$. The strongest of these assumptions appears to be for $j = d$, and this is certainly the case when relying on our reduction from LWE to LWR. For the example parameters $q_j = q^{j+1}$ where $q \approx \sqrt{n}$, the dominating assumption is therefore the hardness of $\mathsf{LWR}_{q^{d+1},q^d}$, which involves a quasi-polynomial inverse error rate of $1/\alpha \approx q^d = n^{O(\lg k)}$. However, because the strongest assumptions are applied to the "innermost" layers of the function, it is unclear whether security actually *requires* such strong assumptions, or even whether the innermost layers need to be rounded at all. We discuss these issues further in Section 1.2 below.

Degree-k synthesizers and shallower PRFs. One moderate drawback of the above function is that it involves $\lg k$ levels of rounding operations, which appears to lower-bound the depth of any circuit computing the function by $\Omega(\lg k)$. Is it possible to do better?

Recall that in later works, Naor and Reingold [NR97] and Naor, Reingold, and Rosen [NRR00] gave direct, more efficient number-theoretic PRF constructions which, while still requiring exponentiation in large multiplicative groups, can in principle be computed in very shallow circuit classes like NC^1 or even TC^0. Their functions can be interpreted as "degree-k" (or k-argument) synthesizers for arbitrary $k = \mathrm{poly}(n)$, which immediately yield k-bit PRFs without requiring any composition. With this in mind, a natural question is whether there are direct LWE/LWR-based synthesizers of degree $k > 2$.

We give a positive answer to this question. Much like the functions of [NR97, NRR00], ours have a subset-product structure. We have public moduli $q \gg p$, and the secret key is a set of k matrices $\mathbf{S}_i \in \mathbb{Z}_q^{n \times n}$ (whose distributions may not necessarily be uniform; see below) for $i = 1, \ldots, k$, along with a uniformly random $\mathbf{a} \in \mathbb{Z}_q^n$.[1] The function $F = F_{\mathbf{a},\{\mathbf{S}_i\}} \colon \{0,1\}^k \to \mathbb{Z}_p^n$ is defined as the "rounded subset-product"

$$F_{\mathbf{a},\{\mathbf{S}_i\}}(x_1 \cdots x_k) = \left\lfloor \mathbf{a}^t \cdot \prod_{i=1}^{k} \mathbf{S}_i^{x_i} \right\rceil_p . \quad (1.4)$$

The ring variant is analogous, replacing $\mathbf{a}$ with uniform $a \in R_q$ (or R_q^*, the set of invertible elements) and each $\mathbf{S}_i$ by some $s_i \in R_q$. This function is particularly efficient to evaluate using the discrete Fourier transform, as is standard with ring-based primitives (see,e.g., [LMPR08, LPR10]). In addition, similarly to [NR97, NRR00], one can optimize the subset-product operation via pre-processing, and evaluate the function in TC^0. We elaborate on these optimizations in Section 5.2.

For the security analysis of construction (1.4), we have meaningful security proofs under various conditions on the parameters and computational assumptions, including standard LWE. In our LWE-based proof, two important issues are the distribution of the secret key components $\mathbf{S}_i$, and the choice of moduli q and p.

[1] To obtain longer function outputs, we can replace $\mathbf{a} \in \mathbb{Z}_q^n$ with a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for any $m = \mathrm{poly}(n)$.

For the former, it turns out that our proof needs the $\mathbf{S}_i$ matrices to be *short*, i.e., their entries should be drawn from the LWE error distribution. (LWE is no easier to solve for such short secrets [ACPS09].) This appears to be an artifact of our proof technique, which can be viewed as a variant of our LWE-to-LWR reduction, enhanced to handle adversarial queries. Summarizing the approach, define

$$G(x) = G_{\mathbf{a},\{\mathbf{S}_i\}}(x) := \mathbf{a}^t \cdot \prod_i \mathbf{S}_i^{x_i}$$

to be the subset-product function inside the rounding operation of (1.4). The fact that $F = \lfloor G \rceil_p$ lets us imagine adding *independent error terms* to each distinct output of G, but *only as part of a thought experiment* in the proof. More specifically, we consider a related *randomized* function $\tilde{G} = \tilde{G}_{\mathbf{a},\{\mathbf{S}_i\}} \colon \{0,1\}^k \to \mathbb{Z}_q^n$ that computes the subset-product by multiplying by each $\mathbf{S}_i^{x_i}$ in turn, but then also adds a fresh error term immediately following each multiplication. Using the LWE assumption and induction on k, we can show that the randomized function $\tilde{G}$ is itself pseudorandom (over $\mathbb{Z}_q$), hence so is $\lfloor \tilde{G} \rceil_p$ (over $\mathbb{Z}_p$). Moreover, we show that for every queried input, with high probability $\lfloor \tilde{G} \rceil_p$ coincides with $\lfloor G \rceil_p = F$, because G and $\tilde{G}$ differ only by a cumulative error term that is small relative to q—this is where we need to assume that the entries of $\mathbf{S}_i$ are *small*. Finally, because $\lfloor \tilde{G} \rceil_p$ is a (randomized) pseudorandom function over $\mathbb{Z}_p$ that coincides with the deterministic function F on all queries, we can conclude that F is pseudorandom as well.

In the above-described proof strategy, the gap between G and $\tilde{G}$ grows *exponentially* in k, because we add a separate noise term following each multiplication by an $\mathbf{S}_i$, which gets enlarged when multiplied by all the later $\mathbf{S}_i$. So in order to ensure that $\lfloor \tilde{G} \rceil_p = \lfloor G \rceil_p$ on all queries, our LWE-based proof needs both the modulus q and inverse error rate $1/\alpha$ to exceed $n^{\Omega(k)}$. In terms of efficiency and security, this compares rather unfavorably with the quasipolynomial $n^{O(\lg k)}$ bound in the proof for our tree-based construction, though on the positive side, the direct degree-k construction has better circuit depth. However, just as with construction (1.3) it is unclear whether such strong assumptions and large parameters are actually *necessary* for security, or whether the matrices $\mathbf{S}_i$ really need to be short.

In particular, it would be nice if the function in (1.4) were secure if the $\mathbf{S}_i$ matrices were *uniformly random* over $\mathbb{Z}_q^{n \times n}$, because we could then recursively compose the function in a k-ary tree to rapidly extend its input length.[2] It would be even better to have a security proof for a smaller modulus q and inverse error rate $1/\alpha$, ideally both polynomial in n even for large k. While we have been unable to find such a security proof under standard LWE, we do give a very tight proof under a new, interactive "*related samples*" LWE/LWR assumption. Roughly speaking, the assumption says that LWE/LWR remains hard even when the sampled $\mathbf{a}_i$ vectors are related by adversarially chosen subset-products of up to k given random matrices (drawn from some known distribution). This provides some evidence that the function may indeed be secure for appropriately distributed $\mathbf{S}_i$, small modulus q, and large k. For further discussion, see Section 1.2.

PRFs via the GGM construction. The above constructions aim to minimize the depth of the circuit evaluating the PRF. However, if parallel complexity is not a concern, and one wishes to minimize the total amount of work per PRF evaluation (or the seed length), then the original GGM construction with an LWR-based pseudorandom generator may turn out to be even more efficient in practice.

Recall that the GGM construction makes generic use of any length-doubling pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$. The generator's output $G(s)$ is viewed as a pair $(G_0(s), G_1(s))$, where $|G_0(s)| = |G_1(s)| = n$. The key for a member of the PRF family is a seed s for G, and on input $x \in \{0,1\}^k$ the

[2] Note that we can always compose the degree-k function with our degree-2 synthesizers from above, but this would only yield a tree with 2-ary internal nodes.

function is defined as

$$F_s(x_1 \cdots x_k) = G_{x_k}(G_{x_{k-1}}(\cdots G_{x_1}(s)\cdots)). \tag{1.5}$$

As mentioned above, the LWR problem immediately yields a simple and practical pseudorandom generator that, in contrast to the generators obtained from the LWE or LPN problems, does not require extracting biased random error terms from its input seed. By plugging this generator into the GGM construction we immediately get a PRF whose evaluation involves precisely k sequential evaluations of the underlying generator.

The LWR-based generator that we have in mind is a function $G_{\mathbf{A}} : \mathbb{Z}_q^n \to \mathbb{Z}_p^m$, where the moduli $q \gg p$ and the (uniformly random) matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ are publicly known. Given a seed $\mathbf{s} \in \mathbb{Z}_q^n$, the generator is defined as

$$G_{\mathbf{A}}(\mathbf{s}) = \left\lfloor \mathbf{A}^t \cdot \mathbf{s} \right\rceil_p. \tag{1.6}$$

The generator's seed length (in bits) is $n \log_2 q$ and its output length is $m \log_2 p$, which gives an expansion rate of $(m \log_2 p)/(n \log_2 q) = (m/n) \log_q p$. For example, to obtain a length-doubling generator we may set $q = p^2 = 2^{2k} > n$ and $m = 4n$. (Other choices yielding different expansion rates are of course possible.) This choice of parameters has the additional benefit of admitting a practical implementation of the rounding and inner-product operations. Note also that when evaluating the resulting PRF, one can get the required part of $G_{\mathbf{A}}(\mathbf{s})$ by computing only the inner products of $\mathbf{s}$ with the corresponding columns of $\mathbf{A}$, not the entire product $\mathbf{A}^t \cdot \mathbf{s}$.

For an even faster implementation one may replace $G_{\mathbf{A}}$ by its analogous ring variant, obtained by replacing $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with uniform $\mathbf{a} \in R_q^m$, and $\mathbf{s} \in \mathbb{Z}_q^n$ with uniform $s \in R_q$. As noted before, the ring variant is particularly efficient to evaluate using Fast Fourier Transform-like algorithms.

1.2 Discussion and Open Questions

The quasipolynomial $n^{O(\log k)}$ or exponential $n^{O(k)}$ moduli and inverse error rates used in our LWE-based security proofs are comparable to those used in recent fully homomorphic encryption (FHE) schemes (e.g., [Gen09, vGHV10, BV11b, BV11a, BGV11]), hierarchical identity-based encryption (HIBE) schemes (e.g., [CHKP10, ABB10a, ABB10b]), and other lattice-based constructions. However, there appears to be a major difference between our use of such strong assumptions, and that of schemes such as FHE/HIBE in the public-key setting. Constructions of the latter systems actually reveal LWE samples having very small error rates (which are needed to ensure correctness of decryption) to the attacker, and the attacker can break the cryptosystems by solving those instances. Therefore, the underlying assumptions and the true security of the schemes are essentially equivalent. In contrast, our PRF uses (small) errors *only as part of a thought experiment* in the security proof, not for any purpose in the operation of the function itself. This leaves open the possibility that our functions (or slight variants) remain secure even for much larger input lengths and smaller moduli than our proofs require. We conjecture that this is the case, even though we have not yet found security proofs (under standard assumptions) for these more efficient parameters. Certainly, determining whether there are effective cryptanalytic attacks is a very interesting and important research direction.

Note that in our construction (1.4), if we draw the secret key components from the uniform (or error) distribution and allow k to be too large relative to q, then the function can become insecure via a simple attack (and our new "interactive" LWR assumption, which yields a tight security proof, becomes false). This is easiest to see for the ring-based function: representing each $s_i \in R_q$ by its vector of "Fourier coefficients" over $\mathbb{Z}_q^n$, each coefficient is 0 with probability about $1/q$ (depending on the precise distribution of s_i). Therefore, with noticeable probability the product of $k = O(q \log n)$ random s_i will have all-0 Fourier coefficients, i.e., will be $0 \in R_q$. In this case our function will return zero on the all-1s input, in violation

of the PRF requirement. (A similar but more complicated analysis can also be applied to the matrix-based function.) Of course, an obvious countermeasure is just to restrict the secret key components to be *invertible*; to our knowledge, this does not appear to have any drawback in terms of security. In fact, it is possible to show that the decision-(ring-)LWE problem remains hard when the secret is restricted to be invertible (and otherwise drawn from the uniform or error distribution), and this fact may be useful in further analysis of the function with more efficient parameters.

In summary, our work raises several interesting concrete questions, including:

- Is $\mathsf{LWR}_{n,q,p}$ really exponentially hard for $p = \mathrm{poly}(n)$ and sufficiently large integer $q/p = \mathrm{poly}(n)$? Are there stronger worst-case hardness guarantees than our current proof based on LWE?
- Is there a security proof for construction (1.4) (with $k = \omega(1)$) for $\mathrm{poly}(n)$-bounded moduli and inverse error rates, under a non-interactive assumption?
- In construction (1.4), is there a security proof (under a non-interactive assumption) for uniformly random $\mathbf{S}_i$? Is there any provable security advantage to using *invertible* $\mathbf{S}_i$?
- Our derandomization technique and LWR problem require working with moduli q greater than 2. Is there an efficient, parallel PRF construction based on the learning parity with noise (LPN) problem?

1.3 Other Related Work

Most closely related to the techniques in this work are two very recent results of Brakerski and Vaikuntanathan [BV11a] and a follow-up work of Brakerski, Gentry, and Vaikuntanathan [BGV11] on fully homomorphic encryption from LWE. In particular, the former work includes a "modulus reduction" technique for LWE-based cryptosystems, which maps a large-modulus ciphertext to a small-modulus one; this induces a shallower decryption circuit and allows the system to be "bootstrapped" into a fully homomorphic scheme using the techniques of [Gen09]. The modulus-reduction technique involves a rounding operation much like the one we use to derandomize LWE; while they use it on ciphertexts that are already "noisy," we apply it to noise-free LWE samples. Our discovery of the rounding/derandomization technique in the PRF context was independent of [BV11a]. In fact, the first PRF and security proof we found were for the direct degree-k construction defined in (1.4), not the synthesizer-based construction in (1.3). As another point of comparison, the "somewhat homomorphic" cryptosystem from [BV11a] that supports degree-k operations (along with all prior ones, e.g., [Gen09, vGHV10]) involves an inverse error rate of $n^{O(k)}$, much like the LWE-based proof for our degree-k synthesizer.

Building on the modulus reduction technique of [BV11a], Brakerski *et al.* [BGV11] showed that homomorphic cryptosystems can support certain degree-k functions using a much smaller modulus and inverse error rate of $n^{O(\log k)}$. The essential idea is to interleave the homomorphic operations with several "small" modulus-reduction steps in a tree-like fashion, rather than performing all the homomorphic operations followed by one "huge" modulus reduction. This very closely parallels the difference between our direct degree-k synthesizer and the Naor-Reingold-like [NR95] composed synthesizer defined in (1.3). Indeed, after we found construction (1.4), the result of [BGV11] inspired our search for a PRF having similar tree-like structure and quasipolynomial error rates. Given our degree-2 synthesizer, the solution turned out to largely be laid out in the work of [NR95]. We find it very interesting that the same quantitative phenomena arise in two seemingly disparate settings (PRFs and FHE).

1.4 Organization

The rest of the paper is organized as follows. In Section 2 we recall the necessary preliminaries regarding PRFs and the (ring-)LWE problem. In Section 3 we introduce the "learning with rounding" (LWR) problem and discuss its relationship with LWE. In Section 4 we describe LWR-based (degree-2) synthesizers and the PRFs that follow from them. In Section 5 we describe our direct degree-k synthesizer/PRF and its security proofs under the LWE and "subset-product" LWR problem.

2 Preliminaries

For a probability distribution X over a domain D, let X^n denote its n-fold product distribution over D^n. The uniform distribution over a finite domain D is denoted by $U(D)$. The discrete Gaussian probability distribution over $\mathbb{Z}$ with parameter $r > 0$, denoted $D_{\mathbb{Z},r}$, assigns probability proportional to $\exp(-\pi x^2/r^2)$ to each $x \in \mathbb{Z}$. It is possible to efficiently sample from this distribution (up to $\mathrm{negl}(n)$ statistical distance) via rejection [GPV08].

For any integer modulus $q \geq 2$, $\mathbb{Z}_q$ denotes the quotient ring of integers modulo q. We define a 'rounding' function $\lfloor \cdot \rceil_p \colon \mathbb{Z}_q \to \mathbb{Z}_p$, where $q \geq p \geq 2$ will be apparent from the context, as

$$\lfloor x \rceil_p = \lfloor (p/q) \cdot \bar{x} \rceil \bmod p, \tag{2.1}$$

where $\bar{x} \in \mathbb{Z}$ is any integer congruent to $x \bmod q$. We extend $\lfloor \cdot \rceil_p$ component-wise to vectors and matrices over $\mathbb{Z}_q$, and coefficient-wise (with respect to the "power basis") to the quotient ring R_q defined in the next subsection. Note that we can use any other common rounding method, like the floor $\lfloor \cdot \rfloor$, or ceiling $\lceil \cdot \rceil$ functions, in Equation 2.1 above, with only minor changes to our proofs. In implementations, it may be advantageous to use the floor function $\lfloor \cdot \rfloor$ when q and p are both powers of some common base b (e.g., 2). In this setting, computing $\lfloor \cdot \rfloor_p$ is equivalent to dropping the least-significant digit(s) in base b.

2.1 Pseudorandom Functions

The main security parameter through this paper is n, and all algorithms (including the adversary) are implicitly given the security parameter n in unary. We write $\mathrm{negl}(n)$ to denote an arbitrary *negligible* function in n, one that vanishes faster than the inverse of any polynomial. We say that a probability is *overwhelming* if it is $1 - \mathrm{negl}(n)$.

We consider adversaries interacting as part of probabilistic experiments called *games*. For an adversary $\mathcal{A}$ and two games H_0, H_1 with which it can interact, $\mathcal{A}$'s *distinguishing advantage* (implicitly, as a function of n) is $\mathbf{Adv}_{H_0,H_1}(\mathcal{A}) := |\Pr[\mathcal{A} \text{ accepts in } H_0] - \Pr[\mathcal{A} \text{ accepts in } H_1]|$.

Definition 2.1 (Computational Indistinguishability)**.** We say that games H_0, H_1 are *computationally indistinguishable*, written $H_0 \overset{c}{\approx} H_1$, if $\mathbf{Adv}_{H_0,H_1}(\mathcal{A}) = \mathrm{negl}(n)$ for any probabilistic polynomial-time $\mathcal{A}$.

By the triangle inequality, $\overset{c}{\approx}$ is a transitive relation over any $\mathrm{poly}(n)$-length sequence of games. If $H_0 \overset{c}{\approx} H_1$ and $\mathcal{S}$ is any probabilistic polynomial-time algorithm, then the outputs of $\mathcal{S}$ playing in games H_0 and H_1 (respectively) are also computationally indistinguishable.

Definition 2.2 (Pseudorandom functions)**.** Let A and B be finite sets, and let $\mathcal{F} = \{F_i \colon A \to B\}$ be a function family, endowed with an efficiently sampleable distribution (more precisely, $\mathcal{F}$, A and B are all indexed by the security parameter n). We say that $\mathcal{F}$ is a *pseudorandom function* (PRF) family if the following two games are computationally indistinguishable:

1. Choose a function $F \leftarrow \mathcal{F}$ and give the adversary adaptive oracle access to $F(\cdot)$.

2. Choose a uniformly random function $U\colon A \to B$ and give the adversary adaptive oracle access to $U(\cdot)$.

To efficiently simulate access to a uniformly random function $U\colon A \to B$, one may think of a process in which the adversary's queries are "lazily" answered with independently and randomly chosen elements in B, while keeping track of the answers so that queries made more than once are answered consistently.

2.2 (Ring) Learning With Errors

We recall the learning with errors (LWE) problem due to Regev [Reg05] and its ring analogue by Lyubashevsky, Peikert, and Regev [LPR10]. For positive integer dimension n (the security parameter) and modulus $q \geq 2$, a probability distribution χ over $\mathbb{Z}$, and a vector $\mathbf{s} \in \mathbb{Z}_q^n$, define the LWE distribution $A_{\mathbf{s},\chi}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, an error term $e \leftarrow \chi$, and outputting $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q)$. We use the following "normal form" of the *decision*-$\mathsf{LWE}_{n,q,\chi}$ problem, which is to distinguish (with advantage non-negligible in n) between any desired number $m = \mathrm{poly}(n)$ of independent samples $(\mathbf{a}_i, b_i) \leftarrow A_{\mathbf{s},\chi}$ where $\mathbf{s} \leftarrow \chi^n \bmod q$ is chosen from the (folded) error distribution, and the same number of samples from the uniform distribution $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. This form of the problem is as hard as the one where $\mathbf{s} \in \mathbb{Z}_q^n$ is chosen uniformly at random [ACPS09].

We extend the LWE distribution to $w \geq 1$ secrets, defining $A_{\mathbf{S},\chi}$ for $\mathbf{S} \in \mathbb{Z}_q^{n \times w}$ to be the distribution obtained by choosing $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, an error vector $\mathbf{e}^t \leftarrow \chi^w$, and outputting $(\mathbf{a}, \mathbf{b}^t = \mathbf{a}^t\mathbf{S} + \mathbf{e}^t \bmod q)$. By a standard hybrid argument, distinguishing such samples (for $\mathbf{S} \leftarrow \chi^{n \times w}$) from uniformly random is as hard as decision-$\mathsf{LWE}_{n,q,\chi}$, for any $w = \mathrm{poly}(n)$. It is often convenient to group many (say, m) sample pairs together in matrices. This allows us to express the LWE problem as: distinguish any desired number of pairs $(\mathbf{A}^t, \mathbf{B}^t = \mathbf{A}^t\mathbf{S} + \mathbf{E} \bmod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times w}$, for the same $\mathbf{S}$, from uniformly random.

For certain moduli q and (discrete) Gaussian error distributions χ, the decision-LWE problem is as hard as the search problem, where the goal is to find $\mathbf{s}$ given samples from $A_{\mathbf{s},\chi}$ (see, e.g., [Reg05, Pei09, ACPS09, MM11], and [MP11] for the mildest known requirements on q, which include the case where q is a power of 2). In turn, for $\chi = D_{\mathbb{Z},r}$ with $r = \alpha q \geq 2\sqrt{n}$, the search problem is as hard as approximating worst-case lattice problems to within $\tilde{O}(n/\alpha)$ factors; see [Reg05, Pei09] for precise statements.[3]

Ring-LWE. For simplicity of exposition, we use the following special case of the ring-LWE problem. (Our results can be extended to the more general form defined in [LPR10].) Throughout the paper we let R denote the cyclotomic polynomial ring $R = \mathbb{Z}[z]/(z^n + 1)$ for n a power of 2. (Equivalently, R is the ring of integers $\mathbb{Z}[\omega]$ for $\omega = \exp(\pi i/n)$.) For any integer modulus q, define the quotient ring $R_q = R/qR$. An element of R can be represented as a polynomial (in z) of degree less than n having integer coefficients; in other words, the "power basis" $\{1, z, \ldots, z^{n-1}\}$ is a $\mathbb{Z}$-basis for R. Similarly, it is a $\mathbb{Z}_q$-basis for R_q.

For a modulus q, a probability distribution χ over R, and an element $s \in R_q$, the ring-LWE (RLWE) distribution $A_{s,\chi}$ is the distribution over $R_q \times R_q$ obtained by choosing $a \in R_q$ uniformly at random, an error term $x \leftarrow \chi$, and outputting $(a, b = a \cdot s + x \bmod qR)$. The normal form of the decision-$\mathsf{RLWE}_{R,q,\chi}$ problem is to distinguish (with non-negligible advantage) between any desired number $m = \mathrm{poly}(n)$ of independent samples $(a_i, b_i) \leftarrow A_{s,\chi}$ where $s \leftarrow \chi \bmod q$, and the same number of samples drawn from the uniform

[3] It is important to note that the original hardness result of [Reg05] for search-LWE is for a *continuous* Gaussian error distribution, which when rounded naïvely to the nearest integer does not produce a true discrete Gaussian $D_{\mathbb{Z},r}$. Fortunately, a suitable randomized rounding method does so [Pei10].

distribution $U(R_q \times R_q)$. We will use the error distribution χ over R where each coefficient (with respect to the power basis) is chosen independently from the discrete Gaussian $D_{\mathbb{Z},r}$ for some $r = \alpha q \geq \omega(\sqrt{n \log n})$.

For a prime modulus $q = 1 \bmod 2n$ and the error distribution χ described above, the decision-RLWE problem is as hard as the search problem, via a reduction that runs in time $q \cdot \mathrm{poly}(n)$ [LPR10]. In turn, the search problem is as hard as quantumly approximating worst-case problems on ideal lattices.[4]

To bound products of samples drawn from the error distribution χ over R, we recall a useful result from [LPR10].

Lemma 2.3. *Let χ be the distribution over the ring R where each coefficient (with respect to the power basis) is chosen independently from $D_{\mathbb{Z},r}$ for some $r > 0$, and let $t = \omega(\sqrt{\log n})$ denote any function that grows asymptotically faster than $\sqrt{\log n}$. Then in the product of $k \geq 1$ independent samples drawn from χ, every coefficient is bounded in magnitude by $(r\sqrt{n} \cdot t)^k/\sqrt{n}$, except with $\exp(-\Omega(t^2)) = \mathrm{negl}(n)$ probability.*

2.3 Subgaussian Distributions and Random Matrices

A random variable X over $\mathbb{R}$ (or its distribution) with $\mathbb{E}[X] = 0$ is *subgaussian* with parameter $r > 0$ if it has Gaussian tails, i.e., for all $t > 0$, $\Pr[|X| > t] \leq 2\exp(-\pi(t/r)^2)$.[5] In particular, $D_{\mathbb{Z},r}$ is subgaussian with parameter r [Ban95]. Here we recall a useful result from the non-asymptotic theory of random matrices [Ver11], which bounds the largest singular value (sometimes called the *spectral norm*) $s_1(\mathbf{X}) := \max_{\mathbf{u} \neq \mathbf{0}} \|\mathbf{X}\mathbf{u}\|/\|\mathbf{u}\|$ of a matrix with independent subgaussian entries.

Lemma 2.4. *Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a random matrix (or vector) whose entries are drawn independently from (not necessarily identical) subgaussian distributions with common parameter r. There exists a universal constant $C > 0$ such that $s_1(\mathbf{X}) \leq r \cdot C(\sqrt{m} + \sqrt{n})$ except with probability at most $2^{-\Omega(m+n)}$.*

3 The Learning With Rounding Problem

We now define the "learning with rounding" (LWR) problem and its ring analogue, which are like "derandomized" versions of the usual (ring)-LWE problems, in that the error terms are chosen deterministically.

Definition 3.1. Let $n \geq 1$ be the main security parameter and moduli $q \geq p \geq 2$ be integers.

- For a vector $\mathbf{s} \in \mathbb{Z}_q^n$, define the LWR distribution $L_{\mathbf{s}}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_p$ obtained by choosing a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, and outputting $(\mathbf{a}, b = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p)$.
- For $s \in R_q$ (defined in Section 2.2), define the ring-LWR (RLWR) distribution L_s to be the distribution over $R_q \times R_p$ obtained by choosing $a \leftarrow R_q$ uniformly at random and outputting $(a, b = \lfloor a \cdot s \rceil_p)$.

For a given distribution over $\mathbf{s} \in \mathbb{Z}_q^n$ (e.g., the uniform distribution), the decision-LWR$_{n,q,p}$ problem is to distinguish (with advantage non-negligible in n) between any desired number of independent samples $(\mathbf{a}_i, b_i) \leftarrow L_{\mathbf{s}}$, and the same number of samples drawn uniformly and independently from $\mathbb{Z}_q^n \times \mathbb{Z}_p$. The decision-RLWR$_{R,q,p}$ problem is defined analogously.

[4] More accurately, to prove that the search problem is hard for an a priori *unbounded* number of RLWE samples, the worst-case connection from [LPR10] requires the error distribution's parameters to themselves be chosen at random from a certain distribution. Our constructions are easily modified to account for this subtlety, but for simplicity, we ignore this issue and assume hardness for a fixed, public error distribution.

[5] This simple definition will suffice for our purposes, because we will always use mean-zero distributions. For a more general definition that applies to any distribution, see [MP11].

Note that we have defined LWR exclusively as a decision problem, as this is the only form of the problem we will need. By a simple (and by now standard) hybrid argument, the (ring-)LWR problem is no easier, up to a $\mathrm{poly}(n)$ factor in advantage, if we reuse each public $\mathbf{a}_i$ across several independent secrets. That is, distinguishing samples $(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s}_1 \rangle \rceil_p, \ldots, \lfloor \langle \mathbf{a}_i, \mathbf{s}_\ell \rangle \rceil_p) \in \mathbb{Z}_q^n \times \mathbb{Z}_p^\ell$ from uniform, where each $\mathbf{s}_j \in \mathbb{Z}_q^n$ is chosen independently for any $\ell = \mathrm{poly}(n)$, is at least as hard as decision-LWR for a single secret $\mathbf{s}$. An analogous statement also holds for ring-LWR.

3.1 Reduction from LWE

We now show that for appropriate parameters, decision-LWR is at least as hard as decision-LWE. We say that a probability distribution χ over $\mathbb{R}$ (more precisely, a family of distributions χ_n indexed by the security parameter n) is *B-bounded* (where $B = B(n)$ is a function of n) if $\Pr_{x \leftarrow \chi}[|x| > B] \leq \mathrm{negl}(n)$. Similarly, a distribution over the ring R is B-bounded if the marginal distribution of every coefficient (with respect to the power basis) of an $x \leftarrow \chi$ is B-bounded.

Theorem 3.2. *Let χ be any efficiently sampleable B-bounded distribution over $\mathbb{Z}$, and let $q \geq p \cdot B \cdot n^{\omega(1)}$. Then for any distribution over the secret $\mathbf{s} \in \mathbb{Z}_q^n$, solving decision-$\mathsf{LWR}_{n,q,p}$ is at least as hard as solving decision-$\mathsf{LWE}_{n,q,\chi}$ for the same distribution over $\mathbf{s}$. The same holds true for $\mathsf{RLWR}_{R,q,p}$ and $\mathsf{RLWE}_{R,q,\chi}$, for any B-bounded χ over R.*

We note that although our proof uses a super-polynomial $q = n^{\omega(1)}$, as long as $q/p \geq \sqrt{n}$ is an integer, the LWR problem appears to be exponentially hard (in n) for any $p = \mathrm{poly}(n)$, and super-polynomially hard for $p \leq 2^{n^\epsilon}$ for any $\epsilon < 1$, given the state of the art in noisy learning algorithms [BKW03, AG11] and lattice reduction algorithms [LLL82, Sch87]. We also note that in our proof, we do not require the error terms drawn from χ in the LWE samples to be independent; we just need them all to have magnitude bounded by B with overwhelming probability.

Proof of Theorem 3.2. We give a detailed proof for the LWR case; the one for RLWR proceeds essentially identically. The main idea behind the reduction is simple: given pairs $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ which are distributed either according to an LWE distribution $A_{\mathbf{s},\chi}$ or are uniformly random, we translate them into the pairs $(\mathbf{a}_i, \lfloor b_i \rceil_p) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, which we show will be distributed according to the LWR distribution $L_{\mathbf{s}}$ (with overwhelming probability) or uniformly random, respectively. Proving this formally takes some care, however. We proceed via a sequence of games.

Game H_0. This is the real attack game against the LWR distribution. That is, we choose $\mathbf{s}$ and upon request generate and give the attacker independent samples from $L_{\mathbf{s}}$.

Game H_1. Here the attack is against a 'rounded' version of the LWE distribution $A_{\mathbf{s},\chi}$. That is, we first choose $\mathbf{s}$. Then each time the attacker requests a sample, we generate a pair $(\mathbf{a}, b)$ distributed according to $A_{\mathbf{s},\chi}$ (that is, choose $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and $b = \langle \mathbf{a}, \mathbf{s} \rangle + x$ for $x \leftarrow \chi$), and return the pair $(\mathbf{a}, \lfloor b \rceil_p)$, but with one exception: we define a 'bad event' BAD to be

$$\mathsf{BAD} := \lfloor b + [-B, B] \rceil_p \neq \{ \lfloor b \rceil_p \} .$$

That is, BAD indicates whether b is "too close" to some value in $\mathbb{Z}_q$ having a different rounded value. (In other words, rounding the sample $(\mathbf{a}, b)$ from $A_{\mathbf{s},\chi}$ may give a different result than the corresponding sample

$(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p)$ from the $L_{\mathbf{s}}$ distribution.) If BAD occurs on any of the attacker's requests for a sample, we immediately abort the game.

If BAD does not occur for a particular sample $(\mathbf{a}, b)$, then we have $\lfloor b \rceil_p := \lfloor \langle \mathbf{a}, \mathbf{s} \rangle + x \rceil_p = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p$ with overwhelming probability over the choice of $x \leftarrow \chi$, because χ is B-bounded. It immediately follows that for any (potentially unbounded) attacker $\mathcal{A}$,

$$\mathbf{Adv}_{H_0,H_1}(\mathcal{A}) \leq \Pr[\mathsf{BAD} \text{ occurs in } H_1 \text{ with attacker } \mathcal{A}] + \mathrm{negl}(n). \tag{3.1}$$

We do not directly bound the probability of BAD occurring in H_1, instead deferring it to the analysis of the next game, where we can show that it is indeed negligible.

Game H_2. Here whenever the attacker requests a sample, we choose $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ uniformly at random and give $(\mathbf{a}, \lfloor b \rceil_p)$ to the attacker, subject to the same "bad event" and abort condition as described in the game H_1 above. Under the decision-LWE assumption and by the fact that BAD can be tested efficiently given b, a straightforward reduction implies that $\mathbf{Adv}_{H_1,H_2}(\mathcal{A}) \leq \mathrm{negl}(n)$ for any efficient attacker $\mathcal{A}$. For the same reason, it also follows that

$$|\Pr[\mathsf{BAD} \text{ occurs in } H_1] - \Pr[\mathsf{BAD} \text{ occurs in } H_2]| \leq \mathrm{negl}(n).$$

Now for each uniform b, $\Pr[\mathsf{BAD} \text{ occurs on } b \text{ in } H_2] \leq (2B+1) \cdot p/q = \mathrm{negl}(n)$, by assumption on q. It follows by a union bound over all the samples, and Equation (3.1), that

$$\Pr[\mathsf{BAD} \text{ occurs in } H_1 \text{ with } \mathcal{A}] \leq \mathrm{negl}(n) \quad \Rightarrow \quad \mathbf{Adv}_{H_0,H_1}(\mathcal{A}) = \mathrm{negl}(n).$$

Game H_3. This game is similar to the game H_2, with pairs $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ being chosen uniformly at random and BAD being defined similarly. However, in this game we always return $(\mathbf{a}, \lfloor b \rceil_p)$ to the attacker, even when BAD occurs. By the analysis above, we have that for any (potentially unbounded) attacker $\mathcal{A}$,

$$\mathbf{Adv}_{H_2,H_3}(\mathcal{A}) \leq \Pr[\mathsf{BAD} \text{ occurs in } H_3 \text{ with } \mathcal{A}] = \Pr[\mathsf{BAD} \text{ occurs in } H_2 \text{ with } \mathcal{A}] = \mathrm{negl}(n).$$

Game H_4. In this game we give the attacker samples drawn uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_p$. The statistical distance between $U(\mathbb{Z}_q^n \times \mathbb{Z}_p)$ and $U(\mathbb{Z}_q^n) \times \lfloor U(\mathbb{Z}_q) \rceil_p$ is at most $p/q = \mathrm{negl}(n)$ by assumption on q, so by a union bound over all the $\mathrm{poly}(n)$ samples, we have $\mathbf{Adv}_{H_3,H_4}(\mathcal{A}) = \mathrm{negl}(n)$ for any efficient attacker $\mathcal{A}$.

Finally, by the triangle inequality, we have $\mathbf{Adv}_{H_0,H_4}(\mathcal{A}) = \mathrm{negl}(n)$ for any efficient adversary $\mathcal{A}$, which completes the proof. Essentially the same proof works for the RLWR problem as well. □

4 Synthesizer-Based PRFs

We now describe the LWR-based synthesizer and our construction of a PRF from it. We first define a *pseudorandom synthesizer*, slightly modified from the definition proposed by Naor and Reingold [NR95].

Let $S : A \times A \to B$ be a function (where A and B are finite domains, which along with S are implicitly indexed by the security parameter n) and let $X = (x_1, \ldots, x_k) \in A^k$ and $Y = (y_1, \ldots, y_\ell) \in A^\ell$ be two sequences of inputs. Then $\mathbf{C}_S(X, Y) \in B^{k \times \ell}$ is defined to be the matrix with $S(x_i, y_j)$ as its (i, j)th entry. (Here $\mathbf{C}$ stands for combinations.)

Definition 4.1 (Pseudorandom Synthesizer)**.** We say that a function $S : A \times A \to B$ is a *pseudorandom synthesizer* if it is polynomial-time computable, and if for every $\mathrm{poly}(n)$-bounded $k = k(n)$, $\ell = \ell(n)$,

$$\mathbf{C}_S\big(\, U(A^k)\,,\, U(A^\ell)\,\big) \overset{c}{\approx} U\big(B^{k\times\ell}\big).$$

That is, the matrix $\mathbf{C}_S(X, Y)$ for uniform and independent $X \leftarrow A^k$, $Y \leftarrow A^\ell$ is computationally indistinguishable from a uniformly random k-by-ℓ matrix over B.

4.1 Synthesizer Constructions

We now describe synthesizers whose security is based on the (ring-)LWR problem.

Definition 4.2 ((Ring-)LWR Synthesizer)**.** For moduli $q > p \geq 2$, the LWR synthesizer is the function $S_{n,q,p} \colon \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathbb{Z}_p$ defined as

$$S_{n,q,p}(\mathbf{x}, \mathbf{y}) = \lfloor \langle \mathbf{x}, \mathbf{y} \rangle \rceil_p.$$

The RLWR synthesizer is the function $S_{R,q,p} \colon R_q \times R_q \to R_p$ defined as

$$S_{R,q,p}(x, y) = \lfloor x \cdot y \rceil_p.$$

Theorem 4.3. *Assuming the hardness of decision-*$\mathsf{LWR}_{n,q,p}$ *(respectively, decision-*$\mathsf{RLWR}_{R,q,p}$*) for a uniformly random secret, the function* $S_{n,q,p}$ *(respectively,* $S_{R,q,p}$*) given in Definition 4.2 above is a pseudorandom synthesizer.*

It follows generically from this theorem that the function $T_{n,q,p} \colon \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$, defined as $T_{n,q,p}(\mathbf{X}, \mathbf{Y}) = \lfloor \mathbf{X} \cdot \mathbf{Y} \rceil_p$, is also a pseudorandom synthesizer, since by the definition of matrix multiplication, we only incur a factor of n increase in the length of the input sequences. This is the synthesizer that we use below in the construction of a PRF.

Proof of Theorem 4.3. Let $\ell, k = \mathrm{poly}(n)$ be arbitrary. Let $X = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ be uniformly random and independent sequences of $\mathbb{Z}_q^n$-vectors. Assuming the hardness of "multiple secrets" version of decision-$\mathsf{LWR}_{n,q,p}$ (see the remark following Definition 3.1), we have that the tuples

$$\big(\mathbf{x}_i, \lfloor \langle \mathbf{x}_i, \mathbf{y}_1 \rangle \rceil_p, \ldots, \lfloor \langle \mathbf{x}_i, \mathbf{y}_\ell \rangle \rceil_p\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_p^\ell$$

for $i = 1, \ldots, k$ are computationally indistinguishable from uniform and independent. That is,

$$\big((\mathbf{x}_i)_{i\in[k]}, \mathbf{C}_S(X, Y)\big) \overset{c}{\approx} U(\mathbb{Z}_q^{n\times k} \times \mathbb{Z}_p^{k\times\ell}).$$

From this stronger fact, we have that $\mathbf{C}_S(X, Y) \overset{c}{\approx} U(\mathbb{Z}_p^{k\times\ell})$, as desired. Essentially the same proof works for the RLWR synthesizer as well. □

4.2 The PRF Construction

Definition 4.4 ((Ring-)LWR PRF)**.** For parameters $n \in \mathbb{N}$, input length $k = 2^d \geq 1$, and moduli $q_d \geq q_{d-1} \geq \ldots \geq q_0 \geq 2$, the LWR family $\mathcal{F}^{(j)}$ for $0 \leq j \leq d$ is defined inductively to consist of functions from $\{0,1\}^{2^j}$ to $\mathbb{Z}_{q_{d-j}}^{n\times n}$. We define $\mathcal{F} = \mathcal{F}^{(d)}$.

- For $j = 0$, a function $F \in \mathcal{F}^{(0)}$ is indexed by $\mathbf{S}_b \in \mathbb{Z}_{q_d}^{n\times n}$ for $b \in \{0,1\}$, and is defined simply as $F_{\{\mathbf{S}_b\}}(x) = \mathbf{S}_x$. We endow $\mathcal{F}^{(0)}$ with the distribution where the $\mathbf{S}_b$ are uniform and independent.

- For $j \geq 1$, a function $F \in \mathcal{F}^{(j)}$ is indexed by some $F_0, F_1 \in \mathcal{F}^{(j-1)}$, and is defined as

$$F_{F_0,F_1}(x_0, x_1) = T^{(j)}\big(\, F_0(x_0)\,,\, F_1(x_1)\,\big)$$

  where $|x_0| = |x_1| = 2^{j-1}$ and $T^{(j)} = T_{n,q_{d-j+1},q_{d-j}}$ is the appropriate synthesizer. We endow $\mathcal{F}^{(j)}$ with the distribution where F_0 and F_1 are chosen independently from $\mathcal{F}^{(j-1)}$.

More explicitly, $F \in \mathcal{F}$ is indexed by a set of matrices $\{\mathbf{S}_{i,b}\}$ (where $i \in [k]$, $b \in \{0,1\}$), and for input $x = x_1 \cdots x_k$ is defined as

$$F_{\{\mathbf{S}_{i,b}\}}(x) \;=\; \Big\lfloor \cdots \Big\lfloor \lfloor \mathbf{S}_{1,x_1} \cdot \mathbf{S}_{2,x_2} \rceil_{q_{d-1}} \cdot \lfloor \mathbf{S}_{3,x_3} \cdot \mathbf{S}_{4,x_4} \rceil_{q_{d-1}} \Big\rceil_{q_{d-2}} \cdots \lfloor \mathbf{S}_{k-1,x_{k-1}} \cdot \mathbf{S}_{k,x_k} \rceil_{q_{d-1}} \cdots \Big\rceil_{q_0}.$$

The ring-LWR family $\mathcal{RF}^{(j)}$ is defined similarly to consist of functions from $\{0,1\}^{2^j}$ to $R_{q_{d-j}}$, where in the base case ($j = 0$) we replace each $\mathbf{S}_b$ with a uniformly random $s_b \in R_{q_d}$, and in the inductive case ($j \geq 1$) we use the ring-LWR synthesizer $S^{(j)} = S_{R,q_{d-j+1},q_{d-j}}$.

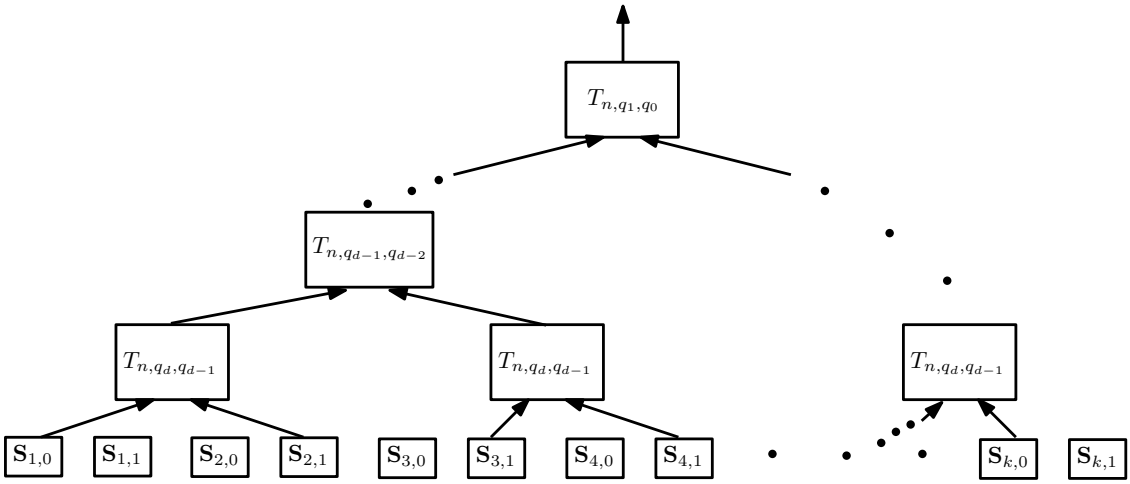


Figure 1: The synthesizer-based PRF evaluated on the input $0111\ldots0$

We remark that the recursive LWR-based construction above does not have to use *square* matrices; any legal dimensions would be acceptable with no essential change to the security proof. Square matrices appear to give the best combination of seed size, computational efficiency, and input/output lengths.

4.3 Efficiency

Consider a function in either one of the families $\mathcal{F}$ or $\mathcal{RF}$ from Definition 4.4. Computing the function at any given point $x \in \{0,1\}^k$ can be done in a tree-like fashion using a tree of depth $d = \lg k$, where each node of the tree corresponds to an evaluation of an appropriate synthesizer. Each synthesizer involves a single matrix (or ring) product mod q_{d-j+1}, followed by a rounding step. Here we discuss implementations of the synthesizers, describing both the simplest practical methods along with depth-optimized parallel solutions (which rely on preprocessing and use larger circuits). In summary, the synthesizers can be computed by small, low-depth arithmetic circuits; moreover, in principle they can be implemented in TC^0, the class of

constant-depth, $\mathrm{poly}(n)$-sized circuits with unbounded fan-in and threshold gates (which is a subset of NC^1). Therefore, the PRFs can be implemented in TC^1, which matches the best constructions from [NR95].

In a practical sequential implementation, we can use any fast matrix multiplication algorithm (e.g., Strassen's), and we can multiply ring elements (in the standard power basis) in $O(n \log n)$ scalar operations mod q (see, e.g., [LPR10]). In a practical parallel implementation, we can compute a matrix multiplication in the natural way using a size-$O(n^2)$, depth-2 arithmetic circuit over $\mathbb{Z}_q$, where the first layer of multiplication gates have fan-in 2 and the second layer of addition gates has fan-in n. The same is true for a product of ring elements in R_q, since it can be expressed as a matrix-vector product: multiplication by any fixed element $a \in R_q$ is a linear transformation.

For computing a synthesizer $T_{n,q,p}$ in TC^0, we note that a matrix product consists of n^2 parallel inner products of n-dimensional vectors, which each involve a multi-sum of (binary) scalar products modulo q. The subsequent rounding step simply amounts to dropping some of the least-significant digits if q and p are both powers of the same small base, or more generally, multiplying by p/q (under suitable precision) and truncating. Both operations can be performed in TC^0, for any $q = 2^{\mathrm{poly}(n)}$ [RT92].

Interestingly, once we allow for threshold gates, there seems to be no asymptotic improvement in depth for the ring-based synthesizer $S_{R,q,p}$. This is because threshold circuits enable binary matrix product to be computed in constant depth, and the depth of computing the PRF is anyway dominated by d, the depth of the tree. The gains in efficiency obtained by using a ring-based construction will be much more pronounced in the case of the degree-k synthesizers described in Section 5. We discuss these gains in detail in Section 5.2.

We remark that Naor and Reingold [NR95] describe several nice optimizations and additional features of their synthesizer-based PRFs, including compression of the secret key and faster amortized computation for a sequence of related inputs. Our functions are amenable to all these techniques as well.

4.4 Security Proof

The security proof for our PRF hinges on the fact that the functions $T^{(j)} = T_{n,q_{d-j+1},q_{d-j}}$ are synthesizers for appropriate choices of the moduli. In fact, the proof is essentially identical to Naor and Reingold's [NR95] for their PRF construction from pseudorandom synthesizers; the only reason we cannot use their theorem exactly as stated is because they assume that the synthesizer output is exactly the same size as its two inputs, which is not quite the case with our synthesizer due to the modulus reduction. This is a minor detail that does not change the proof in any material way; it only limits the number of times we may compose the synthesizer, and hence the input length of the PRF.

Theorem 4.5. *Assuming that $T^{(j)} = T_{n,q_{d-j+1},q_{d-j}}$ is a pseudorandom synthesizer for every $j \in [d]$ (in particular, assuming the hardness of decision-$\mathsf{LWR}_{n,q_{d-j+1},q_{d-j}}$), the LWR family $\mathcal{F}$ from Definition 4.4 is a pseudorandom function family.*

The same holds for the ring-LWR family $\mathcal{RF}$, assuming that $S^{(j)} = S_{R,q_{d-j+1},q_{d-j}}$ is a pseudorandom synthesizer for every $j \in [d]$ (in particular, assuming the hardness of decision-$\mathsf{RLWR}_{R,q_{d-j+1},q_{d-j}}$).

Proof. We give a detailed proof for the family $\mathcal{F}$; the one for $\mathcal{RF}$ proceeds essentially identically. We prove that each $\mathcal{F}^{(j)}$ is a pseudorandom function family by induction for $j = 0, \ldots, d$. The case $j = 0$ is trivial by construction of $\mathcal{F}^{(0)}$. Assuming the inductive hypothesis on $\mathcal{F}^{(j-1)}$ for some $j \geq 1$, we prove the claim for $\mathcal{F}^{(j)}$ via the following series of games.

Game H_0. This is the PRF attack game against $\mathcal{F}^{(j)}$: we choose an $F \leftarrow \mathcal{F}^{(j)}$, i.e., choose $F_0, F_1 \leftarrow \mathcal{F}^{(j-1)}$ independently, and give the attacker oracle access to $F_{F_0,F_1}(x_0, x_1) = T^{(j)}(F_0(x_0), F_1(x_1))$, where as always $|x_0| = |x_1| = 2^{j-1}$.

Game H_1. We replace F_0, F_1 above with truly uniform functions. Specifically, we (lazily) choose two uniform and independent functions $U_0, U_1 \colon \{0,1\}^{2^{j-1}} \to \mathbb{Z}_{q_{d-j+1}}^{n \times n}$. On each query $x = (x_0, x_1) \in \{0,1\}^{2^j}$, we return $T^{(j)}(U_0(x_0), U_1(x_1))$. By a trivial reduction using the inductive hypothesis that $\mathcal{F}^{(j-1)}$ is a PRF family (so F_0, F_1 are computationally indistinguishable from U_0, U_1 given query access), this game is computationally indistinguishable from H_0.

Game H_2. We give the attacker oracle access to a (lazily defined) uniform function $U : \{0,1\}^{2^j} \to \mathbb{Z}_{q_{d-j}}^{n \times n}$.

We claim that games H_0 and H_1 are computationally indistinguishable, because $T^{(j)}$ is a synthesizer by hypothesis. Suppose that an efficient adversary $\mathcal{A}$ makes at most $Q = \mathrm{poly}(n)$ total queries. We design an efficient simulator $\mathcal{S}$ which, given input $(\mathbf{Z}_{i,j})_{i,j \in [Q]} \in (\mathbb{Z}_{q_{d-j}}^{n \times n})^{Q \times Q}$ where either $\mathbf{Z}_{i,j} = T^{(j)}(\mathbf{X}_i, \mathbf{Y}_j)$ for some uniformly random and independent $\mathbf{X}_i, \mathbf{Y}_j \in \mathbb{Z}_{q_{d-j+1}}^{n \times n}$ for $i, j \in [Q]$, or each $\mathbf{Z}_{i,j}$ is uniformly random and independent, simulates game H_1 or H_2, respectively. Because the two types of inputs to $\mathcal{S}$ are computationally indistinguishable by assumption on $T^{(j)}$ (and $\mathcal{S}$ is efficient), it follows that games H_1 and H_2 are indistinguishable as well.

$\mathcal{S}$ works as follows: starting from $i = j = 1$, on each query $x = (x_0, x_1) \in \{0,1\}^{2^j}$ from $\mathcal{A}$, look up whether x_0 (respectively, x_1) is already associated with an index $\hat{\imath}$ (resp., $\hat{\jmath}$); if not, associate it with the current value of i (resp., j) and increment that variable. Return the associated matrix $\mathbf{Z}_{\hat{\imath},\hat{\jmath}}$ to $\mathcal{A}$. It is clear by inspection that the behavior of $\mathcal{S}$ is as claimed above.

We conclude that game H_0 is computationally indistinguishable from game H_2, i.e., that $\mathcal{F}^{(j)}$ is a pseudorandom function family, as desired. □

5 Direct PRF Constructions

Here we present another, potentially more efficient construction of a pseudorandom function family whose security is based on the intractibility of the LWE problem.

5.1 Constructions

Definition 5.1 ((Ring-)LWE degree-k PRF)**.** For parameters $n \in \mathbb{N}$, moduli $q \geq p \geq 2$, positive integer $m = \mathrm{poly}(n)$, and input length $k \geq 1$, the family $\mathcal{F}$ consists of functions from $\{0,1\}^k$ to $\mathbb{Z}_p^{m \times n}$. A function $F \in \mathcal{F}$ is indexed by some $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S}_i \in \mathbb{Z}^{n \times n}$ for each $i \in [k]$, and is defined as

$$F(x) = F_{\mathbf{A},\{\mathbf{S}_i\}}(x_1 \cdots x_k) := \left\lfloor \mathbf{A}^t \cdot \prod_{i=1}^{k} \mathbf{S}_i^{x_i} \right\rceil_p . \tag{5.1}$$

We endow $\mathcal{F}$ with the distribution where $\mathbf{A}$ is chosen uniformly at random, and below we consider a number of natural distributions for the $\mathbf{S}_i$.

The ring-based family $\mathcal{RF}$ is defined similarly to consist of functions from $\{0,1\}^k$ to R_p, where we replace $\mathbf{A}$ with uniformly random $a \in R_q$ and each $\mathbf{S}_i$ with some $s_i \in R$.

5.2 Efficiency

Consider a function $F \in \mathcal{F}$ as in Definition 5.1. Computing the function involves a subset-product of matrices. Generally speaking, matrix multi-product does not appear to be computable in TC^0 (if it were, then TC^0 would equal NC^1 [MP00]). However, in our case the matrices are known in advance (the variable input

is the subset, so it may be possible to reduce the depth of the computation via preprocessing, using ideas from [RT92]. As described in Section 4.3, both binary matrix product and rounding can be implemented with simple depth-2 arithmetic circuits, and hence in TC^0, so at worst F can be computed in TC^1 by computing the subset product in a tree-like fashion, followed by a final rounding step.

The ring variant of Construction 5.1 appears to be more efficient to evaluate, both in practice and in terms of the best theoretical depth. Consider a function $F \in \mathcal{RF}$ as in Definition 5.1. As is standard with ring-based primitives (see, e.g., [LMPR08, LPR10]), one could store the ring elements $a, s_1, \ldots, s_k$ as vectors in $\mathbb{Z}_q^n$ using the discrete Fourier transform or "Chinese remainder" representation modulo q (that is, by evaluating a and the s_i as polynomials at the n roots of $z^n + 1$ modulo q), so that multiplication of two ring elements just corresponds to a coordinate-wise product of their vectors. Then to evaluate the function, one would just compute a subset-product of the appropriate vectors, then interpolate the result to the power-basis representation, using essentially an n-dimensional Fast Fourier Transform over $\mathbb{Z}_q$, in order to perform the rounding operation. For the interesting case of $k = \omega(\log n)$, the sequential runtime of this method is dominated by the kn scalar multiplications in $\mathbb{Z}_q$ to compute the subset-product; in parallel, the arithmetic depth (over $\mathbb{Z}_q$) is $O(\log(nk))$. Alternatively, the subset-product part of the function might be computed even faster by storing the *discrete logs*, with respect to some arbitrary generator g of $\mathbb{Z}_q^*$, of the Fourier coefficients of a and s_i.[6] The subset-product then becomes a subset-sum, followed by exponentiation modulo q, or even just a table lookup if q is relatively small. Assuming that additions mod $q - 1$ are significantly less expensive than multiplications mod q, the sequential runtime of this method is dominated by the $O(n \log n)$ scalar operations in the FFT, and the parallel arithmetic depth is again $O(\log n)$.

In terms of theoretical depth, the multi-product of vectors can be performed in TC^0, as can the Fast Fourier Transform and rounding steps [RT92]. This implies that the entire function can be computed in TC^0, matching (asymptotically) the shallowest known PRFs based on the DDH and factoring problems [NR97, NRR00].

5.3 Security Proof Under LWE

Our first theorem says that when the entries of the $\mathbf{S}_i$ are "small," i.e., chosen from a suitable LWE error distribution, the degree-k construction is a PRF under a suitable LWE assumption.

Theorem 5.2. *Let $\chi = D_{\mathbb{Z},r}$ for some $r > 0$, and let $q \geq p \cdot k(Cr\sqrt{n})^k \cdot n^{\omega(1)}$ for a suitable universal constant C. Endow the family $\mathcal{F}$ from Definition 5.2 with the distribution where each $\mathbf{S}_i$ is drawn independently from $\chi^{n \times n}$. Then assuming the hardness of decision-$\mathsf{LWE}_{n,q,\chi}$, the family $\mathcal{F}$ is pseudorandom.*

An analogous theorem holds for the ring-based family $\mathcal{RF}$, under decision-RLWE.

Theorem 5.3. *Let χ be the distribution over the ring R where each coefficient (with respect to the power basis) is chosen independently from $D_{\mathbb{Z},r}$ for some $r > 0$, and let $q \geq p \cdot k(r\sqrt{n} \cdot \omega(\sqrt{\log n}))^k \cdot n^{\omega(1)}$. Endow the family $\mathcal{RF}$ from Definition 5.2 with the distribution where each s_i is drawn independently from χ. Then assuming the hardness of decision-$\mathsf{RLWE}_{n,q,\chi}$, the family $\mathcal{RF}$ is pseudorandom.*

We first prove Theorem 5.2 for the standard LWE construction.

Proof of Theorem 5.2. To aid the proof, it helps to define a family $\mathcal{G}$ of functions $G\colon \{0,1\}^k \to \mathbb{Z}_q^{n \times n}$, which are simply the unrounded counterparts of the functions in $\mathcal{F}$. That is, for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S}_i \in \mathbb{Z}^{n \times n}$ for $i \in [k]$, we define $G_{\mathbf{A},\{\mathbf{S}_i\}}(x_1 \cdots x_k) := \mathbf{A}^t \cdot \prod_{i=1}^k \mathbf{S}_i^{x_i}$. We endow $\mathcal{G}$ with the same distribution over $\mathbf{A}$ and the $\mathbf{S}_i$ as $\mathcal{F}$ has.

[6] If necessary, one would also store binary mask vectors indicating which Fourier coefficients are zero, and hence not in $\mathbb{Z}_q^*$.

We proceed via a sequence of games, much like in the proof of Theorem 3.2. First as a "thought experiment" we define a new family $\tilde{\mathcal{G}}$ of functions from $\{0,1\}^k$ to $\mathbb{Z}_q^{m \times n}$. This family is a counterpart to $\mathcal{G}$, but with two important differences: it is a PRF family *without* any rounding (and hence, with rounding as well), but each function in the family has an exponentially large key. Alternatively, one may think of the functions in $\tilde{\mathcal{G}}$ as *randomized* functions with small keys. Then we show that with overwhelming probability, the rounding of $\tilde{G} \leftarrow \tilde{\mathcal{G}}$ agrees with the rounding of the corresponding $G \in \mathcal{G}$ on all the attacker's queries, because the outputs of the two functions are relatively close. It follows that the rounding of $G \leftarrow \mathcal{G}$ (i.e., $F \leftarrow \mathcal{F}$) cannot be distinguished from a uniformly random function, as desired.

More formally, we define the following games:

Game H_0. This is the real PRF attack game against the family $\mathcal{F}$: we choose an $F \leftarrow \mathcal{F}$ (so $F(\cdot) = \lfloor G(\cdot) \rceil_p$ for $G \leftarrow \mathcal{G}$), and the attacker has oracle access to $F(\cdot)$.

Game H_1. Here we instead choose $\tilde{G} \leftarrow \tilde{\mathcal{G}}$, where the family $\tilde{\mathcal{G}}$ is given in Definition 5.4 below. The choice of $\tilde{G}$ induces a corresponding $G \in \mathcal{G}$ having the same distribution as in H_0. (This is simply because the key of G is just a portion of the key of $\tilde{G}$.) To be precise, we choose $\tilde{G}$ "lazily" as the attacker makes queries, because the description of $\tilde{G}$ has exponential size; see the remarks following Definition 5.4 for details.

The attacker has oracle access to $\lfloor \tilde{G}(\cdot) \rceil_p$, but with one exception: on query x, define the "bad event" BAD_x for that query to be

$$\left\lfloor \tilde{G}(x) + [-B, B]^{m \times n} \right\rceil_p \neq \{ \lfloor \tilde{G}(x) \rceil_p \},$$

where $B = k(Cr\sqrt{n})^k$. That is, BAD_x indicates whether any entry of $\tilde{G}(x) \in \mathbb{Z}_q^{m \times n}$ is "too close" to another element of $\mathbb{Z}_q$ that rounds to a different value in $\mathbb{Z}_p$. Note that a $y \in \mathbb{Z}_q$ is "too close" in this sense if and only if $\lfloor (\bar{y} - B) \cdot \frac{p}{q} \rceil \neq \lfloor (\bar{y} + B) \cdot \frac{p}{q} \rceil \in \mathbb{Z}$, where $\bar{y} \in \mathbb{Z}$ is any integer congruent to $y \bmod q$, so BAD_x can be efficiently detected given only the value of $\tilde{G}(x)$. If BAD_x occurs any of the attacker's queries, then the game immediately aborts.

In Lemma 5.5 below, we show that for every fixed $x \in \{0,1\}^k$, with overwhelming probability over the choice of $\tilde{G} \leftarrow \tilde{\mathcal{G}}$ and the induced $G \in \mathcal{G}$, it is the case that $G(x) \in \tilde{G}(x) + [-B, B]^{m \times n} \bmod q$. Hence $\lfloor G(x) \rceil_p = \lfloor \tilde{G}(x) \rceil_p$ so long as BAD_x does not occur, and the attacker's queries are answered exactly as they are in H_0, subject to the game not aborting. It follows that for any (potentially unbounded) attacker $\mathcal{A}$,

$$\mathbf{Adv}_{H_0,H_1}(\mathcal{A}) \leq \Pr[\text{some } \mathsf{BAD}_x \text{ occurs in } H_1 \text{ with attacker } \mathcal{A}] + \mathrm{negl}(n). \tag{5.2}$$

We do not directly bound the probability that some BAD_x occurs in H_1, but instead defer to the analysis of the next game, where we can show that it is indeed negligible.

Game H_2. Here we choose U to be a uniformly random function from $\{0,1\}^k$ to $\mathbb{Z}_q^{m \times n}$ (defined "lazily" as the attacker makes queries). The attacker has oracle access to $\lfloor U(\cdot) \rceil_p$, with the same "bad event" and abort condition as in H_1, but defined relative to U instead of $\tilde{G}$.

In Theorem 5.6 below, we show that under the LWE assumption from the theorem statement, no efficient adversary can distinguish (given oracle access) between $\tilde{G} \leftarrow \tilde{\mathcal{G}}$ and a uniformly random function U. Because the BAD_x event in H_1 (respectively, H_2) for a query x can be tested efficiently given query access to $\tilde{G}$ (resp., U), a trivial simulation implies that for any efficient attacker $\mathcal{A}$, we have $\mathbf{Adv}_{H_1,H_2}(\mathcal{A}) \leq \mathrm{negl}(n)$. For the same reasons, it also follows by a straightforward simulation that for any efficient attacker $\mathcal{A}$,

$$|\Pr[\text{some } \mathsf{BAD}_x \text{ occurs in } H_1 \text{ with } \mathcal{A}] - \Pr[\text{some } \mathsf{BAD}_x \text{ occurs in } H_2 \text{ with } \mathcal{A}]| \leq \mathrm{negl}(n).$$

In H_2, because U is a uniformly random function, for any particular query x the probability that BAD_x occurs is bounded by $(2B+1) \cdot p/q = \mathrm{negl}(n)$, by assumption on q. By a union bound over all $\mathrm{poly}(n)$ queries of an efficient $\mathcal{A}$, and then applying Equation (5.2), we therefore have that

$$\Pr[\text{some } \mathsf{BAD}_x \text{ occurs in } H_1 \text{ with } \mathcal{A}] = \mathrm{negl}(n) \quad \Rightarrow \quad \mathbf{Adv}_{H_0,H_1}(\mathcal{A}) = \mathrm{negl}(n).$$

Game H_3. Here we still choose a uniformly random function U and give the attacker oracle access to $\lfloor U(\cdot) \rceil_p$. For each query x we define the event BAD_x as in game H_2, but still answer the query and continue with the game even if BAD_x occurs. From the above analysis of H_2 it follows that for any (potentially unbounded) attacker $\mathcal{A}$ making $\mathrm{poly}(n)$ queries, we have

$$\mathbf{Adv}_{H_2,H_3}(\mathcal{A}) \leq \Pr[\text{some } \mathsf{BAD}_x \text{ occurs in } H_2 \text{ with } \mathcal{A}] = \mathrm{negl}(n).$$

Finally, observe that $\lfloor U(\cdot) \rceil_p$ is a truly random function from $\{0,1\}^k$ to $\mathbb{Z}_p^{m \times n}$, up to the bias involved in rounding the uniform distribution on $\mathbb{Z}_q$ to $\mathbb{Z}_p$. Because $q \geq p \cdot n^{\omega(1)}$, this bias is negligible (and there is no bias if p divides q).

By the triangle inequality, it follows that for any efficient $\mathcal{A}$, we have $\mathbf{Adv}_{H_0,H_3}(\mathcal{A}) = \mathrm{negl}(n)$, and this completes the proof. □

We now define the family $\tilde{\mathcal{G}}$ used in the proof of Theorem 5.2.

Definition 5.4. For parameters n, q, m, k and error distribution χ (over $\mathbb{Z}$) as in Definition 5.1, the family $\tilde{\mathcal{G}}^{(i)}$ for $0 \leq i \leq k$ is defined inductively to consist of functions from $\{0,1\}^i$ to $\mathbb{Z}_q^{m \times n}$; we define $\tilde{\mathcal{G}} = \tilde{\mathcal{G}}^{(k)}$.

- For $i = 0$, a function $\tilde{G} \in \tilde{\mathcal{G}}^{(0)}$ is indexed by some $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, and is defined simply as $\tilde{G}_{\mathbf{A}}(\varepsilon) = \mathbf{A}^t$. We endow $\tilde{\mathcal{G}}^{(0)}$ with the distribution where $\mathbf{A}$ is chosen uniformly at random.
- For $i \geq 1$, a function $\tilde{G} \in \tilde{\mathcal{G}}^{(i)}$ is indexed by some $\tilde{G}' \in \tilde{\mathcal{G}}^{(i-1)}$, plus an $\mathbf{S}_i \in \mathbb{Z}^{n \times n}$ and error matrices $\mathbf{E}_{x'} \in \mathbb{Z}^{m \times n}$ for each $x' \in \{0,1\}^{i-1}$ (where $\{0,1\}^0$ is the singleton set $\{\varepsilon\}$). For $x = (x', x_i) \in \{0,1\}^i$ where $|x'| = i-1$, the function is defined as

  $$\tilde{G}(x) = \tilde{G}_{\tilde{G}', \mathbf{S}_i, \{\mathbf{E}_{x'}\}}(x', x_i) := \tilde{G}'(x') \cdot \mathbf{S}_i^{x_i} + x_i \cdot \mathbf{E}_{x'} \bmod q.$$

  We endow $\tilde{\mathcal{G}}^{(i)}$ with the distribution where $\tilde{G}' \leftarrow \tilde{\mathcal{G}}^{(i-1)}$, and all the entries of $\mathbf{S}_i$ and every $\mathbf{E}_{x'}$ are chosen independently from χ.

Note that a function $\tilde{G} \in \tilde{\mathcal{G}}$ is fully specified by $\mathbf{A}$, $\{\mathbf{S}_i\}_{i \in [k]}$, and *exponentially* (in k) many error matrices $\mathbf{E}_{x_1 \cdots x_{i-1}}$ for all $x \in \{0,1\}^k$ and $i \in [k]$; these error matrices are what prevents $\tilde{\mathcal{G}}$ itself from being used as a PRF family. However, as needed in the proof of Theorem 5.2 (game H_1), the error matrices can be chosen "lazily," since the value of $\tilde{G}(x)$ depends only on $\mathbf{A}$, $\{\mathbf{S}_i\}$, and $\mathbf{E}_{x_1 \cdots x_{i-1}}$ for $i \in [k]$. For a function $\tilde{G} = \tilde{G}_{\mathbf{A}, \{\mathbf{S}_i\}, \{\mathbf{E}_{x'}\}} \in \tilde{\mathcal{G}}$, we define its *induced* function in the family $\mathcal{G}$ to be $G = G_{\mathbf{A}, \{\mathbf{S}_i\}}$. Note that for $\tilde{G} \leftarrow \tilde{\mathcal{G}}$, the induced function G has the same marginal distribution as if it had been chosen from $\mathcal{G}$ directly.

The following lemma is used in the analysis of game H_1.

Lemma 5.5. *Let $x \in \{0,1\}^k$ be arbitrary. Then except with $2^{-\Omega(n)}$ probability over the choice of $\tilde{G} = \tilde{G}_{\mathbf{A}, \{\mathbf{S}_i\}, \{\mathbf{E}_{x'}\}} \leftarrow \tilde{\mathcal{G}}$ and its induced function $G = G_{\mathbf{A}, \{\mathbf{S}_i\}} \in \mathcal{G}$, we have*

$$G(x) \in \tilde{G}(x) + [-B, B]^{m \times n} \bmod q$$

for some $B = k \cdot (Cr\sqrt{n})^k$, where C is a universal constant.

Proof. Observe that

$$\begin{aligned}\tilde{G}(x_1 \cdots x_k) &= (\cdots((\mathbf{A}^t \cdot \mathbf{S}_1^{x_1} + x_1 \cdot \mathbf{E}_\varepsilon) \cdot \mathbf{S}_2^{x_2} + x_2 \cdot \mathbf{E}_{x_1}) \cdots) \cdot \mathbf{S}_k^{x_k} + x_k \cdot \mathbf{E}_{x_1 \cdots x_{k-1}} \bmod q \\ &= \underbrace{\mathbf{A}^t \cdot \prod_{i=1}^{k} \mathbf{S}_i^{x_i}}_{G(x)} + x_1 \cdot \mathbf{E}_\varepsilon \cdot \prod_{i=2}^{k} \mathbf{S}_i^{x_i} + x_2 \cdot \mathbf{E}_{x_1} \cdot \prod_{i=3}^{k} \mathbf{S}_i^{x_i} + \cdots + x_k \cdot \mathbf{E}_{x_1 \cdots x_{k-1}} \bmod q.\end{aligned}$$

Now by Lemma 2.4, except with probability $2^{-\Omega(n)}$, for every $i \in [k]$ we have $s_1(\mathbf{S}_i) \le O(r\sqrt{n})$ and $\|\mathbf{e}\| \le O(r\sqrt{n})$ for every row $\mathbf{e}$ of the error matrices $\mathbf{E}_{x_1 \cdots x_{i-1}}$. Therefore, each row of the k cumulative error matrices $\mathbf{E}_{x_1 \cdots x_{i-1}} \cdot \prod_{j=i+1}^{k} \mathbf{S}_j^{x_j}$ (for $i \in [k]$) has Euclidean length at most $O(r\sqrt{n})^k$, and so its entries are bounded by the same quantity in magnitude. The claim follows. □

Theorem 5.6. *Under the* LWE *assumption from the statement of Theorem 5.2, the family $\tilde{\mathcal{G}}$ of functions from $\{0,1\}^k$ to $\mathbb{Z}_q^{m \times n}$ is pseudorandom.*

In the proof we will need the following intermediate function families.

Definition 5.7. For n, q, m, and χ as in Definition 5.1, and an integer $i \ge 1$, the family $\mathcal{H}^{(i)}$ consists of functions from $\{0,1\}^i$ to $\mathbb{Z}_q^{m \times n}$. A function H from the family is indexed by some $\mathbf{S}_i \in \mathbb{Z}^{n \times n}$ and matrices $\mathbf{A}_{x'} \in \mathbb{Z}_q^{n \times m}, \mathbf{E}_{x'} \in \mathbb{Z}^{m \times n}$ for each $x' \in \{0,1\}^{i-1}$ (where $\{0,1\}^0 = \{\varepsilon\}$). It is defined as

$$H(x) = H_{\mathbf{S}_i, \{\mathbf{A}_{x'}\}, \{\mathbf{E}_{x'}\}}(x', x_i) := \mathbf{A}_{x'}^t \cdot \mathbf{S}_i^{x_i} + x_i \cdot \mathbf{E}_{x'} \bmod q,$$

where $|x'| = i - 1$. We endow $\mathcal{H}$ with the distribution where each $\mathbf{A}_{x'}$ is uniformly random and independent, and all the entries of $\mathbf{S}_i$ and $\mathbf{E}_{x'}$ are chosen independently from χ. We remark that an $H \leftarrow \mathcal{H}^{(i)}$ can be chosen "lazily" in the natural way.

Proof of Theorem 5.6. We prove that each family $\tilde{\mathcal{G}}^{(i)}$ is pseudorandom by induction on i, from 0 to k. The base case of $i = 0$ is trivial by construction. For $i \ge 1$, we prove the claim by the following series of games.

Game H_0. We (lazily) choose a $\tilde{G} \leftarrow \tilde{\mathcal{G}}^{(i)}$ and give the attacker oracle access to $\tilde{G}(\cdot)$.

Game H_1. We (lazily) choose an $H \leftarrow \mathcal{H}^{(i)}$ (defined above) and give the attacker oracle access to $H(\cdot)$.

We claim that $H_0 \overset{c}{\approx} H_1$ under the inductive hypothesis that $\tilde{\mathcal{G}}^{(i-1)}$ is a PRF family. To prove this, we design an efficient simulator $\mathcal{S}$ that is given oracle access to a function $F\colon \{0,1\}^{i-1} \to \mathbb{Z}_q^{m \times n}$, where F is either $\tilde{G}' \leftarrow \tilde{\mathcal{G}}^{(i-1)}$ or a uniformly random function, and $\mathcal{S}$ emulates either game H_0 or H_1 (respectively) to an attacker. The simulator $\mathcal{S}$ first chooses an $\mathbf{S}_i \leftarrow \chi^{n \times n}$, and on each query $x = (x', x_i)$ from the attacker where $|x'| = i - 1$, $\mathcal{S}$ queries its oracle to get $\mathbf{A}_{x'}^t = F(x')$, chooses an $\mathbf{E}_{x'} \leftarrow \chi^{m \times n}$ (if it has not already been defined by a previous query), and returns $\mathbf{A}_{x'}^t \cdot \mathbf{S}_i^{x_i} + x_i \cdot \mathbf{E}_{x'}$ to the attacker. It is clear by the definitions of $\tilde{\mathcal{G}}^{(i)}$ and $\mathcal{H}^{(i)}$ that if F is some $\tilde{G}' \leftarrow \tilde{\mathcal{G}}^{(i-1)}$, then $\mathcal{S}$ emulates access to $\tilde{G}_{\tilde{G}', \mathbf{S}_i, \{\mathbf{E}_{x'}\}} \in \tilde{\mathcal{G}}^{(i)}$ with the appropriate distribution, whereas if F is a uniformly random function, then $\mathcal{S}$ emulates access to $H \leftarrow \mathcal{H}^{(i)}$.

Game H_2. We (lazily) choose a uniformly random function $U\colon \{0,1\}^i \to \mathbb{Z}_q^{m\times n}$ and give the attacker oracle access to $U(\cdot)$.

We claim that $H_1 \overset{c}{\approx} H_2$ under the decision-LWE assumption from Theorem 5.2. To prove this, we design an efficient simulator $\mathcal{S}$ that is given access to an oracle $\mathcal{O}$ that outputs arbitrarily many pairs $(\mathbf{A}^t, \mathbf{B}^t) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times n}$, drawn either as a group of samples $(\mathbf{A}^t, \mathbf{B}^t = \mathbf{A}^t\mathbf{S} + \mathbf{E} \bmod q)$ from the LWE distribution $A_{\mathbf{S},\chi}$ (for the same $\mathbf{S} \leftarrow \chi^{n\times n}$) or from the uniform distribution, and $\mathcal{S}$ emulates either game H_1 or H_2 (respectively) to an attacker. Under the decision-LWE assumption, this will establish the claim. The simulator $\mathcal{S}$ answers queries $x = (x', x_i)$ where $|x'| = i - 1$ in the following way: if x' has never been queried before, then it draws a new sample $(\mathbf{A}^t_{x'}, \mathbf{B}^t_{x'})$ from $\mathcal{O}$ and stores it, otherwise it looks up the already stored $(\mathbf{A}^t_{x'}, \mathbf{B}^t_{x'})$. It then returns $\mathbf{A}^t_{x'}$ if $x_i = 0$, and $\mathbf{B}^t_{x'}$ if $x_i = 1$. It is clear by inspection and the definition of $\mathcal{H}^{(i)}$ that $\mathcal{S}$ has the claimed behavior given the two types of oracles $\mathcal{O}$.

By the triangle inequality, we have $H_0 \overset{c}{\approx} H_2$, i.e., $\tilde{\mathcal{G}}^{(i)}$ is a pseudorandom function family. □

We now analyze the ring-LWE construction.

Proof sketch for Theorem 5.3. The proof proceeds almost identically to the proof of Theorem 5.2, so we only outline the few small differences. We define the function families $\mathcal{RG}$ and $\mathcal{R}\tilde{\mathcal{G}}$ in exactly the same fashion as the families $\mathcal{G}$ and $\tilde{\mathcal{G}}$, respectively, with $a \in R_q$, $s_i \in R$ and $e_{x'} \in R$ substituting $\mathbf{A}$, $\mathbf{S}_i$ and $\mathbf{E}_{x'}$ respectively. In the games, the bad event BAD_x occurs if any *coefficient* of $R\tilde{G}(x) \in R_q$ (for $R\tilde{G} \leftarrow \mathcal{R}\tilde{\mathcal{G}}$) is "too close" to another element in $\mathbb{Z}_q$ having a different rounded value, where "too close" is defined using the interval $[-B, B]$ for $B = k(r\sqrt{n} \cdot \omega(\sqrt{\log n}))^k/\sqrt{n}$. For this bound B, the analogue of Lemma 5.5 (which bounds the cumulative error terms, i.e., the difference $R\tilde{G}(x) - RG(x)$) follows immediately from Lemma 2.3. Finally, pseudorandomness of the family $\mathcal{R}\tilde{\mathcal{G}}$ follows analogously to the proof of Theorem 5.6, via families $\mathcal{RH}^{(i)}$ defined similarly to $\mathcal{H}^{(i)}$. □

Remark 5.8. By almost identical proofs, a similar subset-product-like construction

$$F_{\mathbf{A},\{\mathbf{S}_{i,b}\}}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}^t \cdot \prod_{i=1}^{k} \mathbf{S}_{i,x_i} \right\rceil_p, \tag{5.3}$$

for uniform $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and matrices $\mathbf{S}_{i,b} \in \mathbb{Z}^{n\times n}$ (for $i \in [k], b \in \{0,1\}$), and the analogous function in the ring setting, are also PRF families for the same parameters and distributions as in Theorem 5.2 and Theorem 5.3. (These functions are analogous to the factoring-based PRF of [NRR00].) While the secret keys are about twice as large as their counterparts' from Definition 5.1, these functions are more "symmetric," which may be important in practice (e.g., to prevent timing attacks).

5.4 Security Proof Under Interactive LWR

We now present an "interactive" LWR assumption and prove that under this assumption, the degree-k construction from Definition 5.1 is a PRF under an appropriate distribution of the $\mathbf{S}_i$. The advantage of this proof is that it allows us to prove security for a small modulus q and inverse error rate (both small polynomials in n), and it also works for uniformly random (or uniform invertible) matrices $\mathbf{S}_i$, among other distributions. For example, this allows us to compose the degree-k construction with itself (or with any other PRF) in a k-ary tree. The drawback to our proof is that it relies on a stronger assumption that is harder to evaluate or falsify, because it allows the adversary to make queries to its challenger.

Definition 5.9 (k-subset-product LWR)**.** Let $q \geq p$ be integer moduli. We describe a pair of games, which are parameterized by integers $k \geq 1$ and $m = \mathrm{poly}(n)$, and a distribution ψ over $\mathbb{Z}_q^{n \times n}$ (e.g., the uniform distribution). In both games, we choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random and $\mathbf{S}_i \leftarrow \psi$ independently for each $i \in [k]$, then give $\mathbf{A}$ and $\mathbf{S}_i$ for $i \in [k-1]$ to the attacker. We then allow the attacker to adaptively make queries to a function $H \colon \{0,1\}^{k-1} \to \mathbb{Z}_p^{m \times n}$. In the first game, the function H is defined to be

$$H(x) := \left\lfloor \mathbf{A}^t \cdot \prod_{i=1}^{k-1} \mathbf{S}_i^{x_i} \cdot \mathbf{S}_k \right\rceil_p ;$$

in the second game, H is a uniformly random function. The *k-subset-product* LWR *problem*, denoted k-$\mathsf{LWR}_{q,p,m,\psi}$, is to distinguish between these two games with an advantage non-negligible in n. The k-subset-product ring-LWR problem is defined analogously. (A subset-product version of (ring-)LWE is also easy to formulate, where instead of rounding we add random and independent error terms to each answer.)

We make a few simple observations about the k-LWR problem. First note that $\mathbf{B}_x^t = \mathbf{A}^t \cdot \prod_{i=1}^{k-1} \mathbf{S}_i^{x_i}$ is the part of the product that changes for each new query. Since $\mathbf{A}$ and all the $\mathbf{S}_i$ for $i \in [k-1]$ are given to the attacker, it can compute each $\mathbf{B}_x^t$ on its own, and its goal is to determine whether the challenger is returning rounded products $\lfloor \mathbf{B}_x^t \cdot \mathbf{S}_k \rceil_p$ or uniformly random and independent values. In effect, the k-LWR problem is therefore to solve LWR when the sampled $\mathbf{A}$ matrices are related by adversarially chosen subset-products of given random matrices $\mathbf{S}_i$. To avoid an efficient attack (as outlined in the introduction), the distribution ψ should be chosen so that the product of many $\mathbf{S}_i \leftarrow \psi$ does not significantly reduce the entropy of $\mathbf{A}^t \prod_i \mathbf{S}_i$. It appears that restricting ψ to invertible elements is most effective for this purpose.

We also observe that 1-$\mathsf{LWR}_{q,p,m,\psi}$ is just the standard $\mathsf{LWR}_{q,p}$ problem given m samples, where the secret matrix $\mathbf{S}$ is chosen from ψ. The problems form a hierarchy over k, that is, k-$\mathsf{LWR}_{q,p,m,\psi}$ no harder than $(k-1)$-$\mathsf{LWR}_{q,p,m,\psi}$, by a reduction that just prepends 0 to all queries, and withholds $\mathbf{S}_1$ from the attacker.

Theorem 5.10. *Endow the family $\mathcal{F}$ from Definition 5.2 with the distribution where each $\mathbf{S}_i$ is drawn from some distribution ψ. Then, assuming that k-$\mathsf{LWR}_{q,p,m,\psi}$ problem is hard, the family $\mathcal{F}$ is pseudorandom.*

Unlike our inductive proof of Theorem 5.6, which transitions from the PRF family to a random function by "dropping" the secret key components $\mathbf{S}_i$ from $i = 1$ to k, the proof of Theorem 5.10 drops them from $i = k$ down to 1. This prevents the error terms from growing with k (because the errors are not compounded by multiplication with other $\mathbf{S}_i$), which is what allows us to use a small modulus q if we so desire. However, this style of proof also seems to require an interactive assumption, so that a simulator can answer queries involving the component $\mathbf{S}_i$ that is being dropped between adjacent games.

Proof of Theorem 5.10. We prove this by induction over k. For $k = 0$, the claim follows trivially by construction. For $k \geq 1$, we again proceed via a series of games.

Game H_0. This is the real PRF attack game against the family $\mathcal{F}$: we choose an $F \leftarrow \mathcal{F}$, and the attacker has oracle access to $F(\cdot)$.

Game H_1. We choose $F \leftarrow \mathcal{F}$. For attacker queries of the form $x = x_1 \ldots x_{k-1}1$, we return uniformly random and independent value (consistent with prior answers), and for queries of the form $x = x_1 \ldots x_{k-1}0$, we return $F(x) = \left\lfloor \mathbf{A}^t \cdot \prod_{i=1}^{k-1} \mathbf{S}_i^{x_i} \right\rceil_p$.

We claim that $H_0 \overset{c}{\approx} H_1$ by a straightforward reduction assuming the hardness of k-LWR. As proof, we construction a simulator $\mathcal{S}$ that interacts with an oracle $\mathcal{O}$ that implements one of the two games from the k-LWR problem, and emulates either H_0 or H_1 respectively. The simulator is first given some matrices $\mathbf{A}$ and $\mathbf{S}_i$ for $i \in [k-1]$. It then answers attacker queries $x = (x', 0) \in \{0,1\}^k$ by returning $\lfloor \mathbf{A}^t \cdot \prod_{i=1}^{k-1} \mathbf{S}_i^{x_i} \rceil_p$, and answers queries $x = (x', 1) \in \{0,1\}^k$ by returning $\mathcal{O}(x')$ to the attacker. It is clear by inspection that the behavior of $\mathcal{S}$ is as claimed.

Game H_2. We lazily choose a uniformly random function $U\colon \{0,1\}^k \to \mathbb{Z}_p^{m \times n}$ and give the attacker oracle access to $U(\cdot)$.

We claim that $H_1 \overset{c}{\approx} H_2$ by the inductive hypothesis. This is because in game H_1, queries ending in 1 are already answered uniformly, while queries ending in 0 are answered according to a function drawn from the family $\mathcal{F}$ of degree $(k-1)$. This family is pseudorandom by the inductive hypothesis, and the fact that $(k-1)$-LWR is no easier than k-LWR.

This completes the induction and the proof. □

Acknowledgments. We thank Oded Regev for interesting discussions.

References

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115. 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.

[AG11] S. Arora and R. Ge. New algorithms for learning in presence of errors. In *ICALP (1)*, pages 403–415. 2011.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Ban95] W. Banaszczyk. Inequalites for convex bodies and polar reciprocal lattices in R^n. *Discrete & Computational Geometry*, 13:217–231, 1995.

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/`.

[BKW03] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[BMR10] D. Boneh, H. W. Montgomery, and A. Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *ACM Conference on Computer and Communications Security*, pages 131–140. 2010.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*. 2011. To appear.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Preliminary version in FOCS 1984.

[GKPV10] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. 2010.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[HB01] N. J. Hopper and M. Blum. Secure human identification protocols. In *ASIACRYPT*, pages 52–66. 2001.

[JW05] A. Juels and S. A. Weis. Authenticating pervasive devices with human protocols. In *CRYPTO*, pages 293–308. 2005.

[KPC+11] E. Kiltz, K. Pietrzak, D. Cash, A. Jain, and D. Venturi. Efficient authentication from hard learning problems. In *EUROCRYPT*, pages 7–26. 2011.

[KSS06] J. Katz, J. S. Shin, and A. Smith. Parallel and concurrent security of the HB and HB$^+$ protocols. *J. Cryptology*, 23(3):402–421, 2010. Preliminary version in Eurocrypt 2006.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72. 2008.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23. 2010.

[LW09] A. B. Lewko and B. Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *ACM Conference on Computer and Communications Security*, pages 112–120. 2009.

[MM11] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484. 2011.

[MP00] C. Mereghetti and B. Palano. Threshold circuits for iterated matrix product and powering. *ITA*, 34(1):39–46, 2000.

[MP11] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller, 2011. Manuscript.

[NR95] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999. Preliminary version in FOCS 1995.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. Preliminary version in FOCS 1997.

[NRR00] M. Naor, O. Reingold, and A. Rosen. Pseudorandom functions and factoring. *SIAM J. Comput.*, 31(5):1383–1404, 2002. Preliminary version in STOC 2000.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[Pie10] K. Pietrzak. Subspace LWE, 2010. Manuscript. Last retrieved from `http://homepages.cwi.nl/~pietrzak/publications/SLWE.pdf` on 28 June 2011.

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196. 2008.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[RT92] J. H. Reif and S. R. Tate. On threshold circuits and polynomial computation. *SIAM J. Comput.*, 21(5):896–908, 1992.

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

[Ver11] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, January 2011. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`, last accessed 4 Feb 2011.

[vGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43. 2010.

Field Switching in BGV-Style Homomorphic Encryption

Craig Gentry
IBM Research

Shai Halevi
IBM Research

Chris Peikert
Georgia Institute of Technology

Nigel P. Smart
University of Bristol

September 13, 2013

Abstract

The security of contemporary homomorphic encryption schemes over cyclotomic number field relies on fields of very large dimension. This large dimension is needed because of the large modulus-to-noise ratio in the key-switching matrices that are used for the top few levels of the evaluated circuit. However, a smaller modulus-to-noise ratio is used in lower levels of the circuit, so from a security standpoint it is permissible to switch to lower-dimension fields, thus speeding up the homomorphic operations for the lower levels of the circuit. However, implementing such field-switching is nontrivial, since these schemes rely on the field algebraic structure for their homomorphic properties.

A basic ring-switching operation was used by Brakerski, Gentry and Vaikuntanathan, over rings of the form $\mathbb{Z}[X]/(X^{2^n}+1)$, in the context of bootstrapping. In this work we generalize and extend this technique to work over any cyclotomic number field, and show how it can be used not only for bootstrapping but also during the computation itself (in conjunction with the "packed ciphertext" techniques of Gentry, Halevi and Smart).

1 Introduction

The last few years have seen a rapid advance in the state of fully homomorphic encryption, yet despite these advances, the existing schemes are still too expensive for many practical purposes. In this paper we make another step forward in making such schemes more efficient. In particular, we present a technique for reducing the dimension of the ciphertexts involved in the homomorphic computation of the lower levels of a circuit. Our techniques apply to homomorphic encryption schemes over number fields, such as the schemes of Brakerski et al. [4, 5, 3], as well as the variants due to López-Alt et al. [14] and Brakerski [2].

The most efficient variants of these schemes work over number fields of the form $\mathbb{Q}(\zeta) \cong \mathbb{Q}[X]/F(X)$, and in all of them the field dimension n, which is the degree of $F(X)$, must be set large enough to ensure security: to support homomorphic evaluation of depth-L circuits with security parameter λ, the schemes require $n = \tilde{\Omega}(L \cdot \mathrm{polylog}(\lambda))$, even under the strongest plausible hardness assumptions for their underlying computational problems (e.g., ring-LWE [15]).[1] In practice, the field dimension for moderately deep circuits can easily be many thousands. For example, to be able to evaluate AES homomorphically, Gentry et al. [13] used circuits of depth $L \geq 50$, with a corresponding field dimension of over $50{,}000$.

[1] The schemes from [3, 2] can also obtain security by using high-dimensional vectors over low-dimensional number fields. But their most efficient variants use low-dimensional vectors over high-dimensional fields, since the runtime of certain operations is cubic in the dimension of the vectors.

As homomorphic operations are performed, the ratio of noise to modulus in the ciphertexts grows. Consequently, it becomes permissible to use lower-dimension fields, which can speed up further homomorphic computations. However, since we must start with ciphertexts from a high-dimensional field, we need a method for transforming them into small-field ciphertexts that encrypt the same (or related) messages. Such a "field switching" procedure was described by Brakerski et al. [3], in the context of reducing the ciphertext size prior to bootstrapping. The procedure in [3], however, is specific to number fields of the form $K_{2^k} = \mathbb{Q}[X]/(X^{2^{k-1}} + 1)$, i.e., cyclotomic number fields with power-of-2 index. Moreover, by itself it cannot be combined with the "packed evaluation" techniques from [18, 11]. (These techniques use Chinese-remainder encoding to "pack" many plaintext values into each ciphertext, and then each homomorphic operation is applied to all these values at once. For our purposes, we must consider the effect of the field-switching operation on all these plaintext values.) Extending and improving the field switching procedure is the goal of our work.

1.1 Our Contribution

We present a general field-switching transformation that can be applied to any *cyclotomic* number field $K = \mathbb{Q}(\zeta_m) \cong \mathbb{Q}[X]/\Phi_m(X)$ for arbitrary m (where $\Phi_m(X) \in \mathbb{Z}[X]$ is the mth cyclotomic polynomial), and works well in conjunction with packed ciphertexts. For any divisor m' of m, our procedure takes as input a "big-field ciphertext" c over K that encrypts many plaintext values, and outputs a "small-field ciphertext" c' over $K' = \mathbb{Q}(\zeta_{m'}) \cong \mathbb{Q}[X]/\Phi_{m'}(X) \subseteq K$ that encrypts a certain subset of the input plaintext values.[2]

Our transformation relies heavily on the algebraic properties of the cyclotomic number fields K, K' and their respective rings of algebraic integers R, R'. In particular, we use the interpretation of K as an extension field of K', and relationships between their various embeddings into the complex numbers $\mathbb{C}$; the factorization of integer primes in R and R'; and the *trace function* $\mathrm{Tr}_{K/K'}$ that maps elements in K to the subfield K'. With these tools in hand, the transformation itself is quite simple, and consists of the following three steps:

1. We first apply a key-switching operation to obtain a big-field ciphertext over K with respect to a small-field secret key $s' \in K' \subset K$. Proving the security of this operation relies on a novel way of embedding the ring-LWE problem over K' into K, which may be of independent interest.

2. Next, we multiply the resulting ciphertext by a certain element of the ring $R \subset K$, which depends only on the subset (or other function) of the plaintext values that we want to include in the output ciphertext.

3. Finally, we take the trace of the K-elements in the ciphertext, thus obtaining an output ciphertext over the subfield K', which decrypts under the secret key $s' \in K'$ to the desired plaintext values.

We note that in addition to being simpler and more general than the transformation from [3], our transformation is also more efficient even when applied in the special case of K_{2^k}: when switching from K_{2^k} to $K_{2^{k'}}$, the transformation from [3] includes a step where the size of the ciphertext (and hence the time that it takes to perform operations) is expanded by a factor of $2^{k-k'}$. Our transformation does not need that extra step, hence saving this extra factor in performance.

In Section 2 below we recall the algebraic concepts needed for our transformation, and then the transformation itself it described in Section 3.

[2] More generally, the output ciphertext can even encrypt certain linear functions of the input plaintext values.

Notations	Description
p, $\mathbb{F}_{p^d}$	The (prime) modulus of the cryptosystem's native plaintext space, and the finite field of order p^d.
m, m', $n = \varphi(m), n' = \varphi(m')$	The indices of the cyclotomic fields, where $m'\|m$. We switch from the mth to the m'th cyclotomic number field, which are of degree n, n' (respectively) over the rationals.
$\bar{m},\ d,\ e,\ f$, $\bar{m}', d', e', f'$	$\bar{m}$ is the largest divisor of m that is coprime with p; d is the order of p in $\mathbb{Z}^*_{\bar{m}}$; $e = \varphi(m)/\varphi(\bar{m})$; and $f = \varphi(\bar{m})/d$. Similarly for $\bar{m}', d', e', f'$.
ζ_m, $\zeta_{m'}$	Abstract elements of order m, m' (respectively) over the rationals.
$K = \mathbb{Q}(\zeta_m)$, $K' = \mathbb{Q}(\zeta_{m'})$, $R = \mathbb{Z}[\zeta_m]$, $R' = \mathbb{Z}[\zeta_{m'}]$	The cyclotomic number fields and their rings of integers.
$\sigma\colon K \to \mathbb{C}^n$ $\sigma'\colon K' \to \mathbb{C}^{n'}$	The canonical embeddings of K, K', which endow the number fields with a geometry.
$\mathrm{Tr}_{K/K'}\colon K \to K'$	The trace function, which is the sum of the automorphisms of K that fix K' pointwise.
$R^\vee, (R')^\vee$	The codifferent (or dual) fractional ideals of R and R' (respectively), defined as $R^\vee = \{a : \mathrm{Tr}_{K/\mathbb{Q}}(aR) \subseteq \mathbb{Z}\}$ and similarly for $(R')^\vee$.
$G = \mathbb{Z}^*_{\bar{m}}/\langle p\rangle$, $G' = \mathbb{Z}^*_{\bar{m}'}/\langle p\rangle$	The multiplicative quotient groups that characterize the prime-ideal factorizations of pR, pR', respectively.
$g\colon G \to G'$	The (f/f')-to-1 homomorphism defined via $i \mapsto i \bmod \bar{m}'$.

Table 1: Summary of the main algebraic notations.

2 Preliminaries

This work uses a number of algebraic concepts and notations; to assist the reader we summarize the most important ones in Table 1. For any positive integer u we let $[u] = \{0, \ldots, u-1\}$. Throughout this work, for a coset $z \in \mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ we let $[z]_q \in \mathbb{Z}$ denote its canonical representative in $\mathbb{Z} \cap [-q/2, q/2)$. One can also view $[\cdot]_q$ as the operation that takes an arbitrary integer z and reduces it modulo q into the interval $[-q/2, q/2)$.

2.1 Algebraic Background

Recall that an *ideal* I in a commutative ring R is a nontrivial (i.e., $I \neq \emptyset$ and $I \neq \{0\}$) additive subgroup which is closed under multiplication by R. For ideals I, J, their sum is the ideal $I + J = \{a + b : a \in I, b \in J\}$, and their product IJ is the ideal consisting of all *sums* of terms ab for $a \in I, b \in J$. An R-ideal $\mathfrak{p}$ is prime if $ab \in \mathfrak{p}$ (for some $a, b \in R$) implies $a \in \mathfrak{p}$ or $b \in \mathfrak{p}$ (or both). All the rings we work with have unique factorization of ideals into powers of prime ideals, and a Chinese Remainder Theorem.

A *fractional ideal* is, informally, an ideal with a denominator. Formally, letting K be the field of fractions of R, a fractional ideal of R is a subset $I \subseteq K$ for which there exists a denominator $d \in R$ such that $dI \subseteq R$ is an ideal in R. For an R-ideal I, the quotient ring $R_I = R/I$ consists of the residue classes $a + I$ for all

$a \in R$, with the ring operations induced by R. More generally, for a (possibly fractional) ideal I and an ideal $J \subseteq R$, the quotient $I_J = I/IJ$ is an additive group, and an R-module, with addition and multiplication operations induced by R. We often write $a \bmod I$ instead of $a + I$ to denote the residue classes $a + I$, and we write $a = b \pmod I$ to denote that a, b belong to the same residue class, i.e., $a + I = b + I$.

For computational purposes, all of the rings and fields we work with have efficient representations of their elements, and efficient (i.e., polynomial time in the bit length of the arguments) algorithms for all the operations we use. For quotients A/B, cosets are represented using a fixed set of distinguished representatives. In this work we largely ignore the details of concrete representations and algorithms, and refer to [16] for fast, specialized algorithms for working with the cyclotomic fields and rings that we use in this work.

2.1.1 Cyclotomic Fields and Rings

For a positive integer m, let $K = \mathbb{Q}(\zeta_m)$ be the mth *cyclotomic number field*, where ζ_m is an abstract element of order m. (In particular, we do not view ζ_m as any particular root of unity in $\mathbb{C}$.) The minimal polynomial of ζ_m is the mth *cyclotomic polynomial* $\Phi_m(X) = \prod_{i \in \mathbb{Z}_m^*} (X - \eta_m^i) \in \mathbb{Z}[X]$, where $\eta_m = \exp(2\pi\sqrt{-1}/m) \in \mathbb{C}$ is the principal mth complex root of unity, and the roots $\eta_m^i \in \mathbb{C}$ range over all the *primitive* complex mth roots of unity. Therefore, K is a field extension of degree $n = \varphi(m)$ over $\mathbb{Q}$, and is isomorphic to the polynomial ring $\mathbb{Q}[X]/\Phi_m(X)$ by identifying ζ_m with X. (There are other representations of K as well, and nothing in this work depends on a particular choice of representation.) The *ring of (algebraic) integers* in K, called the mth cyclotomic ring, is $R = \mathbb{Z}[\zeta_m]$, which is isomorphic to $\mathbb{Z}[X]/\Phi_m(X)$.

The field extension $K/\mathbb{Q}$ has n automorphisms $\tau_i \colon K \to K$ that fix $\mathbb{Q}$ pointwise, which are characterized by $\tau_i(\zeta_m) = \zeta_m^i$ for $i \in \mathbb{Z}_m^*$. (Equivalently, $\tau_i(a(X)) = a(X^i) \bmod \Phi_m(X)$ when viewing K as $\mathbb{Q}[X]/\Phi_m(X)$.) Because $K/\mathbb{Q}$ is Galois (i.e., the number of automorphisms equals the dimension of the extension), the $\mathbb{Q}$-linear[3] *(field) trace* $\mathrm{Tr}_{K/\mathbb{Q}} \colon K \to \mathbb{Q}$ can be defined as the sum of the automorphisms: $\mathrm{Tr}_{K/\mathbb{Q}}(a) = \sum_{i \in \mathbb{Z}_m^*} \tau_i(a) \in \mathbb{Q}$. (See below for another formulation.)

Similarly to the automorphisms τ_i (which map K to itself), there are n concrete ways of viewing K as a subfield of the complex numbers $\mathbb{C}$. Namely, there are n injective ring homomorphisms from K to $\mathbb{C}$ that fix $\mathbb{Q}$ pointwise, called *embeddings*, which are denoted $\sigma_i \colon K \to \mathbb{C}$ for $i \in \mathbb{Z}_m^*$ and characterized by $\sigma_i(\zeta_m) = \eta_m^i$. The embeddings may be seen as the compositions of the abstract automorphisms τ_i with the complex embedding that identifies $\zeta_m \in K$ with $\eta_m \in \mathbb{C}$. Therefore, the field trace can also be written as the sum of the embeddings, as $\mathrm{Tr}_{K/\mathbb{Q}}(a) = \sum_{i \in \mathbb{Z}_m^*} \sigma_i(a) \in \mathbb{Q}$. The *canonical embedding* $\sigma \colon K \to \mathbb{C}^n$ is the concatenation of all the complex embeddings, i.e., $\sigma(a) = (\sigma_i(a))_{i \in \mathbb{Z}_m^*}$, and it endows K with a canonical geometry. In particular, define the Euclidean (ℓ_2) and ℓ_∞ norms on K as

$$\|a\| := \|\sigma(a)\| = \sqrt{\sum_i |\sigma_i(a)|^2} \quad \text{and} \quad \|a\|_\infty := \|\sigma(a)\|_\infty = \max_i |\sigma_i(a)|,$$

respectively. Note that $\|a \cdot b\| \leq \|a\|_\infty \cdot \|b\|$ and $\|a \cdot b\|_\infty \leq \|a\|_\infty \cdot \|b\|_\infty$ for any $a, b \in K$, because the σ_i are ring homomorphisms.

2.1.2 Towers of Cyclotomics

For any positive integer m' dividing m, let $K' = \mathbb{Q}(\zeta_{m'})$ and $R' = \mathbb{Z}[\zeta_{m'}]$ be the m'th cyclotomic field and ring (of dimension $n' = \varphi(m')$ over $\mathbb{Q}$ and $\mathbb{Z}$), respectively. As above, the field extension $K'/\mathbb{Q}$ has

[3] A function f is S-linear if $f(a + b) = f(a) + f(b)$ and $f(s \cdot a) = s \cdot f(a)$ for all $s \in S$ and all a, b.

$n' = \varphi(m')$ automorphisms $\tau'_{i'} \colon K' \to K'$ and n' complex embeddings $\sigma'_{i'} \colon K' \to \mathbb{C}$ (for $i' \in \mathbb{Z}^*_{m'}$), the latter of which define the canonical embedding $\sigma' \colon K' \to \mathbb{C}^{n'}$.

We will use extensively the fact that K is a field extension of K', and R is a ring extension of R', both of dimension n/n' (because $K/\mathbb{Q}$ and $K'/\mathbb{Q}$ have dimensions n and n', respectively). That is, K' and R' may respectively be seen as a subfield of $K = K'(\zeta_m)$ and a subring of $R = R'[\zeta_m]$, under the ring embedding that identifies $\zeta_{m'}$ with $\zeta_m^{m/m'}$. Moreover, the field extension K/K' is Galois, i.e., it has n/n' automorphisms that fix K' pointwise, which are precisely those τ_i for which $i = 1 \pmod{m'}$. This follows from the fact that

$$\tau_i(\zeta_{m'}) = \tau_i(\zeta_m^{m/m'}) = \zeta_m^{(m/m')i \bmod m} = \zeta_{m'}^{i \bmod m'}, \tag{2.1}$$

and that reducing modulo m' induces an (n/n')-to-1 mapping from $\mathbb{Z}^*_m$ to $\mathbb{Z}^*_{m'}$. The K'-linear (intermediate) trace function $\mathrm{Tr}_{K/K'} \colon K \to K'$ may be defined as the sum of these automorphisms:

$$\mathrm{Tr}_{K/K'}(a) = \sum_{i=1 \pmod{m'}} \tau_i(a).$$

A standard fact from field theory is that the intermediate trace satisfies $\mathrm{Tr}_{K/\mathbb{Q}} = \mathrm{Tr}_{K'/\mathbb{Q}} \circ \mathrm{Tr}_{K/K'}$. Another standard fact is that $\mathrm{Tr}_{K/K'}$ is a "universal" K'-linear function, in that any such function $L \colon K \to K'$ can be expressed as $L(a) = \mathrm{Tr}_{K/K'}(r \cdot a)$ for some fixed $r \in K$.

Similarly to Equation (2.1), for any $i \in \mathbb{Z}^*_m$ the embedding σ_i coincides with $\sigma'_{i \bmod m'}$ on the subfield K'. Using this fact we get the following relation between the intermediate trace and the complex embeddings of K and K'.

Lemma 2.1. *For any $a \in K$ and $i' \in \mathbb{Z}^*_{m'}$,*

$$\sigma'_{i'}(\mathrm{Tr}_{K/K'}(a)) = \sum_{i=i' \pmod{m'}} \sigma_i(a).$$

In matrix form, $\sigma'(\mathrm{Tr}_{K/K'}(a)) = P \cdot \sigma(a)$, where P is the $\varphi(m')$-by-$\varphi(m)$ matrix (with rows indexed by $i' \in \mathbb{Z}^*_{m'}$ and columns by $i \in \mathbb{Z}^*_m$) whose (i', i)th entry is 1 if $i = i' \pmod{m'}$, and is 0 otherwise.

Proof. Fix an arbitrary $k \in \mathbb{Z}^*_m$ such that $k = i' \pmod{m'}$. Then because $\sigma'_{i'}$ coincides with σ_k on K', and by definition of $\mathrm{Tr}_{K/K'}$ and linearity of σ_k, we have

$$\begin{aligned}\sigma'_{i'}(\mathrm{Tr}_{K/K'}(a)) &= \sigma_k\Bigg(\sum_{j=1 \pmod{m'}} \tau_j(a)\Bigg) \\ &= \sum_{j=1 \pmod{m'}} \sigma_k(\tau_j(a)) = \sum_{i=i' \pmod{m'}} \sigma_i(a),\end{aligned}$$

where for the last equality we have used $\sigma_k \circ \tau_j = \sigma_{k \cdot j}$ and $k \in \mathbb{Z}^*_m$, so $i = k \cdot j \in \mathbb{Z}^*_m$ runs over all indices congruent to i' modulo m' when $j \in \mathbb{Z}^*_m$ runs over all indices congruent to 1 modulo m'. □

An immediate corollary is that the intermediate trace maps short elements of K to short elements of K'.

Corollary 2.2. *For any $a \in K$, we have $\|\mathrm{Tr}_{K/K'}(a)\| \le \|a\| \cdot \sqrt{n/n'}$.*

Proof. By Lemma 2.1, we have $\sigma'(\mathrm{Tr}_{K/K'}(a)) = P \cdot \sigma(a)$. The rows of P are orthogonal (since each column of P has exactly one nonzero entry), and each has Euclidean norm exactly $\sqrt{n/n'}$. □

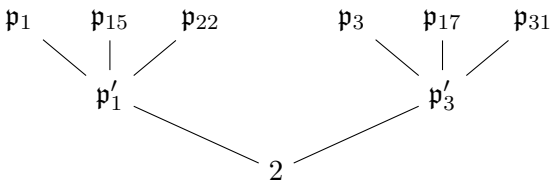


Figure 1: Factorization of $2 \in \mathbb{Z}$ into distinct prime ideals $\mathfrak{p}'_{i'}$ in $R' = \mathbb{Z}[\zeta_7]$, and $\mathfrak{p}_i$ in $R = \mathbb{Z}[\zeta_{91}]$. The displayed subscripts indicate a choice of representatives from the cosets of the multiplicative subgroups $\langle 2 \rangle \subseteq \mathbb{Z}_7^*$ and $\langle 2 \rangle \subseteq \mathbb{Z}_{91}^*$, which have orders $d' = 3$ and $d = 12$, respectively.

2.1.3 Prime Splitting and Plaintext Arithmetic

We now describe the factorization ("splitting") of prime integers in cyclotomic rings, how it allows for encoding and operating on several finite-field elements, and the particular functions induced by the (intermediate) trace function $\mathrm{Tr}_{K/K'}$. Further details and proofs can be found in many texts on algebraic number theory, e.g., [19].

Prime splitting. Let $p \in \mathbb{Z}$ be a prime integer. In the mth cyclotomic ring $R = \mathbb{Z}[\zeta_m]$ (which has degree $n = \varphi(m)$ over $\mathbb{Z}$), pR is often not a prime ideal, but instead factors into prime ideals. To describe how, we first need to introduce some notation. Divide out all the factors of p from m, writing $m = \bar{m} \cdot p^k$ where $p \nmid \bar{m}$. Let $e = \varphi(p^k)$, and let d be the multiplicative order of p modulo $\bar{m}$ (i.e., in $\mathbb{Z}_{\bar{m}}^*$); note that d divides $\varphi(\bar{m}) = n/e$. (The values d, e are respectively called the *inertial degree* and *ramification index* of p in R.) Let $G = \mathbb{Z}_{\bar{m}}^*/\langle p \rangle$, the multiplicative quotient group $\mathbb{Z}_{\bar{m}}^*$ modulo the order-d subgroup generated by p, so G has order $f = \varphi(\bar{m})/d = n/(de)$. For an element $i \in G$ of this group, we sometimes write $i\langle p \rangle$ to emphasize that it is a coset, and (slightly abusing notation) also let $i \in \mathbb{Z}_{\bar{m}}^*$ denote some element of the coset. The ideal pR factors as

$$pR = \prod_{i \in G} \mathfrak{p}_i^e, \tag{2.2}$$

where the $\mathfrak{p}_i$ are distinct prime ideals in R, all having norm $|R/\mathfrak{p}_i| = p^d$. These are called the prime ideals *lying over* p in R. Each quotient ring $R/\mathfrak{p}_i$ is therefore isomorphic to the finite field $\mathbb{F}_{p^d}$. (In fact there are exactly d isomorphisms between them, because $\mathbb{F}_{p^d}$ has d automorphisms.)

Concretely, the prime ideals $\mathfrak{p}_i$, and the isomorphisms between $R/\mathfrak{p}_i$ and (some canonical representation of) $\mathbb{F}_{p^d}$, are as follows. Let $\omega_{\bar{m}}$ denote some arbitrary element of order $\bar{m}$ in $\mathbb{F}_{p^d}$; such an element exists because the multiplicative group $\mathbb{F}_{p^d}^*$ is cyclic and has order $p^d - 1 = 0 \pmod{\bar{m}}$. For any $i\langle p \rangle \in G$, the prime ideal $\mathfrak{p}_i$ is the kernel of the ring homomorphism $h_i \colon R \to \mathbb{F}_{p^d}$ defined by $h_i(\zeta_m) = \omega_{\bar{m}}^i$. It is immediate that this kernel is an ideal; furthermore, it is invariant under the choice of representative i from the coset $i\langle p \rangle$, because $h_{ip}(r) = h_i(r)^p$ for any $r \in R$ (since $(a+b)^p = a^p + b^p$ for any $a, b \in \mathbb{F}_{p^d}$). Because $\mathfrak{p}_i$ is the kernel of h_i, we have the induced isomorphism $h_i \colon R/\mathfrak{p}_i \to \mathbb{F}_{p^d}$; indeed, we have d distinct such isomorphisms, one for each element of the coset $i\langle p \rangle$.

Looking ahead, the isomorphisms h_i (for appropriate choices of representatives i) will be used to define several "plaintext slots" in a homomorphic cryptosystem, i.e., an encoding of f plaintext elements of $\mathbb{F}_{p^d}$ as a single element of the cryptosystem's plaintext ring $R/2R$.

Splitting in cyclotomic towers. Of course, the above derivation also applies to the ideals that lie over p in $R' = \mathbb{Z}[\zeta_{m'}] \subseteq R$. For each such ideal $\mathfrak{p}'$, we next describe the factorization of $\mathfrak{p}'R$ into prime ideals in R. These are the prime ideals that lie over $\mathfrak{p}'$ in R, and since "lying over" is an associative property, they also lie over p (as illustrated in Figure 1).

Let $\bar{m}, d, e, f, G$ and the prime ideals $\mathfrak{p}_i$ for $i \in G$ be as above for R, and define $\bar{m}', d', e', f', G' = \mathbb{Z}^*_{\bar{m}'}/\langle p \rangle$ and prime ideals $\mathfrak{p}'_{i'}$ for $i' \in G'$ similarly for R'. Note that $d'|d$, $e'|e$, and $f'|f$, and that the natural homomorphism $g\colon G \to G'$ defined via $i \mapsto i \bmod \bar{m}'$ is surjective and (f/f')-to-1. Then for every $i' \in G'$, the factorization of $\mathfrak{p}'_{i'}R$ is

$$\mathfrak{p}'_{i'}R = \prod_{i \in g^{-1}(i')} \mathfrak{p}_i^{e/e'} = \prod_{i = i' \pmod{\bar{m}}} \mathfrak{p}_i^{e/e'}.$$

Therefore, there are f/f' prime ideals of R lying over each $\mathfrak{p}'_{i'}$, and taken over all $i' \in G'$ they partition the prime ideals of R lying over p.

Plaintext encoding. Let $\mathbb{F} = \mathbb{F}_{p^d}$ and $\mathbb{F}' = \mathbb{F}_{p^{d'}} \subseteq \mathbb{F}$. By the above and the Chinese Remainder Theorem, the natural ring homomorphisms yield the following (where $\cong$ denotes a ring isomorphism):

$$R'/pR' \to R'/\Big(\prod_{i' \in G'} \mathfrak{p}'_{i'}\Big) \cong \bigoplus_{i' \in G'} R'/\mathfrak{p}'_{i'} \cong \mathbb{F}'^{f'}$$
$$R/pR \to R/\Big(\prod_{i \in G} \mathfrak{p}_i\Big) = R/\Big(\prod_{i' \in G'} \prod_{i \in g^{-1}(i')} \mathfrak{p}_i\Big) \cong \bigoplus_{i' \in G'} \bigoplus_{i \in g^{-1}(i')} R/\mathfrak{p}_i \cong (\mathbb{F}^{f/f'})^{f'}.$$

(Note that the first homomorphism in each line is surjective, but not necessarily an isomorphism, due to possible ramification.) Following [18, 3, 11, 12, 13], in the context of homomorphic encryption the above morphisms allow for encoding a vector of f' individual elements of $\mathbb{F}'$ (respectively, f elements of $\mathbb{F}$) into the plaintext ring $R'_p = R/pR'$ (resp., $R_p = R/pR$), so that a single homomorphic addition and multiplication acts component-wise on the underlying vectors of field elements.

Trace operations. As mentioned in the introduction, our field-switching technique is built around applying the trace function $\mathrm{Tr}_{K/K'}$ to the elements of a big-field ciphertext, thus obtaining a related small-field ciphertext. Since we use "packed" ciphertexts that encrypt arrays of elements in $\mathbb{F}$ via the above isomorphisms, we need to understand the effect of the trace function on those $\mathbb{F}$-elements.

The remainder of this subsection is therefore devoted to characterizing the functions $(\mathbb{F}^{f/f'})^{f'} \to \mathbb{F}'^{f'}$ that can be induced by $\mathrm{Tr}_{K/K'}$. More specifically, we determine exactly which functions

$$L\colon R/\Big(\prod_{i \in G} \mathfrak{p}_i\Big) \to R'/\Big(\prod_{i' \in G'} \mathfrak{p}'_{i'}\Big)$$

can be expressed as $L(a) = \mathrm{Tr}_{K/K'}(r \cdot a)$ for some fixed $r \in K$. It turns out that by fixing an appropriate choice of isomorphisms between the quotient rings and finite fields above, we can obtain the *concatenation* of any f' individual $\mathbb{F}'$-linear functions $\mathbb{F}^{f/f'} \to \mathbb{F}'$ (see Corollary 2.5 for a precise statement).[4]

As already noted, the isomorphisms between the quotient rings and finite fields are not necessarily unique; they are determined by the choice of representatives i', i of the cosets $i'\langle p \rangle \subseteq \mathbb{Z}^*_{\bar{m}'}$ and $i\langle p \rangle \subseteq \mathbb{Z}^*_{\bar{m}}$ (respectively), and roots of unity $\omega_{\bar{m}'} \in \mathbb{F}'$ and $\omega_{\bar{m}} \in \mathbb{F}$. For our purposes, it is important to choose

[4]Note that any $\mathbb{F}'$-linear function $L\colon \mathbb{F}^{f/f'} \to \mathbb{F}'$ can always be expressed as $L(\vec{a}) = \mathrm{Tr}_{\mathbb{F}/\mathbb{F}'}(\langle \vec{d}, \vec{a} \rangle)$ for some fixed $\vec{d} \in \mathbb{F}^{f/f'}$, where $\langle \cdot, \cdot \rangle$ is the usual inner product and $\mathrm{Tr}_{\mathbb{F}/\mathbb{F}'}$ denotes the ($\mathbb{F}'$-linear) trace of the field extension $\mathbb{F}/\mathbb{F}'$.

these in a "consistent" fashion, as follows. First, given $\omega_{\bar{m}}$, let $\omega_{\bar{m}'} = \omega_{\bar{m}}^{\bar{m}/\bar{m}'} \in \mathbb{F}'$. (Note that all $\varphi(\bar{m}')$ elements of order $\bar{m}'$ in $\mathbb{F}$ are indeed in the subfield $\mathbb{F}'$.) Next, let $\ell \geq 0$ be the integer exponent such that $m/m' = (\bar{m}/\bar{m}') \cdot p^\ell$. Then given representative i' of $i'\langle p \rangle \in G'$, choose representative i for each $i\langle p \rangle \in g^{-1}(i')$ so that $p^\ell \cdot i = i' \pmod{\bar{m}'}$. Note that such i always exists, by definition of the quotient group G and the mapping g. As explained above, these choices fix particular isomorphisms

$$h_i \colon R/\mathfrak{p}_i \to \mathbb{F} \text{ (for } i\langle p \rangle \in G) \quad \text{and} \quad h'_{i'} \colon R'/\mathfrak{p}'_{i'} \to \mathbb{F}' \text{ (for } i'\langle p \rangle \in G'),$$

which are characterized by $h_i(\zeta_m) = \omega_{\bar{m}}^i$ and $h'_{i'}(\zeta_{m'}) = \omega_{\bar{m}'}^{i'}$.

Next, for each $i' \in G'$ denote the product of prime ideals lying over $\mathfrak{p}'_{i'}$ in R (called the *radical* of $\mathfrak{p}'_{i'}R$) by $\tilde{\mathfrak{p}}_{i'} = \prod_{i \in g^{-1}(i')} \mathfrak{p}_i$, and define the ring isomorphism

$$\tilde{h}_{i'} \colon R/\tilde{\mathfrak{p}}_{i'} \to \mathbb{F}^{f/f'}, \quad \tilde{h}_{i'}(a) = \big(h_i(a \bmod \mathfrak{p}_i)\big)_{i \in g^{-1}(i')},$$

where $\mathbb{F}^{f/f'}$ denotes the product ring with coordinate-wise operations.

In Lemma 2.4 below, we show that under the above isomorphisms, the $\mathbb{F}'$-linear functions $\bar{L} \colon \mathbb{F}^{f/f'} \to \mathbb{F}'$ correspond bijectively with the R'-linear functions $L \colon R/\tilde{\mathfrak{p}}_{i'} \to R'/\mathfrak{p}'_{i'}$, for all $i' \in G'$. Recall that any function of the latter type can be expressed as $L(a) = \mathrm{Tr}_{K/K'}(r \cdot a)$ for some fixed $r \in K$. Conversely, every function L (with domain and range as above) that can be expressed as $L(a) = \mathrm{Tr}_{K/K'}(r \cdot a)$ is clearly R'-linear, so it always induces an $\mathbb{F}'$-linear function. The heart of Lemma 2.4 is the following fact.

Lemma 2.3. *Let $\mathfrak{p}'_{i'}$ for some $i' \in G'$ be a prime ideal lying over p in R', and let $\tilde{\mathfrak{p}}_{i'}$ be the radical of $\mathfrak{p}_{i'}R$. Let $r' \in R' \subseteq R$ be arbitrary, and let $s = h'_{i'}(r' \bmod \mathfrak{p}'_{i'}) \in \mathbb{F}' \subseteq \mathbb{F}$. Then*

$$\tilde{h}_{i'}(r' \bmod \tilde{\mathfrak{p}}_{i'}) = (s, s, \ldots, s) \in \mathbb{F}'^{f/f'},$$

i.e., every entry of $\tilde{h}_{i'}(r' \bmod \tilde{\mathfrak{p}}_{i'})$ is equal to $h'_{i'}(r' \bmod \mathfrak{p}'_{i'})$.

Proof. Recall that under our choice of isomorphisms, $\omega_{\bar{m}'} = \omega_{\bar{m}}^{\bar{m}/\bar{m}'} \in \mathbb{F}'$ is of order $\bar{m}'$, and $p^\ell \cdot i = i' \bmod \bar{m}'$, where $\ell \geq 0$ is the integer satisfying $m/m' = (\bar{m}/\bar{m}') \cdot p^\ell$. Also recall that

$$\tilde{h}_{i'}(r' \bmod \tilde{\mathfrak{p}}_{i'}) = \big(h_i(r' \bmod \mathfrak{p}_i)\big)_{i \in g^{-1}(i')}.$$

For the representative i of each coset $i\langle p \rangle \in g^{-1}(i')$, the entry $h_i(r' \bmod \mathfrak{p}_i)$ is obtained by mapping ζ_m to $\omega_{\bar{m}}^i$, and hence also mapping $\zeta_{m'} = \zeta_m^{m/m'} = \zeta_m^{(\bar{m}/\bar{m}') \cdot p^\ell}$ to

$$\omega_{\bar{m}}^{(\bar{m}/\bar{m}') \cdot p^\ell \cdot i} = \omega_{\bar{m}'}^{p^\ell \cdot i} = \omega_{\bar{m}'}^{i'} \in \mathbb{F}',$$

which is exactly the mapping done by $h'_{i'}$. Since $r' \in R' = \mathbb{Z}[\zeta_{m'}]$, this proves the claim. □

Lemma 2.4. *Let $i' \in G'$ be arbitrary, and let $\mathfrak{p}' = \mathfrak{p}'_{i'}$ and $\tilde{\mathfrak{p}} = \tilde{\mathfrak{p}}_{i'}$. Then under the isomorphisms $h' = h'_{i'}$ and $\tilde{h} = \tilde{h}_{i'}$ defined above, the $\mathbb{F}'$-linear functions $\bar{L} \colon \mathbb{F}^{f/f'} \to \mathbb{F}'$ are in bijective correspondence with the R'-linear functions $L \colon R/\tilde{\mathfrak{p}} \to R'/\mathfrak{p}'$.*

Proof. For any $\mathbb{F}'$-linear function $\bar{L}$, we claim that $L = h'^{-1} \circ \bar{L} \circ \tilde{h}$ is the corresponding R'-linear function. To see this, note that by Lemma 2.3 and the fact that $\tilde{h}$ is a ring homomorphism, for any $r' \in R'$ and $a \in R/\tilde{\mathfrak{p}}$ we have

$$\tilde{h}(r' \cdot a) = \tilde{h}(r' \bmod \tilde{\mathfrak{p}}) \odot \tilde{h}(a) = h'(r' \bmod \mathfrak{p}') \cdot \tilde{h}(a) \in \mathbb{F}^{f/f'},$$

where multiplication $\odot$ in $\mathbb{F}'^{f'}$ and $\mathbb{F}^f$ is coordinate-wise. By $\mathbb{F}'$-linearity of $\bar{L}$ and the fact that h' is a ring homomorphism, we have

$$L(r' \cdot a) = h'^{-1}(\bar{L}(\tilde{h}(r' \cdot a))) = h'^{-1}\big(h'(r' \bmod \mathfrak{p}') \cdot \bar{L}(\tilde{h}(a)) \big) = r' \cdot L(a) \in R'/\mathfrak{p}',$$

as desired. The other direction proceeds essentially identically, with $\bar{L} = h' \circ L \circ \tilde{h}^{-1}$. □

An application of the Chinese Remainder Theorem with the prime ideals $\tilde{\mathfrak{p}}_{i'}$ in R, combined with Lemma 2.4, immediately yields the following corollary.

Corollary 2.5. *Let $\mathfrak{p}' = \prod_{i' \in G'} \mathfrak{p}'_{i'}$ and $\mathfrak{p} = \prod_{i' \in G'} \tilde{\mathfrak{p}}_i$ be the radicals of pR' and pR, respectively. Then under the isomorphisms $\{h_{i'}\}_{i' \in G'}$ and $\{\tilde{h}_{i'}\}_{i' \in G'}$ defined above, the R'-linear functions $L\colon R/\mathfrak{p} \to R'/\mathfrak{p}'$ are in bijective correspondence with the functions $\bar{L}\colon (\mathbb{F}^{f/f'})^{f'} \to \mathbb{F}'^{f'}$ of the form*

$$\bar{L}\Big((\vec{a}_{i'})_{i' \in G'}\Big) = \big(\bar{L}_{i'}(\vec{a}_{i'})\big)_{i' \in G'},$$

where every $\bar{L}_{i'}\colon \mathbb{F}^{f/f'} \to \mathbb{F}'$ is $\mathbb{F}'$-linear.

We note that given a function $\bar{L}\colon (\mathbb{F}^{f/f'})^{f'} \to \mathbb{F}'^{f'}$ as in Corollary 2.5, we can efficiently find an R'-linear function $\hat{L}\colon R \to R'$ that induces the corresponding L: first, fix an arbitrary R'-basis $B = \{b_j\}$ of R. Then, using the isomorphisms $h'_{i'}$ and $\tilde{h}_{i'}$, the values of $L(b_j \bmod \mathfrak{p}) \in R'/\mathfrak{p}'$ are determined by $\bar{L}$, and uniquely define L by R'-linearity. We can then define each $\hat{L}(b_j) \in R'$ to be an arbitrary representative of $L(b_j \bmod \mathfrak{p})$; these choices uniquely determine $\hat{L}$, by R'-linearity. Finally, we can represent $\hat{L}$ explicitly in trace form as $\hat{L}(a) = \mathrm{Tr}_{K/K'}(r \cdot a)$ for some $r \in K$: recalling that K is a vector space over K' with K'-basis B, we have a full-rank system of linear equations $\hat{L}(b_j) = \mathrm{Tr}_{K/K'}(r \cdot b_j) \in K'$, which we can solve to obtain $r \in K$.

Looking ahead, in our application to homomorphic computation we will have certain linear functions that we want to evaluate (e.g., projection functions), and we will do so by finding the corresponding constant r, then multiplying by r and taking the trace (see Section 3.3 for further details). To apply these steps in the context of a homomorphic encryption scheme, we need the notion of the *dual* of the ring of integers, described next.

2.1.4 Duality

An important and useful object in K is the *dual* of R (also known as the *codifferent* of K), defined as

$$R^\vee = \{a \in K : \mathrm{Tr}_{K/\mathbb{Q}}(aR) \subseteq \mathbb{Z}\} \supseteq R.$$

Because $\mathrm{Tr}_{K/\mathbb{Q}} = \mathrm{Tr}_{K'/\mathbb{Q}} \circ \mathrm{Tr}_{K/K'}$, it is easy to verify that also $R^\vee = \{a \in K : \mathrm{Tr}_{K/K'}(aR) \subseteq R'^\vee\}$. Therefore, we have the convenient equation

$$\mathrm{Tr}_{K/K'}(R^\vee) = R'^\vee. \tag{2.3}$$

Note that by contrast, frequently $\mathrm{Tr}_{K/K'}(R)$ does *not* equal R', but is instead some proper ideal of it.[5] Many other algebraic and geometric advantages of working with $R^\vee$ instead of R are discussed in [15, 16].

[5] This is easily seen, e.g., for $R = \mathbb{Z}[\zeta_{2^k}]$ and $R' = \mathbb{Z}$, where $\mathrm{Tr}(R) = 2^{k-1}R'$ because $\mathrm{Tr}(1) = 2^{k-1}$ and $\mathrm{Tr}(\zeta_{2^k}^j) = 0$ for $j = 1, \ldots, 2^{k-1} - 1$.

The codifferent is a principal fractional ideal, i.e., $R^\vee = t^{-1}R$ for some $t \in R$ (which is not unique). Therefore, division by t induces a bijection from R to $R^\vee$, and from any quotient ring $R_\mathfrak{p} = R/\mathfrak{p}$ to $R^\vee_\mathfrak{p} = R^\vee/\mathfrak{p}R^\vee$. Although the target objects are *not* rings (because $R^\vee \cdot R^\vee \not\subseteq R^\vee$), they are R-modules, and the bijections are R-module isomorphisms.

Of course, we also have $R'^\vee = t'^{-1}R'$ for some $t' \in R'$. By Equation (2.3) and K'-linearity of the trace, for any ideal $\mathfrak{p}$ in R', we have

$$\mathrm{Tr}_{K/K'}(R^\vee_\mathfrak{p}) = \mathrm{Tr}_{K/K'}(R^\vee/\mathfrak{p}R^\vee) = R'^\vee/\mathfrak{p}R'^\vee = R'^\vee_\mathfrak{p}.$$

In the previous subsection we considered R'-linear functions $L\colon R \to R'$ (or their induced functions $R_p \to R'_p$), which can always be expressed as $L(a) = \mathrm{Tr}_{K/K'}(r^\vee \cdot a)$ for some fixed $r^\vee \in K$. Typically, $r^\vee$ is *not* in R because $\mathrm{Tr}_{K/K'}(R) \neq R'$, but it is easy to see that $r^\vee \in t'R^\vee$ always, because if not, then $\mathrm{Tr}_{K/K'}(r^\vee R) \not\subseteq t'R'^\vee = R'$. For the purposes of our field-switching procedure, it will be more convenient to instead work with corresponding R'-linear functions from $R^\vee$ to $R'^\vee$, which can be represented in trace form by elements of R. Namely, for an R'-linear function $L : R \to R'$, where $L(a) = \mathrm{Tr}_{K/K'}(r^\vee \cdot a)$ for some $r^\vee \in t'R^\vee$, we will consider the corresponding function

$$L^\vee\colon R^\vee \to R'^\vee, \quad L^\vee(a^\vee) = L(t \cdot a^\vee)/t' = \mathrm{Tr}_{K/K'}((t/t')r^\vee \cdot a^\vee) = \mathrm{Tr}_{K/K'}(r \cdot a^\vee),$$

which is represented by $r = (t/t')r^\vee \in R$.

Following [16], we extend the operation $[\cdot]_q$ to $R^\vee_p$ by fixing a particular $\mathbb{Z}$-basis of $R^\vee$ (and $\mathbb{Z}_q$-basis of $R^\vee_q$), called the *decoding basis*, and representing the argument as a $\mathbb{Z}_q$-combination of the basis vectors and applying the $[\cdot]_q$ operation to each of its coefficients. It is shown in [16, Section 6.2] that every sufficiently short (as always, under the canonical embedding) $e \in R^\vee$ is indeed the "canonical" representative of its coset modulo $qR^\vee$. Specifically, if $\|e\| < q/(2\sqrt{n})$ then $[e \bmod qR^\vee]_q = e$.

2.1.5 Good Bases of R and $R^\vee$

We now have almost all the ingredients we need to describe the homomorphic cryptosystem and our field-switching transformation. The final background material we need concerns the geometry of R as a module over R' (respectively, $R^\vee$ as a module over $R'^\vee$). Specifically, we construct certain "good" bases of the ring R and its dual $R^\vee$ in terms of R' and $R'^\vee$ (respectively), and prove some of their useful geometrical properties. This (somewhat technical) material is used only in Section 3.1, where we prove the hardness of ring-LWE over K with secret in R', assuming its hardness over K' with secret in R'.

Since K is a vector space of dimension n/n' over K', the field K has a K'-*basis* (which is not unique), i.e., a set of n/n' elements of K that are linearly independent over K', so that every element of K can be represented uniquely as a K'-linear combination of the basis elements. Similarly, an R'-*basis* of R is a set of n/n' elements in R, such that every element of R can be represented uniquely as an R'-linear combination of the basis elements. An $R'^\vee$-basis of $R^\vee$ is defined analogously.

We wish to construct an R'-basis of R, and a corresponding dual $R'^\vee$-basis of $R^\vee$ (any of which are K'-bases of K), which are "good" in the following sense: for any vector of K'-coefficients (with respect to the basis) which are *short* under σ', the corresponding K-element is also short under σ. More formally, represent an ordered K'-basis of K as a vector $\vec{b} = (b_j) \in K^{n/n'}$, and similarly for an arbitrary vector of K'-coefficients $\vec{a} = (a_j) \in K'^{(n/n')}$, which defines the K-element $a = \langle \vec{a}, \vec{b} \rangle = \sum_j a_j \cdot b_j$. Then by linearity, the basis $\vec{b}$ induces a matrix $B \in \mathbb{C}^{n \times n}$ such that

$$\sigma(a) = B \cdot \sigma'(\vec{a}), \quad \text{where } \sigma'(\vec{a}) = \big(\sigma'(a_j)\big)_j. \tag{2.4}$$

We seek an R'-basis $\vec{b}$ of R for which B (nearly) preserves Euclidean norms up to some scaling factor, i.e., all of its singular values are (nearly) equal.

In addition, for any K'-basis $\vec{b} = (b_j)$ of K, its *dual* K'-basis $\vec{b}^\vee = (b_j^\vee) \subseteq K$ is uniquely defined by the linear constraints $\mathrm{Tr}_{K/K'}(b_j \cdot b_{j'}^\vee) = 1$ if $j = j'$, and 0 otherwise. It is a straightforward exercise to verify that if $\vec{b}$ is an R'-basis of R, then $\vec{b}^\vee$ is an $R'^\vee$-basis of $R^\vee$. Moreover, the matrix $B^\vee$ induced by $\vec{b}^\vee$ is $B^\vee = B^{-T}$, so its singular values are simply the inverses of those of B.

Lemma 2.6. *Let $\hat{m} = m/2$ if m is even and m' is odd, otherwise $\hat{m} = m$, and let $r = \mathrm{rad}(m)/\mathrm{rad}(m')$ be the product of all primes that divide m but not m'. There exists an efficiently computable R'-basis $\vec{b}$ of R, for which the corresponding matrix B has largest and smallest singular values*

$$s_1(B) = \sqrt{\hat{m}/m'} \quad \textit{and} \quad s_n(B) = \sqrt{m/(rm')},$$

respectively. In particular, if $r \in \{1, 2\}$ then B is a unitary matrix scaled by a $\sqrt{\hat{m}/m'}$ factor.

Lemma 2.6 implies that for any $\vec{a} \in K'^{(n/n')}$ defining $a = \langle \vec{a}, \vec{b} \rangle \in K$ and $a^\vee = \langle \vec{a}, \vec{b}^\vee \rangle \in K$,

$$\|\sigma(a)\| \leq \sqrt{\hat{m}/m'} \cdot \|\sigma'(\vec{a})\| \quad \text{and} \quad \|\sigma(a^\vee)\| \leq \sqrt{rm'/m} \cdot \|\sigma'(\vec{a})\|. \tag{2.5}$$

More generally, if the a_j are independent and have Gaussian distributions over (the canonical embedding of) K', then a and $a^\vee$ also have (possibly non-spherical) Gaussian distributions over K.[6] Since we are not too concerned with the exact distributions, we omit a precise calculation, which is standard. However, one particular case of interest is when the a_j are all i.i.d. according to a spherical Gaussian of parameter s, and $r \in \{1, 2\}$ so that B (respectively, $B^\vee$) is a scaled unitary matrix. Then because spherical Gaussians are invariant under unitary transformations, a (resp., $a^\vee$) is distributed according to a spherical Gaussian of parameter $s\sqrt{\hat{m}/m'}$ (resp., $s\sqrt{m'/\hat{m}}$).

The remainder of this subsection is devoted to proving Lemma 2.6. We denote the k-dimensional identity matrix by I_k, we use $\otimes$ to denote the Kronecker (or tensor) product of vectors and matrices, and we apply functions to vectors and matrices component-wise.

Following the treatment given in [16], let $m = \prod_\ell m_\ell$ be the prime-power factorization of m, i.e., the $m_\ell > 1$ are powers of distinct primes. The ring $R = \mathbb{Z}[\zeta_m]$ has the following $\mathbb{Z}$-basis $\vec{p}$, which is called the "powerful" basis:

$$\vec{p} = \bigotimes\nolimits_\ell \vec{p}_{m_\ell}\ , \quad \text{where } \vec{p}_{m_\ell} = \left(\zeta_{m_\ell}^j\right)_{j \in [\varphi(m_\ell)]}.$$

The set $\vec{p}_{m_\ell}$ is called the "power" $\mathbb{Z}$-basis of $\mathbb{Z}[\zeta_{m_\ell}] = \mathbb{Z}[\zeta_m^{m/m_\ell}] \subseteq R$.

Similarly, let $m' = \prod_\ell m'_\ell$ where each m'_ℓ divides m_ℓ, i.e., they are both powers of the same prime (though possibly $m'_\ell = 1$). Then the powerful $\mathbb{Z}$-basis of R' is defined as $\vec{p}' = \bigotimes_\ell \vec{p}_{m'_\ell}$, where the power bases $\vec{p}_{m'_\ell}$ are defined as above. Notice that when $m'_\ell > 1$, there is a bijective correspondence between $j \in [\varphi(m_\ell)]$ and $(j', k) \in [\varphi(m'_\ell)] \times [m_\ell/m'_\ell]$, via $j = (m_\ell/m'_\ell)j' + k$. Therefore, the power bases $\vec{p}_{m_\ell}$ factor as

$$\vec{p}_{m_\ell} = \vec{p}_{m'_\ell} \otimes \vec{b}_\ell, \quad \text{where } \vec{b}_\ell = \begin{cases} \left(\zeta_{m_\ell}^k\right)_{k \in [m_\ell/m'_\ell]} & \text{if } m'_\ell > 1 \\ \vec{p}_{m_\ell} & \text{if } m'_\ell = 1. \end{cases}$$

Hence, using the commutativity of the Kronecker product (up to some permutation) we can factor the powerful basis $\vec{p}$ of R as

$$\vec{p} = \vec{p}' \otimes \vec{b}, \quad \text{where } \vec{b} = \bigotimes\nolimits_\ell \vec{b}_\ell. \tag{2.6}$$

[6] To be completely formal, the Gaussians should be over continuous spaces of the form $K \otimes_{\mathbb{Q}} \mathbb{R}$; see [16].

Because $\vec{p}'$ is a $\mathbb{Z}$-basis of R', it follows that $\vec{b}$ is an R'-basis of R. We next calculate the matrix $B \in \mathbb{C}^{n \times n}$ induced by $\vec{b}$, and verify that it indeed satisfies the claims in the lemma statement.

Following [16, Section 3], for any prime power $\tilde{m}$ we define $\mathrm{CRT}_{\tilde{m}}$ to be the complex $\varphi(\tilde{m})$-by-$\varphi(\tilde{m})$ matrix with $\omega_{\tilde{m}}^{i \cdot j}$ in its ith row and jth column, for $i \in \mathbb{Z}_{\tilde{m}}^*$ and $j \in [\varphi(\tilde{m})]$. Using the prime-power factorizations of our m, m', we define $\mathrm{CRT}_m = \bigotimes_\ell \mathrm{CRT}_{m_\ell}$ and $\mathrm{CRT}_{m'} = \bigotimes_\ell \mathrm{CRT}_{m'_\ell}$. Then up to a permutation of the rows (determined by the CRT correspondence between $\mathbb{Z}_m^*$ and $\prod_\ell \mathbb{Z}_{m_\ell}^*$), we have

$$\sigma(\vec{p}^T) = \mathrm{CRT}_m,$$

i.e., the columns of CRT_m are $\sigma(p_j)$ for each entry p_j of the row vector $\vec{p}^T$. In particular, $\sigma(\langle \vec{c}, \vec{p} \rangle) = \mathrm{CRT}_m \cdot \vec{c}$ for any $\vec{c} \in \mathbb{Q}^n$. Similarly, $\sigma'((\vec{p}')^T) = \mathrm{CRT}_{m'}$ up to a row permutation.

We now claim that, up to some permutations of B's rows and columns,

$$B = \mathrm{CRT}_m \cdot \left(\mathrm{CRT}_{m'}^{-1} \otimes I_{n/n'}\right) = \bigotimes\nolimits_\ell \left(\mathrm{CRT}_{m_\ell} \cdot \left(\mathrm{CRT}_{m'_\ell}^{-1} \otimes I_{\varphi(m_\ell)/\varphi(m'_\ell)}\right)\right), \tag{2.7}$$

where the second equality follows by the mixed-product property and the commutativity (up to row and column permutations) of the Kronecker product. To see the first equality, notice that for any $\vec{a} \in K'^{(n/n')}$ defining $a = \langle \vec{a}, \vec{b} \rangle \in K$, the matrix $(\mathrm{CRT}_{m'}^{-1} \otimes I)$ maps from (a suitable permutation of) the concatenated embeddings $\sigma'(\vec{a})$, to a vector $\vec{c} \in \mathbb{Z}^n$ of coefficients such that $\vec{a} = \langle \vec{c}, \vec{p}' \otimes I_{n/n'} \rangle$. In addition,

$$a = \langle \vec{a}, \vec{b} \rangle = \vec{c}^T \cdot (\vec{p}' \otimes I_{n/n'}) \cdot \vec{b} = \langle \vec{c}, \vec{p}' \otimes \vec{b} \rangle = \langle \vec{c}, \vec{p} \rangle.$$

Therefore, $\sigma(a) = \mathrm{CRT}_m \cdot \vec{c} = \mathrm{CRT}_m \cdot (\mathrm{CRT}_{m'}^{-1} \otimes I) \cdot \sigma'(\vec{a})$, as desired.

Now, by the last expression in Equation (2.7), and because singular values are multiplicative under the Kronecker product, from now on we drop all the ℓ subscripts, and assume without loss of generality that m and m' are powers of the same prime p (where possibly $m' = 1$). We analyze the singular values of $\mathrm{CRT}_m(\mathrm{CRT}_{m'}^{-1} \otimes I)$, for the cases $m' = 1$ and $m' > 1$. In the first case, clearly $\mathrm{CRT}_{m'} = I_1$, and it is shown in [16, Section 4] that the largest singular value of CRT_m is $\sqrt{m/2}$ if m is even and $\sqrt{m}$ otherwise, and its smallest singular value is $\sqrt{m/p}$.

For the case $m' > 1$, it follows from the decompositions given in [16, Section 3] that, up to some row permutation,

$$\mathrm{CRT}_m = \sqrt{m/p} \cdot Q \cdot (\mathrm{CRT}_p \otimes I_{m/p})$$

for some unitary matrix Q, and similarly for $\mathrm{CRT}_{m'}$. Then a routine calculation using elementary properties of the Kronecker product reveals that $\mathrm{CRT}_m(\mathrm{CRT}_{m'}^{-1} \otimes I)$ is some unitary matrix scaled by a $\sqrt{m/m'}$ factor, so all its singular values are $\sqrt{m/m'}$. This completes the proof of Lemma 2.6.

2.2 Homomorphic Cryptosystems

In ring-LWE-based cryptosystems for arbitrary cyclotomics [16] (generalizing those of [15, 4, 3]), the plaintext space is R_p for some integer $p \geq 2$ that is coprime with all the odd primes dividing m. We assume that p is prime, which is without loss of generality by the Chinese Remainder Theorem. Ciphertexts are elements of $(R_q^\vee)^2$ for some integer q that is coprime with p, and the secret key is some $s \in R$. A ciphertext $c = (c_0, c_1) \in (R_q^\vee)^2$ that encrypts a plaintext $b \in R_p$ with respect to s satisfies the decryption relation

$$c_0 + c_1 \cdot s = e \pmod{qR^\vee} \tag{2.8}$$

for some sufficiently short $e \in R^\vee$ such that $t \cdot e = b \pmod{pR}$. (Recall that $R^\vee = t^{-1}R$ for some $t \in R$, so $t \cdot e \in R$.) We refer to e as the *noise* of the ciphertext. Throughout this work we implicitly assume that the modulus q is large enough relative to $\|e\|$, so that $[c_0 + c_1 \cdot s]_q = e \in R^\vee$ (see Section 2.1.4 above). Therefore, the decryption algorithm can simply compute e and output $t \cdot e \bmod pR$. As shown in [4, 3, 16], this system (augmented by some additional public values, for greater efficiency) supports additive and multiplicative homomorphisms.

3 The Field-Switching Procedure

Our procedure performs the following operation. Given a big-field ciphertext $c \in (R_q^\vee)^2$ that encrypts a plaintext $b \in R_p$ with respect to a big-ring secret key $s \in R$, and a description of an R'-linear function $L\colon R_\mathfrak{p} \to R'_\mathfrak{p}$ to apply to the plaintext (where recall that $\mathfrak{p}$ and $\mathfrak{p}'$ are the radicals of p in R and R', respectively), it outputs a small-field ciphertext $c' \in (R_q'^\vee)^2$ that encrypts $b' = L(b) \in R'_{\mathfrak{p}'}$ with respect to some small-ring secret key $s' \in R'$. (Recall that Corollary 2.5 characterizes how L corresponds to the induced function $\bar{L}\colon \mathbb{F}^f \to \mathbb{F}'^{f'}$ that is applied to the vector of finite field elements encoded by b.)

The procedure consists of the following three steps:

1. **Switch to a small-ring secret key.** We use the key-switching method from [5, 3, 16] to produce a ciphertext which is still over the big field K and encrypts the same plaintext $b \in R_\mathfrak{p}$, but with respect to a secret key $s' \in R' \subseteq R$ belonging to the small subring.

2. **Multiply by an appropriate (short) scalar.** We multiply the components of the resulting ciphertext by a short element $r \in R$ that corresponds to the desired R'-linear function to be applied to the input plaintext b.

3. **Map to the small field.** We map the resulting big-field ciphertext (over $R_q^\vee$) to a small-field ciphertext (over $R_q'^\vee$) simply by taking the trace $\mathrm{Tr}_{K/K'}$ of its two components. The resulting ciphertext will still be with respect to the small-ring secret key $s' \in R'$, but will encrypt the plaintext $b' = L(b) \in R'_{\mathfrak{p}'}$.

Note that Steps 2 and 3 can be repeated multiple times on the same ciphertext (from Step 1), to apply several different R'-linear functions. In this way, the entire input plaintext can be preserved, but in a decomposed form.

3.1 Step 1: Switching to a Small-Ring Secret Key

To switch to a small-field secret key, we publish a "key-switching hint," which essentially encrypts the big-ring secret key $s \in R$ under the small-ring key $s' \in R'$, using ciphertexts over the big field. Note that encrypting s under a small-ring secret key s' has security implications, since the dimension of the underlying RLWE problem is smaller. In our case, though, the ultimate goal is to switch to a ciphertext over the smaller field, so we will not lose any additional security by publishing the hint. Indeed, we show below that assuming the hardness of the decision RLWE problem in the small field, the key-switching hint reveals nothing about the big-ring secret key. The essence of that claim is Lemma 3.1 below, which says (informally) that RLWE in the big field, with secret chosen in the small ring $R' \subseteq R$, is no easier than RLWE in the small field.

Ring-LWE. The ring-LWE (RLWE) problem [15] (in K) with *continuous* error is parameterized by a modulus q, a "secret distribution" υ over R, and an "error distribution" ψ over K, which is usually a

Gaussian (in the canonical embedding) and is therefore concentrated on short elements.[7] For $s \in R$, define the distribution $A_{s,\psi}$ that is sampled by choosing $\alpha \in R_q^\vee$ uniformly at random, choosing $\epsilon \leftarrow \psi$, and outputting the pair $(\alpha, \beta = \alpha \cdot s + \epsilon \bmod qR^\vee) \in R_q^\vee \times K/qR^\vee$. One equivalent form of the (average-case) decision $\mathrm{RLWE}_{q,\psi,\upsilon}$ problem (in K) is, given some ℓ pairs $(\alpha_i, \beta_i) \in R_q^\vee \times K/qR^\vee$, distinguish between the following two cases: in one case, the pairs are chosen independently from $A_{s,\psi}$ for a random $s \leftarrow \upsilon$ (which remains the same for all samples); in the other case, the pairs are all independent and uniformly random over $R_q^\vee \times K/qR^\vee$. For appropriate parameters q, ψ, υ and ℓ, solving this decision problem with non-negligible distinguishing advantage is as hard as approximating the shortest vector problem on ideal lattices in R, via a quantum reduction. See [15, 16] for precise statements and further details.

Let $\vec{b}^\vee = (b_j^\vee)_{j \in [n/n']}$ be any $R'^\vee$-basis of $R^\vee$, and hence a K'-basis of K. Then for any error distribution ψ' over K', we can define an error distribution ψ over K as $\psi = \langle \psi'^{(n/n')}, \vec{b}^\vee \rangle$, i.e., a sample from ψ is generated by choosing independent $\epsilon_j \leftarrow \psi'$ (for $j \in [n/n']$) and outputting $\epsilon = \sum_j \epsilon_j b_j^\vee \in K$.

Lemma 3.1. *Let ψ' be an error distribution over K', and let $\psi = \langle \psi'^{(n/n')}, \vec{b}^\vee \rangle$ be the error distribution over K as described above. If the decision $\mathrm{RLWE}_{q,\psi',\upsilon'}$ problem (in K') is hard for some distribution υ' over $R' \subseteq R$, then the decision $\mathrm{RLWE}_{q,\psi,\upsilon'}$ problem (in K) is also hard.*

Although the lemma holds for any $R'^\vee$-basis of $R^\vee$, it is most useful with a basis having "good geometric properties." Specifically, in our case we need the property that if ψ' is concentrated on short elements of K', then ψ is similarly concentrated on short elements of K. Such a basis $\vec{b}^\vee$ is constructed in Lemma 2.6 of Section 2.1.5. For example, if ψ' is a continuous (spherical) Gaussian with parameter s and $r = \mathrm{rad}(m)/\mathrm{rad}(m') = 1$, then ψ' is a spherical Gaussian with parameter $s\sqrt{m'/m} = s\sqrt{n'/n}$.[8]

Proof. It suffices to give an efficient, deterministic reduction that takes n/n' pairs $(\alpha_j, \beta_j) \in R_q'^\vee \times K'/qR'^\vee$ and outputs a single pair $(\alpha, \beta) \in R^\vee \times K/qR^\vee$, with the following properties: if the pairs (α_j, β_j) are i.i.d. according to $A_{s',\psi'}$ for some $s' \in R'$, then (α, β) is distributed according to $A_{s,\psi}$; and if the pairs (α_j, β_j) are independent and uniformly random, then (α, β) is uniformly random. The reduction simply outputs $(\alpha = \langle \vec{\alpha}, \vec{b}^\vee \rangle, \beta = \langle \vec{\beta}, \vec{b}^\vee \rangle)$, where $\vec{\alpha} = (\alpha_j)_j$ and $\vec{\beta} = (\beta_j)_j$.

Since $\vec{b}^\vee$ is an $R'^\vee$-basis of $R^\vee$ and hence an $R_q'^\vee$-basis of $R_q^\vee$, it is immediate that the reduction maps the uniform distribution to the uniform distribution. On the other hand, if the samples $(\alpha_j \beta_j)$ are drawn from $A_{s',\psi'}$, i.e, $\beta_j = \alpha_j \cdot s' + \epsilon_j \bmod qR'^\vee$ for $\epsilon_j \leftarrow \psi$, then α is still uniformly random, and

$$\beta = \langle \vec{\beta}, \vec{b}^\vee \rangle = \langle \vec{\alpha}, \vec{b}^\vee \rangle \cdot s' + \langle \vec{\epsilon}, \vec{b}^\vee \rangle = \alpha \cdot s' + \epsilon \pmod{qR^\vee},$$

where $\vec{\epsilon} = (\epsilon_j)_j$ and ϵ has distribution ψ. This completes the proof. □

Key switching. In [5, 3, 16] it is shown how, given an $s \in R$ and sufficiently many RLWE samples (over K) with short noise and any secret $s' \in R$, it is possible to generate a "key-switching hint" with the following functionality: given the hint and any valid ciphertext c (over K) encrypted under s and with sufficiently short noise, it is possible to efficiently generate a ciphertext c' (also over K) with short noise encrypted under s'. Moreover, the hint is indistinguishable from uniformly random over its domain (even given s), assuming that the RLWE samples are.

[7] Again, to be completely formal, a Gaussian should be defined over $K_\mathbb{R}$; see Footnote 6.

[8] Note that the factor $\sqrt{n'/n} \leq 1$ does not really amount to any effective decrease in the noise, because the "sparsity" of $R'^\vee$ versus $R^\vee$ is greater by a corresponding factor.

For our transformation, we apply Lemma 3.1 using the "good basis" $\vec{b}^\vee$ from Lemma 2.6, thus obtaining RLWE samples over K relative to the secret $s' \in R' \subseteq R$, with noise distribution ψ which is concentrated on short vectors, and with security based on the hardness of $\mathrm{RLWE}_{q,\psi',\upsilon'}$ problem in K'. We then construct the key-switching hint from these samples as described in [16, Section 8.3],

3.2 Steps 2 and 3: Mapping to the Small Field

Our goal now is to transform a valid big-field ciphertext $c = (c_0, c_1) \in (R_q^\vee)^2$, which encrypts some $b \in R_\mathfrak{p}$ with respect to some secret key $s' \in R' \subseteq R$, into a small-field ciphertext $c' = (c'_0, c'_1) \in (R'^\vee_q)^2$ that encrypts the related plaintext $b' = L(b)$ with respect to the same secret key s', where $L \colon R_\mathfrak{p} \to R'_{\mathfrak{p}'}$ is any desired R'-linear function.

The process works as follows:

1. Since L is R'-linear, by the discussion at the end of Section 2.1.3 and in Section 2.1.4, we can find some $r^\vee \in t' R^\vee$ such that $L(a) = \mathrm{Tr}_{K/K'}(r^\vee \cdot a) \bmod \mathfrak{p}'$.

2. We then find a *short* representative $r \in (t/t')r^\vee + pR \in R_p$, using a "good" basis of pR (i.e., one that has small singular values under σ, e.g., the "powerful" basis as constructed in Section 2.1.5).

   The chosen r defines the R'-linear function $L^\vee \colon R^\vee \to R'^\vee$ of the form $L^\vee(a^\vee) = \mathrm{Tr}_{K/K'}(r \cdot a^\vee)$, whose induced function from $R_\mathfrak{p}^\vee$ to $R'^\vee_{\mathfrak{p}'}$ satisfies

$$t' \cdot L^\vee(a^\vee) = L(t \cdot a^\vee) \pmod{\mathfrak{p}'}. \tag{3.1}$$

3. We obtain our small-field ciphertext by applying $L^\vee$ (or more precisely, the induced function from $R_q^\vee$ to $R'^\vee_q$) to c_0, c_1, setting

$$c'_i = L^\vee(c_i) = \mathrm{Tr}_{K/K'}(r \cdot c_i) \in R'^\vee_q, \quad i = 0, 1.$$

Lemma 3.2. *The ciphertext $c' = (c'_0, c'_1)$ is an encryption of $b' = L(b) \in R'_{\mathfrak{p}'}$ under secret key $s' \in R'$, with noise $e' = L^\vee(e) \in R'^\vee$ of length $\|e'\| \le \|e\| \cdot \|r\|_\infty \cdot \sqrt{n/n'}$, where e is the noise in the original ciphertext c.*

We note that the factor $\sqrt{n/n'}$ in the bound on $\|e'\|$ does not actually amount to any effective increase in the noise, because the dimension has decreased by a corresponding factor, and hence the size of e' relative to $R'^\vee$ remains the same as that of e relative to $R^\vee$. More precisely, the original ciphertext c decrypts correctly if $q > 2\sqrt{n}\|e\|$, whereas c' decrypts correctly if $q > 2\sqrt{n'}\|e'\|$ (see Section 2.1.4). Therefore, the only practical increase in the noise is due solely to $\|r\|_\infty$.

Proof. We need to show three things: that $\|e'\|$ is bounded as claimed, that $c'_0 + c'_1 \cdot s = e' \pmod{qR'^\vee}$, and that $t' \cdot e' = b' = L(b) \pmod{\mathfrak{p}'}$.

1. The first claim follows immediately by Corollary 2.2 and the inequality $\|r \cdot e\| \le \|r\|_\infty \cdot \|e\|$.

2. For the second claim, recall that $c_0 + c_1 \cdot s = e \pmod{qR^\vee}$. Then because the induced function $L^\vee \colon R_q^\vee \to R'^\vee_q$ is R'-linear and $s' \in R'$, we have

$$c'_0 + c'_1 \cdot s' = L^\vee(c_0 + c_1 \cdot s') = L^\vee(e) = e' \pmod{R'^\vee_q}.$$

3. For the last claim, because $t \cdot e = b \bmod pR$ and by Equation (3.1), we have

$$t' \cdot e' = t' \cdot L^\vee(e) = L(t \cdot e) = L(b) \pmod{\mathfrak{p}'}.$$

□

3.3 Applying the Field-Switching Procedure

A typical application of the field-switching procedure during homomorphic evaluation of some circuit will begin with a big-field ciphertext that encrypts an array of plaintext values in the subfield $\mathbb{F}'$, as embedded in $\mathbb{F}$.[9] The above procedure is then applied to decompose the ciphertext into a number of small-field ciphertexts, each encrypting a subset of the plaintext values. Since big-field ciphertexts have room for f plaintext elements, but small-field ciphertexts can only hold f' elements, we may need up to f/f' small-field ciphertexts to hold all the plaintext values that we are interested in. That is, we apply our field-switching transformation using the f'-fold concatenations $\bar{L}_i^{f'}$ of the $\mathbb{F}'$-linear selection functions $\bar{L}_i\colon \mathbb{F}^{f/f'} \to \mathbb{F}'$, $i \in [f/f']$, where $\bar{L}_i$ just selects the ith value (in $\mathbb{F}'$).[10]

Referring to Figure 1 for an example, the big-field ciphertext holds (up to) six plaintext values, and each small-field ciphertext can hold two values, with the big-field plaintext "slots" corresponding to $\mathfrak{p}_1, \mathfrak{p}_{15}, \mathfrak{p}_{22}$ lying over the small-field plaintext slot of $\mathfrak{p}'_1$, and the big-field slots corresponding to $\mathfrak{p}_3, \mathfrak{p}_{17}, \mathfrak{p}_{31}$ lying over the small-field plaintext slot of $\mathfrak{p}'_3$. Then we can produce three small-field ciphertexts, using the three selection functions

$$\begin{aligned}(x_1, x_{15}, x_{22},\ x_3, x_{17}, x_{31}) &\mapsto (\ x_1\ ,\ x_3\ ),\\ (x_1, x_{15}, x_{22},\ x_3, x_{17}, x_{31}) &\mapsto (x_{15}\ , x_{17}),\\ (x_1, x_{15}, x_{22},\ x_3, x_{17}, x_{31}) &\mapsto (x_{22}\ , x_{31}).\end{aligned}$$

Acknowledgments

The first and second authors are supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

The third author is supported by the National Science Foundation under CAREER Award CCF-1054495, by the Alfred P. Sloan Foundation, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Sloan Foundation, DARPA or the U.S. Government.

The fourth author is supported by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and via an ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant COED–EP/I03126X, and by a Royal Society Wolfson Merit Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Commission or EPSRC.

References

[1] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO'09*, volume 5677 of *Lecture*

[9] For example, when evaluating AES homomorphically, we would have plaintext values from $\mathbb{F}_{2^8}$ even though $\mathbb{F}$ may be a larger field such as $\mathbb{F}_{2^{16}}$ or $\mathbb{F}_{2^{24}}$, etc.

[10] More precisely, $\bar{L}_i(\vec{a}) = \mathrm{Tr}_{\mathbb{F}/\mathbb{F}'}(\rho \cdot a_i)$ for some $\rho \in \mathbb{F}$ such that $\mathrm{Tr}_{\mathbb{F}/\mathbb{F}'}(\rho) = 1$, so that $\bar{L}_i(\vec{a}) = a_i$ for any $a_i \in \mathbb{F}'$, by $\mathbb{F}'$-linearity.

Notes in Computer Science, pages 595–618. Springer, 2009.

[2] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012. Available at http://eprint.iacr.org/2012/078.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS'12)*, ACM, 2012. Available at http://eprint.iacr.org/2011/277.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[5] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS'11*. IEEE Computer Society, 2011.

[6] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012. Available at http://eprint.iacr.org/2011/535.

[7] Leo Ducas and Alain Durmus. Ring-LWE in Polynomial Rings. In *PKC'12*, volume 7293 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2012. Available at http://eprint.iacr.org/2012/235

[8] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT'08*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.

[9] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC'09*, pages 169–178. ACM, 2009.

[10] Craig Gentry, Shai Halevi, Chris Peikert and Nigel P. Smart. Ring Switching in BGV-Style Homomorphic Encryption. In *SCN'12*, volume 7485 of *Lecture Notes in Computer Science*, pages 19–37. Springer, 2012. Available at http://eprint.iacr.org/2012/240.

[11] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 446-464, 2012. Available at http://eprint.iacr.org/2011/566.

[12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping for fully homomorphic encryption. In *PKC'12*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012. Available at http://eprint.iacr.org/2011/680.

[13] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012. Available at http://eprint.iacr.org/2012/099.

[14] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption In *STOC'12 Proceedings of the 44th symposium on Theory of Computing*, Pages 1219-1234. ACM, 2012.

[15] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Journal of the ACM. To appear. Preliminary version in *EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, 2010.

[16] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54, 2013. Available at `http://eprint.iacr.org/2013/293`.

[17] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[18] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Designs, Codes and Cryptography. Springer. To appear. DOI 10.1007/s10623-012-9720-4. Available at http://eprint.iacr.org/2011/133.

[19] Lawrence C. Washington. *Introduction to Cyclotomic Fields*, volume 83 of *Graduate Texts in Mathematics*. Springer, 1996.

An update on Scalable Implementation of Primitives for Homomorphic EncRyption – FPGA implementation using Simulink

David Bruce Cousins, Kurt Rohloff, Chris Peikert, Rick Schantz
Raytheon BBN Technologies, Georgia Institute of Technology
{dcousins, krohloff, schantz}@bbn.com cpeikert@cc.gatech.edu

Abstract

Accellerating the development of a practical Fully Homomorphic Encryption (FHE) scheme is the goal of the DARPA PROCEED program. For the past year, this program has had as its focus the acceleration of various aspects of the FHE concept toward practical implementation and use. FHE would be a game-changing technology to enable secure, general computation on encrypted data, e.g., on untrusted off-site hardware. However, FHE will still require several orders of magnitude improvement in computation before it will be practical for widespread use.

Recent theoretical breakthroughs demonstrated the existence of FHE schemes [1, 2], and to date much progress has been made in both algorithmic and implementation improvements. Specifically our contribution to the Proceed program has been the development of FPGA based hardware primitives to accelerate the computation on encrypted data using FHE based on lattice techniques [3]. Our project, SIPHER, has been using a state of the art tool-chain developed by Mathworks to implement VHDL code for FPGA circuits directly from Simulink models. Our baseline Homomorphic Encryption prototypes are developed directly in Matlab using the fixed point toolbox to perform the required integer arithmetic. Constant improvements in algorithms require us to be able to quickly implement them in a high level language such as Matlab. We reported on our initial results at HPEC 2011 [4]. In the past year, increases in algorithm complexity have introduced several new design requirements for our FPGA implementation. This report presents new Simulink primitives that had to be developed to deal with these new requirements.

A review of Fully and Somewhat Homomorphic Encryption

Fully Homomorphic Encryption (FHE) holds the promise to securely run arbitrary computations over encrypted data on untrusted computation hosts [2]. The general FHE concept of operations is that sensitive data is encrypted with a public key, then sent to an untrusted computation host, which can perform arbitrary computations on the encrypted data without first needing to decrypt it. It has been shown to be theoretically possible to evaluate arbitrary programs using just two special purpose FHE operations, EvalAdd and EvalMult, which at the simplest level, roughly correspond to bitwise XOR and AND gates operating on encrypted bits. A sequence of these operations is run against the encrypted data, resulting in an encryption of the output of the original program run on the unencrypted data. This encrypted result can then be sent back to the original client, who decrypts the result using its secret key. The encrypted data is protected at all times with reasonable security guarantees based on computational hardness results.

A 'Fully' Homomorphic Encryption scheme allows and unlimited number of these Eval operations to be performed. All known FHE schemes are based on computationally hard stochastic lattice theory problems, which add some noise with each operation and require a very computationally expensive "recryption" operation that is periodically run on intermediate ciphertexts to keep the noise at a level that still permits decryption. A 'Somewhat' Homomorphic scheme, on the other hand, supports several (but not unlimited) EvalMult and EvalAdd operations while preserving the correctness of decryption. In other words, SHE can schemes support secure computation for only a small subset of programs. By focusing on an SHE scheme, we can direct our research towards the implementation of efficient hardware primitives, while the FHE community develops more efficient recryption algorithms.

Recent Developments in the SIPHER SHE Scheme

Our current SHE scheme relies on operations that are generally inefficient to implement on standard CPU architectures (i.e. modular arithmetic with a large modulus). The EvalAdd and EvalMult operations for example are element wise vector adds and multiplies taken modulo some particular prime integer q. These are trivial to express using Matlab: c = mod(a+b, q) and c = mod(a.*b, q).

For convenience most of the previously published SHE and FHE implementations have used standard tools such as the GNU Multiple Precision Arithmetic Library (GMP) [5], which enable researchers to code operations using very large integers. This limits their focus to operations on CPUs and does not allow them to take advantage of specialized parallel computation hardware like FPGAs which provide highly cost-effective parallelism. Our approach to developing the FPGA code for implementing EvalAdd and EvalMult is to develop arithmetic circuits that will achieve high throughput by using parallelism and pipelining on the FPGA.

We initially develop prototype descriptions in Matlab that we re-implement in a stream-oriented hardware implementable manner in Simulink. The results of the implementations are compared to verify correctness. A conversion from Simulink to VHDL is done in a completely automated fashion using Mathwork's HDL coder. This tool chain provides us the means to develop our primitives, including cyclic VHDL based FPGA prototyping, much

Sponsored by Air Force Research Laboratory (AFRL) Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A"

faster than traditional methods. Some examples of efficiency are:

1. The Matlab and Simulink Models are driven with the same fixed point data variables, and generate the same format output, simplifying test and comparison
2. The bit width of the circuits is specified at compile time by specifying the bit width of the input data. The sizing of intermediate mathematical operations is done automatically by the fixed point toolbox. Thus many of the same models can be used for 8 bit or 64 bit inputs.
3. The resulting VHDL is vendor independent. This allows for rapid benchmarking on multiple architectures. However, hand optimization of VHDL may be required for optimum performance in order to take advantage of vendor specific IP.

Implementing fast modulo add, subtract and multiply in Simulink for HDL generation

Software implementations of modulus usually use some form of trial division to determine the remainder operation. Implementing modulus integers with large numbers of bits in an efficient manner requires the use of special numerical algorithms that have been developed, such as the Montgomery Reduction [6]. These algorithms avoid division by q, but rather scale the integers so that many of the divisions can be performed by a power of 2, requiring only simple bit shifts. Our SHE requires circuits for fast modulo addition and multiplication (to directly implement the EvalAdd and EvalMult mentioned above). In addition, our scheme relies heavily on the Chinese Remainder Transform (CRT), which can be implemented as an EvalMult, followed by an FFT [7] that uses modulo integer instead of complex arithmetic. The implementation of the FFT requires us to perform a standard radix 2 'Butterfly' operation, which uses one addition, one subtraction and one multiply, all modulo q. Thus we need to implement a modulo subtraction as well as addition.

Initially, our selection of lattice based HE led to looking at relatively modest sized modulus, on the order of twenty bits. An implementation of Montgomery Reduction based arithmetic would be relatively efficient, requiring hardware multipliers on the order of 40 bits. However, later research showed that for any reasonable security requirements our SHE scheme would need O(64) bits for our modulus. Our implementation of Montgomery arithmetic in Simulink required us to double our bit width to represent intermediate values represented in Montgomery form. We found that there is an intrinsic limitation of 128 bit width in Simulink even when using the fixed point toolbox. This meant that we could not compile our multipliers for bit widths on the order of 64 bits.

Additionally, our early arithmetic models were all designed for a single value of modulus q to be used for all operations. During the development of our SHE scheme we found that using multiple values of related moduli resulted in more efficient implementations. Thus our circuits would need to operate with multiple (but not unlimited) values of q. As a response to this we eliminated Montgomery arithmetic and take a simpler approach to modulo addition and subtraction.

Figure 1 shows the Matlab code and resulting Simulink block for performing a streaming EvalAdd when the inputs are constrained to be less than a given modulus q. The model can operate on one pair of inputs every clock cycle. The model shown does not have any additional pipeline registers for simplicity, but they can be added to the model in order to increase the maximum clock speed of the resulting VHDL, at a cost of additional pipeline stages. In our applications we expect to process streams of input on the order of several thousand entries, so this additional pipeline latency is trivial.

Figure 2 shows the Matlab and resulting Simlink block for modulo subtraction. The same comments about pipelining the circuit apply.

Modulo multiplication is a much more complicated operation, even if the input multiplier and multiplicand are bounded by q. Furthermore, we determined in our earlier work that the VHDL code generated by Simulink for large multiplications is not automatically pipelined, so the resulting multiplies severely restrict the resulting clock rates of the circuits. To address these two constraints, we adopted a recently developed interleaved modular multiplication based on a generalized Barrett reduction [8]. This multiplier has the following properties:

1) Long words of bit length L can be represented by n

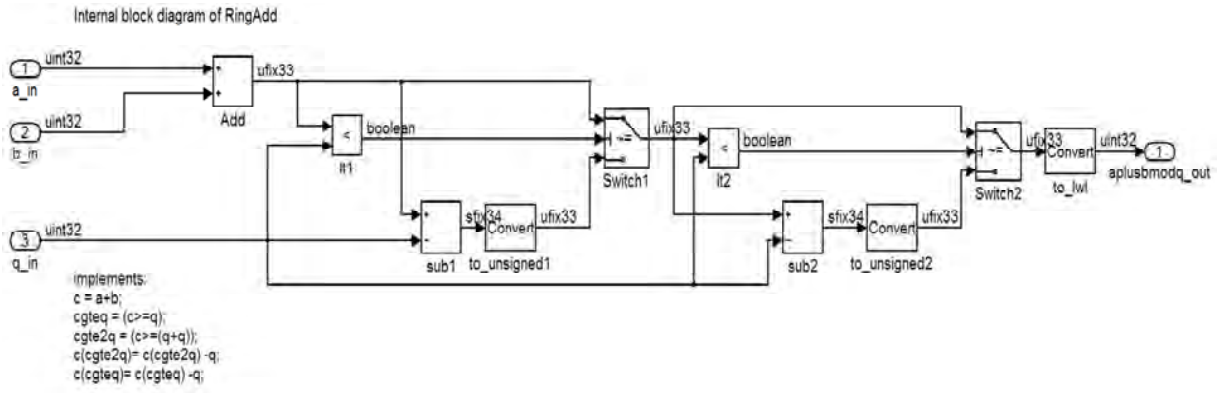


Figure 1: Internal Structure of Simulink HDL ready Modulo Add primitive.

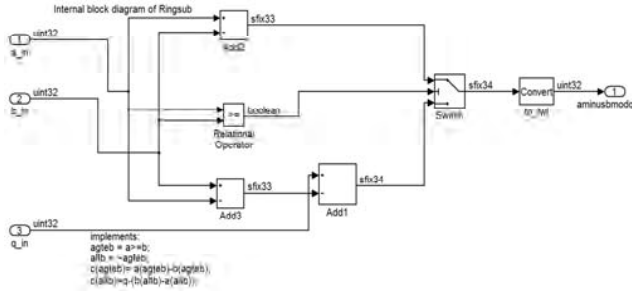


Figure 2: Internal Structure of Simulink HDL ready Modulo Subtract primitive.

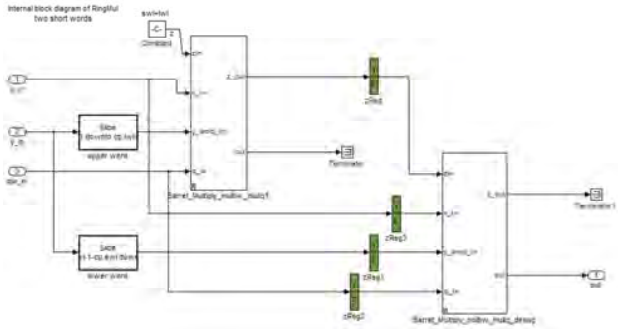


Figure 3: Top level structure of Simulink HDL ready two stage Barrett Modulo Multiply primitive.

smaller words of bit length S (i.e. four 16 bit words to represent a 64 bit modulus).

2) The multiplication is performed in n stages, where each stage performs one modulo multiplication that is L+S bits long. The stage can be pipelined to perform one modulo multiply per clock cycle.

3) Each stage has a Barrett modulus performed on the partial product, which reduces overall bit growth of the partial products to L+S. Each stage requires 3 multiplies, and all divisions required by the Barrett algorithm are implemented as simple bit shifts.

4) One circuit can support multiple moduli. All parameters that are specific to a given modulus can be stored in a table and indexed.

Figure 3 shows the structure of our resulting multiplier for a two stage operation (i.e. L = 2S). Figure 4 shows the model for a single stage in the pipeline. All stages use the same model. Again, internal pipelining in the stage is not shown.

Implementing fast CRT in Simulink for HDL generation

As mentioned earlier, our scheme uses the CRT, which relies heavily on modulo arithmetic. We have developed a Simulink model for performing a fast CRT, based on the primitives discussed above. We implemented one of the standard pipeline decimation in frequency FFT architectures, known as the Radix 2, Multiplath Delay Commutator [7]. The fundamental structure of the model is identical for a complex version that computes the standard FFT, and the modulo arithmetic version that computes the FFT portion of our CRT. The only difference is in the Simulink Model that implements the radix 2 butterfly.

Figure 3 shows the structure of this pipelined CRT. The design trades off area for processing speed. For an N point transform, log2(N) radix 2 Butterflies are required (though the last butterfly does not require multiples). Additionally, 3/2N-2 delay elements are required. The data needs to be presented to the circuit in two parallel streams, and the resulting output is in bit reverse order.

We are currently in the process of analyzing the performance of this circuit, and determining the size CRT operation that can be fit into our candidate FPGA architecture. Our analysis has shown that for high security applications we may need to perform CRT operations on vectors of up to 2^{14} in length. For such large vector sizes, an alternative design approach may be necessary in order to fit the circuit within the FGPA.

Interim Results

Our presentation will include examples of our primitives coded in Matlab and Simulink and examples of VHDL code generated by the HDL coder. We will also be able to show timing results from Modelsim based simulations of the resulting code., as well as actual timings using a Virtex 6 on the Xilinx ML605 evaluation board

References

[1] C. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic encryption scheme. In Kenneth Paterson, editor, Advances in Cryptology – EUROCRYPT 2011, volume 6632 of Lecture Notes in Computer Science, chapter 9, pages 129–148. Springer, 2011.

[2] D. Micciancio. A first glimpse of cryptography's Holy Grail. Comm. ACM 53, 3 (March 2010), 96-96.

[3] V. Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings". In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science*, chapter 1, pages

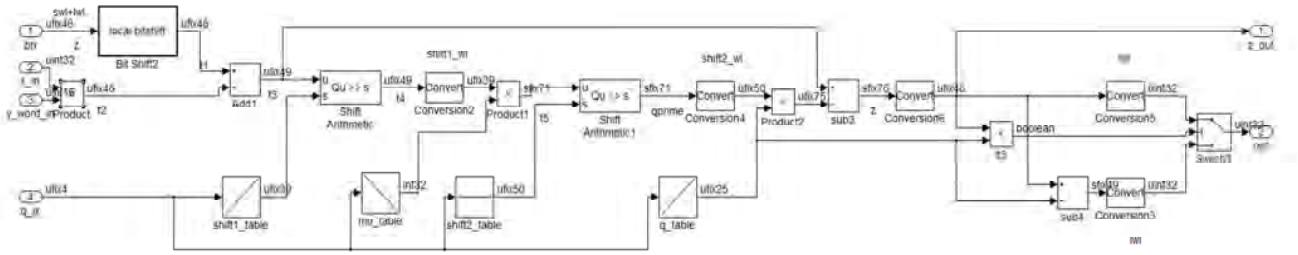


Figure 4: Internal structure of Barrett Modulo Multiply stage

1–23. Springer Berlin / Heidelberg, Berlin.

[4] D. Cousins, K. Rohloff, C. Peikert, R. Schantz "Scalable Implementation of Primitives for Homomorphic EncRyption – FPGA implementation using Simulink" 2011 High Performance Extreme Computing Workshop Sep 21-22 2011, Lexington MA

[5] http://gmplib.org/ last accessed May 14, 2012.

[6] P. L. Montgomery "Modular Multiplication Without Trial Division", *Mathematics of Computation* Vol. 44, No. 170 (Apr., 1985), pp. 519-521, American Mathematical Society.

[7] L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing.* Prentice-Hall, Inc., 1975.

[8] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster Interleaved Modular Multiplication Based on Barrett and Montgomery Reduction Methods", *IEEE Transactions on Computers*, Vol. 59, No. 12, Dec 2010.

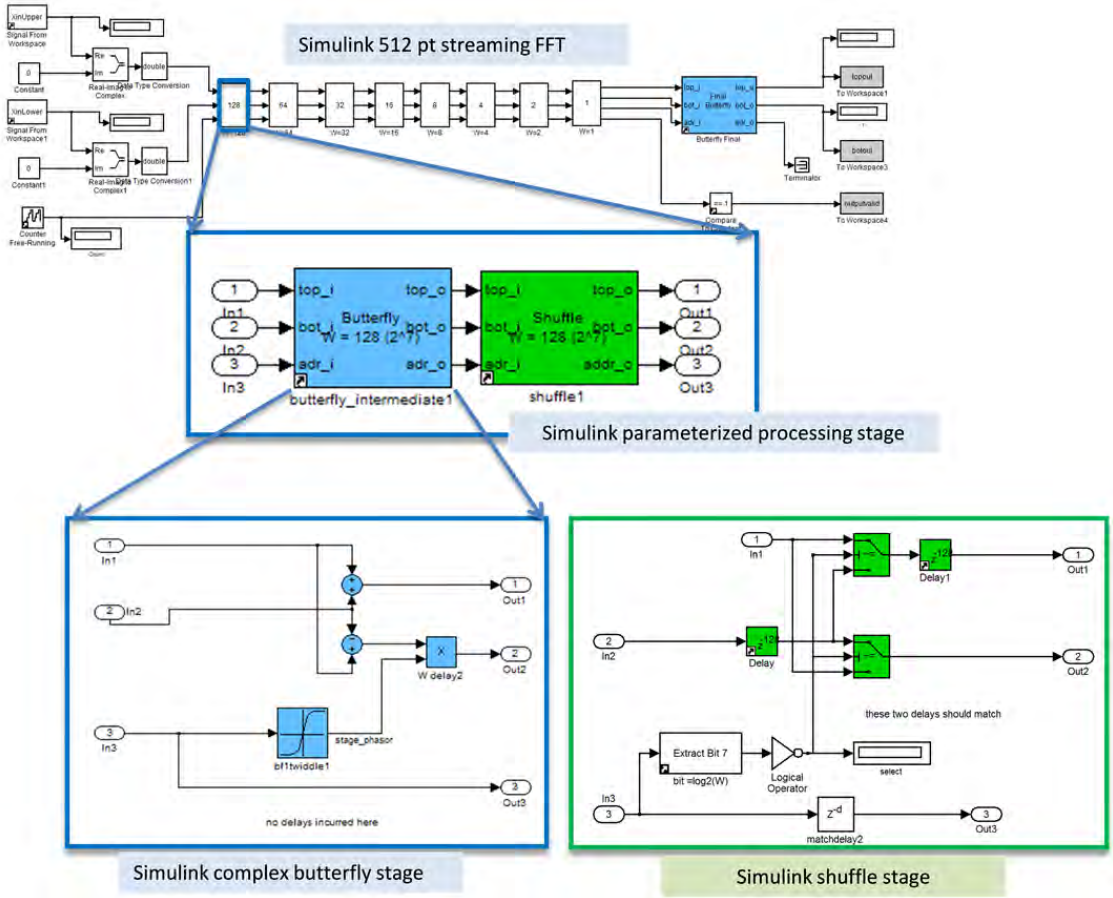


Figure 5: Simulink Pipeline FFT Structure

An extended abstract of this paper appears in the Proceedings of Eurocrypt 2012, LNCS 7237, Springer. This is the full version.

Identity-Based (Lossy) Trapdoor Functions and Applications

Mihir Bellare[1] Eike Kiltz[2] Chris Peikert[3] Brent Waters[4]

May 2012

Abstract

We provide the first constructions of identity-based (injective) trapdoor functions. Furthermore, they are lossy. Constructions are given both with pairings (DLIN) and lattices (LWE). Our lossy identity-based trapdoor functions provide an automatic way to realize, in the identity-based setting, many functionalities previously known only in the public-key setting. In particular we obtain the first deterministic and efficiently searchable IBE schemes and the first hedged IBE schemes, which achieve best possible security in the face of bad randomness. Underlying our constructs is a new definition, namely *partial* lossiness, that may be of broader interest.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. Email: mihir@cs.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grants CNS-0627779 and CCF-0915675.

[2] Horst Görtz Institut für IT-Sicherheit, Ruhr-Universität Bochum, D-44780 Bochum. Email: eike.kiltz@rub.de. URL: http://www.cits.rub.de/personen/kiltz.html.

[3] School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332-0765. Email: cpeikert@cc.gatech.edu. URL: http://www.cc.gatech.edu/~cpeikert/.

[4] Department of Computer Science, University of Texas at Austin, 1616 Guadalupe, Suite 2.408, Austin, TX 78701. Email: bwaters@cs.utexas.edu. URL: http://userweb.cs.utexas.edu/~bwaters/.

1 Introduction

A trapdoor function F specifies, for each public key pk, an injective, *deterministic* map F_{pk} that can be inverted given an associated secret key (trapdoor). The most basic measure of security is one-wayness. The canonical example is RSA [55].

Suppose there is an algorithm that generates a "fake" public key pk^* such that F_{pk^*} is no longer injective but has image much smaller than its domain and, moreover, given a public key, you can't tell whether it is real or fake. Peikert and Waters [52] call such a TDF lossy. Intuitively, F_{pk} is close to a function F_{pk^*} that provides information-theoretic security. Lossiness implies one-wayness [52].

Lossy TDFs have quickly proven to be a powerful tool. Applications include IND-CCA [52], deterministic [16], hedged [8] and selective-opening secure public-key encryption [10]. Lossy TDFs can be constructed from DDH [52], QR [35], DLIN [35], DBDH [24], LWE [52] and HPS (hash proof systems) [40]. RSA was shown in [44] to be lossy under the Φ-hiding assumption of [26], leading to the first proof of security of RSA-OAEP [13] without random oracles.

Lossy TDFs and their benefits belong, so far, to the realm of public-key cryptography. The purpose of this paper is to bring them to identity-based cryptography, defining and constructing identity-based TDFs (IB-TDFs), both one-way and lossy. We see this as having two motivations, one more theoretical, the other more applied, yet admittedly both foundational, as we discuss before moving further.

Theoretical angle. Trapdoor functions are the primitive that began public key cryptography [31, 55]. Public-key encryption was built from TDFs. (Via hardcore bits.) Lossy TDFs enabled the first DDH and lattice (LWE) based TDFs [52].

It is striking that identity-based cryptography developed entirely differently. The first realizations of IBE [21, 30, 58] directly used randomization and were neither underlain by, nor gave rise to, any IB-TDFs.

We ask whether this asymmetry between the public-key and identity-based worlds (TDFs in one but not the other) is inherent. This seems to us a basic question about the nature of identity-based cryptography that is worth asking and answering.

Application angle. Is there anything here but idle curiosity? IBE has already been achieved *without* IB-TDFs, so why go backwards to define and construct the latter? The answer is that *losssy* IB-TDFs enable new applications that we do not know how to get in other ways.

Stepping back, identity-based cryptography [59] offers several advantages over its public-key counterpart. Key management is simplified because an entity's identity functions as their public key. Key revocation issues that plague PKI can be handled in alternative ways, for example by using identity+date as the key under which to encrypt to identity [21]. There is thus good motivation to go beyond basics like IBE [21, 30, 58, 17, 18, 62, 36] and identity-based signatures [11, 32] to provide identity-based counterparts of other public-key primitives.

Furthermore we would like to do this in a systematic rather than ad hoc way, leading us to seek tools that enable the transfer of multiple functionalities in relatively blackbox ways. The applications of lossiness in the public-key realm suggest that lossy IBTDFs will be such a tool also in the identity-based realm. As evidence we apply them to achieve identity-based deterministic encryption and identity-based hedged encryption. The first, the counterpart of deterministic public-key encryption [7, 16], allows efficiently searchable identity-based encryption of database entries while maintaining the maximal possible privacy, bringing the key-management benefits of the identity-based setting to this application. The second, counterpart of hedged symmetric and public-key encryption [56, 8], makes IBE as resistant as possible in the face of low-quality randomness, which is important given the widespread deployment of IBE and the real danger of bad-randomness based attacks evidenced by the ones on the Sony Playstation and Debian Linux. We hope that our framework will facilitate further such transfers.

We clarify that the solutions we obtain are not practical but they show that the security goals can be achieved in principle, which was not at all clear prior to our work. Allowed random oracles, we can give solutions that are much more efficient and even practical.

CONTRIBUTIONS IN BRIEF. We define IB-TDFs and two associated security notions, one-wayness and lossiness, showing that the second implies the first.

The first wave of IBE schemes was from pairings [21, 58, 17, 18, 62, 61] but another is now emerging from lattices [36, 29, 2, 3]. We aim accordingly to reach our ends with either route and do so successfully. We provide lossy IB-TDFs from a standard pairings assumption, namely the Decision Linear (DLIN) assumption of [19]. We also provide IB-TDFs based on Learning with Errors (LWE) [53], whose hardness follows from the worst-case hardness of certain lattice-related problems [53, 50]. (The same assumption underlies lattice-based IBE [36, 29, 2, 3] and public-key lossy TDFs [52].) None of these results relies on random oracles.

Existing work brought us closer to the door with lattices, where one-way IB-TDFs can be built by combining ideas from [36, 29, 2]. Based on techniques from [50, 45] we show how to make them lossy. With pairings, however it was unclear how to even get a one-way IB-TDF, let alone one that is lossy. We adapt the matrix-based framework of [52] so that by populating matrix entries with ciphertexts of a very special kind of *anonymous* IBE scheme it becomes possible to implicitly specify per-identity matrices defining the function. No existing anonymous IBE has the properties we need but we build one that does based on methods of [23]. Our results with pairings are stronger because the lossy branches are universal hash functions which is important for applications.

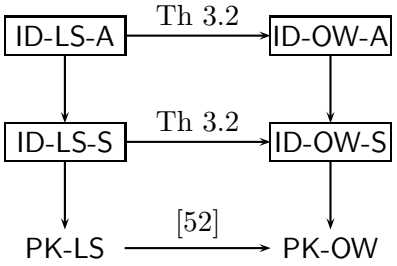
Public-key lossy TDFs exist aplenty and IBE schemes do as well. It is natural to think one could easily combine them to get IB-TDFs. We have found no simple way to do this. Ultimately we do draw from both sources for techniques but our approaches are intrusive. Let us now look at our contributions in more detail.

NEW PRIMITIVES AND DEFINITIONS. Public parameters *pars* and an associated master secret key having been chosen, an IB-TDF F associates to any identity a map $F_{pars,id}$, again injective and deterministic, inversion being possible given a secret key derivable from id via the master secret key. One-wayness means F_{pars,id^*} is hard to invert on random inputs for an adversary-specified challenge identity id^*. Importantly, as in IBE, this must hold even when the adversary may obtain, via a key-derivation oracle, a decryption key for any non-challenge identity of its choice [21]. This key-derivation capability contributes significantly to the difficulty of realizing the primitive. As with IBE, security may be selective (the adversary must specify id^* before seeing *pars*) [28] or adaptive (no such restriction) [21].

The most direct analog of the definition of lossiness from the public-key setting would ask that there be a way to generate "fake" parameters $pars^*$, indistinguishable from the real ones, such that F_{pars^*,id^*} is lossy (has image smaller than domain). In the selective setting, the fake parameter generation algorithm Pg^* can take id^* as input, making the goal achievable at least in principle, but in the adaptive setting it is impossible to achieve, since, with id^* not known in advance, Pg^* is forced to make $F_{pars^*,id}$ lossy for all id, something the adversary can immediately detect using its key-derivation oracle.

We ask whether there is an adaptation of the definition of lossiness that is achievable in the adaptive case while sufficing for applications. Our answer is a definition of δ-*lossiness*, a metric of partial lossiness parameterized by the probability δ that F_{pars^*,id^*} is lossy. The definition is unusual, involving an adversary advantage that is the difference, not of two probabilities as is common in cryptographic metrics, but of two differently weighted ones. We will achieve selective lossiness with degree $\delta = 1$, but in the adaptive case the best possible is degree $1/\text{poly}$ with the polynomial depending on the number of key-derivation queries of the adversary, and this what we will achieve. We show that lossiness with degree δ implies one-wayness, in both the selective and adaptive settings, as long as δ is at least $1/\text{poly}$.

In summary, in the identity-based setting (ID) there are two notions of security, one-wayness (OW) and lossiness (LS), each of which could be selective (S) or adaptive (A), giving rise to four kinds of IB-TDFs. The left side of Figure 1 shows how they relate to each other and to the two kinds of TDFs —OW and LS— in the public-key setting (PK). The un-annotated implications are trivial, ID-LS-A $\rightarrow$ ID-LS-S meaning that δ-lossiness of the first type implies δ-lossiness of the other for all δ. It is not however via this implication that we achieve ID-LS-S, for, as the table shows, we achieve it with degree higher than



Primitive	δ	Achieved under
ID-LS-A	$1/\text{poly}$	DLIN, LWE
ID-LS-S	1	DLIN, LWE

Figure 1: Types of TDFs based on setting (PK=Public-key, ID=identity-based), security (OW=one-way, LS=loss) and whether the latter is selective (S) or adaptive (A). An arrow A $\rightarrow$ B in the diagram on the left means that TDF of type B is implied by (can be constructed from) TDF of type A. Boxed TDFs are the ones we define and construct. The table on the right shows the δ for which we prove δ-lossiness and the assumptions used. In both the S and A settings the δ we achieve is best possible and suffices for applications.

ID-LS-A.

Closer Look. One's first attempt may be to build an IB-TDF from an IBE scheme. In the random oracle (RO) model, this can be done by a method of [9], namely specify the coins for the IBE scheme by hashing the message with the RO. It is entirely unclear how to turn this into a standard model construct and it is also unclear how to make it lossy.

To build ID-TDFs from lattices we consider starting from the public-key TDF of [52] (which is already lossy) and trying to make it identity-based, but it is unclear how to do this. However, Gentry, Peikert and Vaikuntanathan (GPV) [36] showed that the function $g_{\mathbf{A}} \colon B_{\alpha}^{n+m} \rightarrow \mathbb{Z}_q^n$ defined by $g_{\mathbf{A}}(\mathbf{x}, \mathbf{e}) = \mathbf{A}^T \cdot \mathbf{x} + \mathbf{e}$ is a TDF for appropriate choices of the domain and parameters, where matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a uniformly random public key which is constructed together with a trapdoor as for example in [4, 5, 46]. We make this function identity-based using the trapdoor extension and delegation methods introduced by Cash, Hofheinz, Kiltz and Peikert [29], and improved in efficiency by Agrawal, Boneh and Boyen [2] and Micciancio and Peikert [46]. Finally, we obtain a lossy IB-TDF by showing that this construction is already lossy.

With pairings there is no immediate way to get an IB-TDF that is even one-way, let alone lossy. We aim for the latter, there being no obviously simpler way to get the former. In the selective case we need to ensure that the function is lossy on the challenge identity id^* yet injective on others, this setup being indistinguishable from the one where the function is always injective. Whereas the matrix diagonals in the construction of [52] consisted of ElGamal ciphertexts, in ours they are ciphertexts for identity id^* under an anonymous IBE scheme, the salient property being that the "anonymity" property should hide whether the underlying ciphertext is to id^* or is a random group element. Existing anonymous IBE schemes, in particular that of Boyen and Waters (BW) [23], are not conducive and we create a new one. A side benefit is a new anonymous IBE scheme with ciphertexts and private keys having one less group element than BW but still proven secure under DLIN.

A method of Boneh and Boyen [17] can be applied to turn selective into adaptive security but the reduction incurs a factor that is equal to the size of the identity space and thus ultimately exponential in the security parameter, so that adaptive security according to the standard asymptotic convention would not have been achieved. To achieve it, we want to be able to "program" the public parameters so that they will be lossy on about a $1/Q$ fraction of "random-ish" identities, where Q is the number of key-derivation queries made by the attacker. Ideally, with probability around $1/Q$ all of (a successful) attacker's queries will land outside the lossy identity-space, but the challenge identity will land inside it so that we achieve δ-lossiness with δ around $1/Q$.

This sounds similar to the approach of Waters [62] for achieving adaptively secure IBE but there are some important distinctions, most notably that the technique of Waters is information-theoretic while ours is of necessity computational, relying on the DLIN assumption. In the reduction used by Waters the

partitioning of the identities into two classes was based solely on the reduction algorithm's internal view of the public parameters; the parameters themselves were distributed independently of this partitioning and thus the adversary view was the same as in a normal setup. In contrast, the partitioning in our scheme will actually directly affect the parameters and how the system behaves. This is why we must rely on a computational assumption to show that the partitioning in undetectable. A key novel feature of our construction is the introduction of a system that will produce lossy public parameters for about a $1/Q$ fraction of the identities.

Applications. Deterministic PKE is a TDF providing the best possible privacy subject to being deterministic, a notion called PRIV that is much stronger than one-wayness [7]. An application is encryption of database records in a way that permits logarithmic-time search, improving upon the linear-time search of PEKS [20]. Boldyreva, Fehr and O'Neill [16] show that lossy TDFs whose lossy branch is a universal hash (called universal lossy TDFs) achieve (via the LHL [15, 39]) PRIV-security for message sequences which are blocksources, meaning each message has some min-entropy even given the previous ones, which remains the best result without ROs. Deterministic IBE and the resulting efficiently-searchable IBE are attractive due to the key-management benefits. We can achieve them because our DLIN-based lossy IB-TDFs are also universal lossy. (This is not true, so far, for our LWE based IB-TDFs.)

To provide IND-CPA security in practice, IBE relies crucially on the availability of fresh, high-quality randomness. This is fine in theory but in practice RNGs (random number generators) fail due to poor entropy gathering or bugs, leading to prominent security breaches [37, 38, 25, 49, 48, 1, 63, 33]. Expecting systems to do a better job is unrealistic. Hedged encryption [8] takes poor randomness as a fact of life and aims to deliver best possible security in the face of it, providing privacy as long as the message together with the "randomness" have some min-entropy. Hedged PKE was achieved in [8] by combining IND-CPA PKE with universal lossy TDFs. We can adapt this to IBE and combine existing (randomized) IBE schemes with our DLIN-based universal lossy IB-TDFs to achieved hedged IBE. This is attractive given the widespread use of IBE in practice and the real danger of randomness failures.

Both applications are for the case of selective security. We do not achieve them in the adaptive case.

Related Work. A number of papers have studied security notions of trapdoor functions beyond traditional one-wayness. Besides lossiness [52] there is Rosen and Segev's notion of correlated-product security [57], and Canetti and Dakdouk's extractable trapdoor functions [27]. The notion of adaptive one-wayness for tag-based trapdoor functions from Kiltz, Mohassel and O'Neill [43] can be seen as the special case of our selective IB-TDF in which the adversary is denied key-derivation queries. Security in the face of these queries was one of the main difficulties we faced in realizing IB-TDFs.

Organization. We define IB-TDFs, one-wayness and δ-lossiness in Section 2. We also define extended IB-TDFs, an abstraction that will allow us to unify and shorten the analyses for the selective and adaptive security cases. In Section 3 we show that δ-lossiness implies one-wayness as long as δ is at least $1/\operatorname{poly}$. This allows us to focus on achieving δ-lossiness. In Section 4 we provide our pairing-based schemes and in Appendix 5 our lattice-based schemes. In Appendix B we sketch how to apply δ-lossy IB-TDFs to achieve deterministic and hedged IBE.

Subsequent work. Escala, Herranz, Libert and Ráfols [34] provide an alternative definition of partial lossiness based on which they achieve deterministic, PRIV-secure IBE for blocksources, and hedged IBE, in the adaptive case, which answers an open question from our work. They also define and construct hierarchical identity-based (lossy) trapdoor functions.

2 Definitions

Notation and conventions. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes the number of its coordiates and $\mathbf{x}[i]$ denotes its i-th coordinate. Coordinates may be numbered $1, \ldots, |\mathbf{x}|$ or $0, \ldots, |\mathbf{x}| - 1$ as convenient. A string x is identified with a vector over $\{0,1\}$ so that $|x|$ denotes its length and $x[i]$ its i-th bit. The

proc Initialize(id) // $\mathrm{OW}_{\mathsf{F}}, \mathrm{Real}_{\mathsf{F}}$	**proc Initialize**(id) // $\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$
$(pars, msk) \xleftarrow{\$} \mathsf{F.Pg}$; $IS \leftarrow \emptyset$; $id^* \leftarrow id$	$(pars, msk) \xleftarrow{\$} \mathsf{LF.Pg}(id)$; $IS \leftarrow \emptyset$; $id^* \leftarrow id$
Return $pars$	Return $pars$
proc GetDK(id) // $\mathrm{OW}_{\mathsf{F}}, \mathrm{Real}_{\mathsf{F}}$	**proc GetDK**(id) // $\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$
$IS \leftarrow IS \cup \{id\}$	$IS \leftarrow IS \cup \{id\}$
$dk \leftarrow \mathsf{F.Kg}(pars, msk, id)$	$dk \leftarrow \mathsf{LF.Kg}(pars, msk, id)$
Return dk	Return dk
proc Ch(id) // OW_{F}	**proc Ch**(id) // $\mathrm{Real}_{\mathsf{F}}, \mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$
$id^* \leftarrow id$; $x \xleftarrow{\$} \mathsf{InSp}$	$id^* \leftarrow id$
$y \leftarrow \mathsf{F.Ev}(pars, id^*, x)$	**proc Finalize**(d') // $\mathrm{Real}_{\mathsf{F}}$
Return y	Return $((d' = 1)$ and $(id^* \notin IS))$
proc Finalize(x') // OW_{F}	**proc Finalize**(d') // $\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$
Return $((x' = x)$ and $(id^* \notin IS))$	Return $((d' = 1)$ and $(id^* \notin IS)$ and $(\lambda(\mathsf{F.Ev}(pars, id^*, \cdot)) \geq \ell))$

Figure 2: Games defining one-wayness and δ-lossiness of IBTDF F with associated sibling LF.

empty string is denoted ε. If S is a set then $|S|$ denotes its size, S^a denotes the set of a-vectors over S, $S^{a\times b}$ denotes the set of a by b matrices with entries in S, and so on. The (i,j)-th entry of a 2 dimensional matrix $\mathbf{M}$ is denoted $\mathbf{M}[i,j]$ and the (i,j,k)-th entry of a 3 dimensional matrix $\mathbf{M}$ is denoted $\mathbf{M}[i,j,k]$. If $\mathbf{M}$ is a n by μ matrix then $\mathbf{M}[j,\cdot]$ denotes the vector $(\mathbf{M}[j,1],\ldots,\mathbf{M}[j,\mu])$. If $a = (a_1,\ldots,a_n)$ then $(a_1,\ldots,a_n) \leftarrow a$ means we parse a as shown. Unless otherwise indicated, an algorithm may be randomized. By $y \xleftarrow{\$} A(x_1,x_2,\ldots)$ we denote the operation of running A on inputs $x_1,x_2,\ldots$ and fresh coins and letting y denote the output. We denote by $[A(x_1,x_2,\ldots)]$ the set of all possible outputs of A on inputs $x_1,x_2,\ldots$. The (Kronecker) delta function Δ is defined by $\Delta(a,b) = 1$ if $a = b$ and 0 otherwise. If a,b are equal-length vectors of reals then $\langle a,b\rangle = a[1]b[1] + \cdots + a[|a|]b[|b|]$ denotes their inner product.

GAMES. A game —look at Figure 2 for an example— has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. To execute a game G is executed with an adversary A means to run the adversary and answer its oracle queries by the corresponding procedures of G. The adversary must make exactly one query to **Initialize**, this being its first oracle query. (This means the adversary can give **Initialize** an input, an extension of the usual convention [14].) It must make exactly one query to **Finalize**, this being its last oracle query. The reply to this query, denoted G^A, is called the output of the game, and we let "G^A" denote the event that this game output takes value true. Boolean flags are assumed initialized to false.

IBTDFS. An *identity-based trapdoor function* (IBTDF) is a tuple $\mathsf{F} = (\mathsf{F.Pg}, \mathsf{F.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ of algorithms with associated input space InSp and identity space IDSp. The parameter generation algorithm $\mathsf{F.Pg}$ takes no input and returns common parameters $pars$ and a master secret key msk. On input $pars, msk, id$, the key generation algorithm $\mathsf{F.Kg}$ produces a decryption key dk for identity id. For any $pars$ and $id \in \mathsf{IDSp}$, the *deterministic* evaluation algorithm $\mathsf{F.Ev}$ defines a function $\mathsf{F.Ev}(pars, id, \cdot)$ with domain InSp. We require *correct inversion*: For any $pars$, any $id \in \mathsf{IDSp}$ and any $dk \in [\mathsf{F.Kg}(pars, id)]$, the deterministic inversion algorithm $\mathsf{F.Ev}^{-1}$ defines a function that is the inverse of $\mathsf{F.Ev}(pars, id, \cdot)$, meaning $\mathsf{F.Ev}^{-1}(pars, id, dk, \mathsf{F.Ev}(pars, id, x)) = x$ for all $x \in \mathsf{InSp}$.

E-IBTDF. To unify and shorten the selective and adaptive cases of our analyses it is useful to define and specify a more general primitive. An extended IBTDF (E-IBTDF) $\mathsf{E} = (\mathsf{E.Pg}, \mathsf{E.Kg}, \mathsf{E.Ev}, \mathsf{E.Ev}^{-1})$ consists of four algorithms that are just like the ones for an IBTDF except that $\mathsf{F.Pg}$ takes an additional *auxiliary* input from an auxiliary input space AxSp. Fixing a particular auxiliary input $aux \in \mathsf{AxSp}$ for $\mathsf{F.Pg}$ results in an IBTDF scheme that we denote $\mathsf{E}(aux)$ and call the IBTDF induced by aux. Not all

these induced schemes need, however, satisfy the correct inversion requirement. If the one induced by *aux* does, we say that *aux* grants invertibility. Looking ahead we will build an E-IBTDF and then obtain our IBTDF as the one induced by a particular auxiliary input, the other induced schemes being the basis of the siblings and being used in the proof.

ONE-WAYNESS. One-wayness of IBTDF $\mathsf{F} = (\mathsf{F.Pg}, \mathsf{F.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ is defined via game OW_{F} of Figure 2. The adversary is allowed only one query to its challenge oracle **Ch**. The advantage of such an adversary I is $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(I) = \Pr\left[\mathrm{OW}^{I}_{\mathsf{F}}\right]$.

SELECTIVE VERSUS ADAPTIVE ID. We are interested in both these variants for all the notions we consider. To avoid a proliferation of similar definitions, we capture the variants instead via different adversary classes relative to the same game. To exemplify, consider game OW_{F} of Figure 2. Say that an adversary A is *selective-id* if the identity *id* in its queries to **Initialize** and **Ch** is always the same, and say it is *adaptive-id* if this is not necessarily true. Selective-id security for one-wayness is thus captured by restricting attention to selective-id adversaries and full (adaptive-id) security by allowing adaptive-id adversaries. Now, adopt the same definitions of selective and adaptive adversaries relative to *any* game that provides procedures called **Initialize** and **Ch**, regardless of how these procedures operate. In this way, other notions we will introduce, including partial lossiness defined via games also in Figure 2, will automatically have selective-id and adaptive-id security versions.

PARTIAL LOSSINESS. We first provide the formal definitions and later explain them and their relation to standard definitions. If f is a function with domain a (non-empty) set $\mathrm{Dom}(f)$ then its image is $\mathrm{Im}(f) = \{\, f(x) \,:\, x \in \mathrm{Dom}(f) \,\}$. We define the *lossiness* $\lambda(f)$ of f via

$$\lambda(f) = \lg \frac{|\mathrm{Dom}(f)|}{|\mathrm{Im}(f)|} \qquad \text{or equivalently} \qquad |\mathrm{Im}(f)| = |\mathrm{Dom}(f)| \cdot 2^{-\lambda(f)} \ .$$

We say that f is ℓ-lossy if $\lambda(f) \geq \ell$. Let IBTDF $\mathsf{F} = (\mathsf{F.Pg}, \mathsf{F.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ be an IBTDF with associated input space InSp and identity space IDSp. A *sibling* for F is an E-IBTDF $\mathsf{LF} = (\mathsf{LF.Pg}, \mathsf{LF.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ whose evaluation and inversion algorithms, as the notation indicates, are those of F and whose auxiliary input space is IDSp. Algorithm $\mathsf{LF.Pg}$ will use this input in the selective-id case and ignore it in the adaptive-id case. Consider games $\mathrm{Real}_{\mathsf{F}}$ and $\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$ of Figure 2. The first uses the real parameter and key-generation algorithms while the second uses the sibling ones. A los-adversary A is allowed just one **Ch** query, and the games do no more than record the challenge identity id^*. The advantage of the adversary is *not*, as usual, the difference in the probabilities that the games return true, but is instead parameterized by a probability $\delta \in [0, 1]$and defined via

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A) = \delta \cdot \Pr\left[\mathrm{Real}^{A}_{\mathsf{F}}\right] - \Pr\left[\mathrm{Lossy}^{A}_{\mathsf{F},\mathsf{LF},\ell}\right] . \tag{1}$$

DISCUSSION. The PW [52] notion of lossy TDFs in the public-key setting asks for an alternative "sibling" key-generation algorithm, producing a public key but no secret key, such that two conditions hold. The first, which is combinatorial, asks that the functions defined by sibling keys are lossy. The second, which is computational, asks that real and sibling keys are indistinguishable. The first change for the IB setting is that one needs an alternative parameter generation algorithm which produces not only *pars* but a master secret key *msk*, and an alternative key-generation algorithm that, based on *msk*, can issue decryption keys to users. Now we would like to ask that the function $\mathsf{F.Ev}(\mathit{pars}, id^*, \cdot)$ be lossy on the challenge identity id^* when *pars* is generated via $\mathsf{LF.Pg}$, but, in the adaptive-id case, we do not know id^* in advance. Thus the requirement is made via the games.

We would like to define the advantage normally, meaning with $\delta = 1$, but the resulting notion is not achievable in the adaptive-id case. (This can be shown via attack.) With the relaxation, a low (close to zero) advantage means that the probability that the adversary finds a lossy identity id^* and then outputs 1 is less than the probability that it merely outputs 1 by a factor not much less than δ. Roughly, it means that a δ fraction of identities are lossy. The advantage represents the computational loss while δ represents a necessary information-theortic loss.

IBE. Recall that an IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Pg}, \mathsf{IBE.Kg}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ is a tuple of algorithms with associated message space InSp and identity space IDSp. The parameter generation algorithm $\mathsf{IBE.Pg}$ takes no input and returns common parameters *pars* and a master secret key *msk*. On input $pars, msk, id$, the key generation algorithm $\mathsf{IBE.Kg}$ produces a decryption key dk for identity id. On input $pars$, $id \in \mathsf{IDSp}$ and a message $M \in \mathsf{InSp}$ the encryption algorithm $\mathsf{IBE.Enc}$ returns a ciphertext. The decryption algorithm $\mathsf{IBE.Dec}$ is deterministic. The scheme has decryption error ϵ if $\Pr[\mathsf{IBE.Dec}(pars, id, dk, \mathsf{IBE.Enc}(pars, id, M)) \neq M] \leq \epsilon$ for all $pars$, all $id \in \mathsf{IDSp}$, all $dk \in [\mathsf{F.Kg}(pars, id)]$ and all $M \in \mathsf{InSp}$. We say that IBE is deterministic if $\mathsf{IBE.Enc}$ is deterministic. A deterministic IBE scheme is identical to an IBTDF.

3 Implications of Partial Lossiness

Theorem 3.2 shows that partial lossiness implies one-wayness. We discuss other applications in Appendix B. We first need a simple lemma.

Lemma 3.1 *Let f be a function with non-empty domain $\mathrm{Dom}(f)$. Then for any adversary A*

$$\Pr[A(y) = x \,:\, x \xleftarrow{\$} \mathrm{Dom}(f)\,;\, y \leftarrow f(x)] \;\leq\; 2^{-\lambda(f)}\,. \quad \blacksquare$$

Proof of Lemma 3.1: For $y \in \mathrm{Im}(f)$ let $f^{-1}(y)$ be the set of all $x \in \mathrm{Dom}(f)$ such that $f(x) = y$. The probability in question is

$$\sum_{y \in \mathrm{Im}(f)} \Pr[\, A(y) = x \mid f(x) = y \,] \cdot \Pr[\, f(x) = y \,] \;\leq\; \sum_{y \in \mathrm{Im}(f)} \frac{1}{|f^{-1}(y)|} \cdot \frac{|f^{-1}(y)|}{|\mathrm{Dom}(f)|} \;=\; \frac{|\mathrm{Im}(f)|}{|\mathrm{Dom}(f)|} \;=\; 2^{-\lambda(f)}$$

where the probability is over x chosen at random from $\mathrm{Dom}(f)$ and the coins of A if any. (Since A is unbounded, it can be assumed wlog to be deterministic.) $\blacksquare$

Theorem 3.2 [δ-lossiness implies one-wayness] *Let $\mathsf{F} = (\mathsf{F.Pg}, \mathsf{F.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ be a IBTDF with associated input space InSp. Let $\mathsf{LF} = (\mathsf{LF.Pg}, \mathsf{LF.Kg}, \mathsf{F.Ev}, \mathsf{F.Ev}^{-1})$ be a lossy sibling for F. Let $\delta > 0$ and let $\ell \geq 0$. Then for any ow-adversary I there is a los-adversary A such that*

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(I) \leq \frac{\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A) + 2^{-\ell}}{\delta}\,. \tag{2}$$

The running time of A is that if I plus the time for a computation of $\mathsf{F.Ev}$. If I is a selective adversary then so is A. $\blacksquare$

In asymptotic terms, the theorem says that δ-lossiness implies one-wayness as long as δ^{-1} is bounded above by a polynomial in the security parameter and ℓ is super-logarithmic. This means δ need only be non-negligible. The last sentence of the theorem, saying that if I is selective then so is A, is important because it says that the theorem covers both the selective and adaptive security cases, meaning selective δ-lossiness implies selective one-wayness and adaptive δ-lossiness implies adaptive one-wayness.

Proof of Theorem 3.2: Adversary A runs I. When I makes query $\mathbf{Initialize}(id)$, adversary A does the same, obtaining *pars* and returning this to I. Adversary A answers I's queries to its **GetDK** oracle via its own oracle of the same name. When I makes its (single) **Ch** query id^*, adversary A also makes query $\mathbf{Ch}(id^*)$. Additionally, it picks x at random from InSp and returns $y = \mathsf{F.Ev}(pars, id^*, x)$ to I. The latter eventually halts with output x'. Adversary A returns 1 if $x' = x$ and 0 otherwise. By design we clearly have $\Pr\left[\mathrm{Real}^A_{\mathsf{F}}\right] = \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(I)$. But game $\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}$ returns true only if $\mathsf{F.Ev}(pars, id^*, \cdot)$ is ℓ-lossy, in which case the probability that $x = x'$ is small by Lemma 3.1. In detail, assuming wlog that I never queries id^* to **GetDK**, we have

$$\begin{aligned}
\Pr\left[\mathrm{Lossy}^A_{\mathsf{F},\mathsf{LF},\ell}\right] &= \Pr\left[\, x = x' \mid \lambda(\mathsf{F.Ev}(pars, id^*, \cdot)) \geq \ell \,\right] \cdot \Pr\left[\, \lambda(\mathsf{F.Ev}(pars, id^*, \cdot)) \geq \ell \,\right] \\
&\leq \Pr\left[\, x = x' \mid \lambda(\mathsf{F.Ev}(pars, id^*, \cdot)) \geq \ell \,\right] \;\leq\; 2^{-\ell}\,,
\end{aligned}$$

the last inequality by Lemma 3.1 applied to the function $f = \mathsf{F.Ev}(pars, id^*, \cdot)$. From Equation (1) we have

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A) \;=\; \delta \cdot \Pr\left[\,\mathrm{Real}^A_{\mathsf{F}}\,\right] - \Pr\left[\,\mathrm{Lossy}^A_{\mathsf{F},\mathsf{LF},\ell}\,\right] \;\geq\; \delta \cdot \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(I) - 2^{-\ell}\ .$$

Equation (2) follows. ▌ In Section B we discuss the application to deterministic and hedged IBE.

4 IB-TDFs from pairings

In Section 3 we show that δ-lossiness implies one-wayness in both the selective and adaptive cases. We now show how to achieve δ-lossiness using pairings.

SETUP. Throughout we fix a bilinear map $\mathbf{e}$: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}, \mathbb{G}_T$ are groups of prime order p. By $\mathbf{1}, \mathbf{1}_T$ we denote the identity elements of $\mathbb{G}, \mathbb{G}_T$, respectively. By $\mathbb{G}^* = \mathbb{G} - \{\mathbf{1}\}$ we denote the set of generators of $\mathbb{G}$. The advantage of a dlin-adversary B is

$$\mathbf{Adv}^{\mathrm{dlin}}(B) \;=\; 2\Pr[\mathrm{DLIN}^B] - 1\ ,$$

where game DLIN is as follows. The **Initialize** procedure picks $g, \hat{g}$ at random from $\mathbb{G}^*$, s at random from $\mathbb{Z}_p^*$, $\hat{s}$ at random from $\mathbb{Z}_p$ and X at random from $\mathbb{G}$. It picks a random bit b. If $b = 1$ it lets $T \leftarrow X^{s+\hat{s}}$ and otherwise picks T at random from $\mathbb{G}$. It returns $(g, \hat{g}, g^s, \hat{g}^{\hat{s}}, X, T)$ to the adversary B. The adversary outputs a bit b' and **Finalize**, given b' returns true if $b = b'$ and false otherwise. For integer $\mu \geq 1$, vectors $\mathbf{U} \in \mathbb{G}^{\mu+1}$ and $\mathbf{y} \in \mathbb{Z}_p^{\mu+1}$, and vector $id \in \mathbb{Z}_p^{\mu}$ we let

$$\overline{id} = (1, id[1], \ldots, id[\mu]) \in \mathbb{Z}_p^{\mu+1} \qquad \text{and} \qquad \mathcal{H}(\mathbf{U}, id) = \textstyle\prod_{k=0}^{\mu} \mathbf{U}[k]^{\overline{id}[k]}\ .$$

$\mathcal{H}$ is the BB hash function [17] when $\mu = 1$, and the Waters' one [23] when $\mathsf{IDSp} = \{0,1\}^{\mu}$ and an $id \in \mathsf{IDSp}$ is viewed as a μ-vector over $\mathbb{Z}_p$. We also let

$$f(\mathbf{y}, id) = \textstyle\sum_{k=0}^{\mu} \mathbf{y}[k]\overline{id}[k] \qquad \text{and} \qquad \overline{f}(\mathbf{y}, id) = f(\mathbf{y}, id) \bmod p\ .$$

4.1 Overview

In the Peikert-Waters [52] design, the matrix entries are ciphertexts of an underlying homomorphic encryption scheme, and the function output is a vector of ciphertexts of the same scheme. We begin by presenting an IBE scheme, that we call the basic IBE scheme, such that the function outputs of our eventual IB-TDF will be a vector of ciphertexts of this IBE scheme. Towards building the IB-TDF, the first difficulty we run into in setting up the matrix is that ciphertexts depend on the identity and we cannot have a different matrix for every identity. Thus, our approach is more intrusive. We will have many matrices which contain certain "atoms" from which, given an identity, one can reconstruct ciphertexts of the IBE scheme. The result of this intrusive approach is that security of the IB-TDF relies on more than security of the base IBE scheme. Our ciphertext pseudorandomness lemma (Lemma 4.1) shows something stronger, namely that even the atoms from which the ciphertexts are created look random under DLIN. This will be used to establish Lemma 4.2, which moves from the real to the lossy setup. The heart of the argument is the proofs of the lemmas, which are in the appendices.

We introduce a general framework that allows us to treat both the selective-id and adaptive-id cases in as unified a way as possible. We will first specify an E-IBTDF. The selective-id and adaptive-id IB-TDFs are obtained via different auxiliary inputs. Furthermore, the siblings used to prove lossiness also emanate from this E-IBTDF. With this approach, the main lemmas become usable in both the selective-id and adaptive-id cases with only minor adjustments for the latter due to artifical aborts. This saves us from repeating similar arguments and significantly compacts the proof.

4.2 Our basic IBE scheme

We associate to any integer $\mu \geq 1$ and any identity space $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$ an IBE scheme $\mathsf{IBE}[\mu, \mathsf{IDSp}]$ that has message space $\{0,1\}$ and algorithms as follows:

1. Parameters: Algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Pg}$ lets $g \xleftarrow{\$} \mathbb{G}^*$; $t \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{g} \leftarrow g^t$. It then lets $H, \hat{H} \xleftarrow{\$} \mathbb{G}$; $\mathbf{U}, \hat{\mathbf{U}} \xleftarrow{\$} \mathbb{G}^{\mu+1}$. It returns $pars = (g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}})$ as the public parameters and $msk = t$ as the master secret key.
2. Key generation: Given parameters $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}})$, master secret t and identity $id \in \mathsf{IDSp}$, algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Kg}$ returns decryption key (D_1, D_2, D_3, D_4) computed by letting $r, \hat{r} \xleftarrow{\$} \mathbb{Z}_p$ and setting
$$D_1 \leftarrow \mathcal{H}(\mathbf{U}, id)^{tr} \cdot H^{t\hat{r}} \ ; \ D_2 \leftarrow \mathcal{H}(\hat{\mathbf{U}}, id)^{r} \cdot \hat{H}^{\hat{r}} \ ; \ D_3 \leftarrow g^{-tr} \ ; \ D_4 \leftarrow g^{-t\hat{r}} \ .$$
3. Encryption: Given parameters $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}})$, identity $id \in \mathsf{IDSp}$ and message $M \in \{0,1\}$, algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Enc}$ returns ciphertext (C_1, C_2, C_3, C_4) computed as follows. If $M = 0$ then it lets $s, \hat{s} \xleftarrow{\$} \mathbb{Z}_p$ and $C_1 \leftarrow g^s$; $C_2 \leftarrow \hat{g}^{\hat{s}}$; $C_3 \leftarrow \mathcal{H}(\mathbf{U}, id)^s \cdot \mathcal{H}(\hat{\mathbf{U}}, id)^{\hat{s}}$; $C_4 \leftarrow H^s \hat{H}^{\hat{s}}$. If $M = 1$ it lets $C_1, C_2, C_3, C_4 \xleftarrow{\$} \mathbb{G}$.
4. Decryption: Given parameters $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}})$, identity $id \in \mathsf{IDSp}$, decryption key (D_1, D_2, D_4, D_4) for id and ciphertext (C_1, C_2, C_3, C_4), algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Dec}$ returns 0 if $\mathbf{e}(C_1, D_1)\mathbf{e}(C_2, D_2)$ $\mathbf{e}(C_3, D_3)\mathbf{e}(C_4, D_4) = \mathbf{1}_T$ and 1 otherwise.

This scheme has non-zero decryption error (at most $2/p$) yet our IBTDF will have zero inversion error. This scheme turns out to be IND-CPA+ANON-CPA although we will not need this in what follows. Instead we will have to consider a distinguishing game related to this IBE scheme and our IBTDF. In Appendix A we give a (more natural) variant of $\mathsf{IBE}[\mu, \mathsf{IDSp}]$ that is more efficient and encrypts strings rather than bits. The improved IBE scheme can still be proved IND-CPA+ANON-CPA but it cannot be used for our purpose of building IB-TDFs.

4.3 Our E-IBTDF and IB-TDF

Our E-IBTDF $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}]$ is associated to any integers $n, \mu \geq 1$ and any identity space $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$. It has message space $\{0,1\}^n$ and auxiliary input space $\mathbb{Z}_p^{\mu+1}$, and the algorithms are as follows:

1. Parameters: Given auxiliary input $\mathbf{y}$, algorithm $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Pg}$ lets $g \xleftarrow{\$} \mathbb{G}^*$; $t \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{g} \leftarrow g^t$; $U \xleftarrow{\$} \mathbb{G}^*$. It then lets $\mathbf{H}, \hat{\mathbf{H}} \xleftarrow{\$} \mathbb{G}^n$; $\mathbf{V}, \hat{\mathbf{V}} \xleftarrow{\$} \mathbb{G}^{n \times (\mu+1)}$ and $\mathbf{s} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$; $\hat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^n$. It returns $pars = (g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$ as the public parameters and $msk = t$ as the master secret key where for $1 \leq i, j \leq n$ and $0 \leq k \leq \mu$:
$$\mathbf{G}[i] \leftarrow g^{\mathbf{s}[i]} \ ; \ \hat{\mathbf{G}}[i] \leftarrow \hat{g}^{\hat{\mathbf{s}}[i]} \ ; \ \mathbf{J}[i,j] \leftarrow \mathbf{H}[j]^{\mathbf{s}[i]} \hat{\mathbf{H}}[j]^{\hat{\mathbf{s}}[i]} \ ; \ \mathbf{W}[i,j,k] \leftarrow \mathbf{V}[j,k]^{\mathbf{s}[i]} \hat{\mathbf{V}}[j,k]^{\hat{\mathbf{s}}[i]} U^{\mathbf{s}[i]\mathbf{y}[k]\Delta(i,j)} \ ,$$
where we recall that $\Delta(i,j) = 1$ if $i = j$ and 0 otherwise is the Kronecker Delta function.
2. Key generation: Given parameters $(g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$, master secret t and identity $id \in \mathsf{IDSp}$, algorithm $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Kg}$ returns decryption key $(\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4)$ where $\mathbf{r} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$; $\hat{\mathbf{r}} \xleftarrow{\$} \mathbb{Z}_p^n$ and for $1 \leq i \leq n$
$$\mathbf{D}_1[i] \leftarrow \mathcal{H}(\mathbf{V}[i,\cdot], id)^{t\mathbf{r}[i]} \cdot \mathbf{H}[i]^{t\hat{\mathbf{r}}[i]} \ ; \ \mathbf{D}_2[i] \leftarrow \mathcal{H}(\hat{\mathbf{V}}[i,\cdot], id)^{\mathbf{r}[i]} \cdot \hat{H}[i]^{\hat{\mathbf{r}}[i]} \ ; \ \mathbf{D}_3[i] \leftarrow g^{-t\mathbf{r}[i]} \ ; \ \mathbf{D}_4[i] \leftarrow g^{-t\hat{\mathbf{r}}[i]} \ .$$
3. Evaluate: Given parameters $(g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$, identity $id \in \mathsf{IDSp}$ and input $x \in \{0,1\}^n$, algorithm $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Ev}$ returns $(C_1, C_2, \mathbf{C}_3, \mathbf{C}_4)$ where for $1 \leq j \leq n$
$$C_1 \leftarrow \textstyle\prod_{i=1}^n \mathbf{G}[i]^{x[i]} \ ; \ C_2 \leftarrow \prod_{i=1}^n \hat{\mathbf{G}}[i]^{x[i]} \ ; \ \mathbf{C}_3[j] \leftarrow \prod_{i=1}^n \prod_{k=0}^{\mu} \mathbf{W}[i,j,k]^{x[i]\overline{id}[k]} \ ; \ \mathbf{C}_4[j] \leftarrow \prod_{i=1}^n \mathbf{J}[i,j]^{x[i]}$$
4. Invert: Given parameters $(g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$, identity $id \in \mathsf{IDSp}$, decryption key $(\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4)$ for id and output (ciphertext) $(C_1, C_2, \mathbf{C}_3, \mathbf{C}_4)$, algorithm $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Ev}^{-1}$ returns $x \in$

$\{0,1\}^n$ where for $1 \le j \le n$ it sets $x[j] = 0$ if $\mathbf{e}(C_1, \mathbf{D}_1[j])\mathbf{e}(C_2, \mathbf{D}_2[j])\mathbf{e}(\mathbf{C}_3[j], \mathbf{D}_3[j])\mathbf{e}(\mathbf{C}_4[j], \mathbf{D}_4[j]) = \mathbf{1}_T$ and 1 otherwise.

INVERTIBILITY. We observe that if parameters $(g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$ were generated with auxiliary input $\mathbf{y}$ and $(C_1, C_2, \mathbf{C}_3, \mathbf{C}_4) = \overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Ev}((g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}), id, x)$ then for $1 \le j \le n$

$$
\begin{aligned}
C_1 &= \textstyle\prod_{i=1}^{n} g^{\mathbf{s}[i]x[i]} = g^{\langle \mathbf{s}, x\rangle} && (3)\\
C_2 &= \textstyle\prod_{i=1}^{n} \hat{g}^{\hat{\mathbf{s}}[i]x[i]} = \hat{g}^{\langle \hat{\mathbf{s}}, x\rangle} && (4)\\
\mathbf{C}_3[j] &= \textstyle\prod_{i=1}^{n}\prod_{k=0}^{\mu} \mathbf{V}[j,k]^{\mathbf{s}[i]x[i]\overline{id}[k]}\hat{\mathbf{V}}[j,k]^{\hat{\mathbf{s}}[i]x[i]\overline{id}[k]} U^{\mathbf{s}[i]x[i]\mathbf{y}[k]\overline{id}[k]\Delta(i,j)}\\
&= \textstyle\prod_{i=1}^{n} \mathcal{H}(\mathbf{V}[j,\cdot], id)^{\mathbf{s}[i]x[i]}\mathcal{H}(\hat{\mathbf{V}}[j,\cdot], id)^{\hat{\mathbf{s}}[i]x[i]} U^{\mathbf{s}[i]x[i]f(\mathbf{y},id)\Delta(i,j)}\\
&= \mathcal{H}(\mathbf{V}[j,\cdot], id)^{\langle \mathbf{s}, x\rangle}\mathcal{H}(\hat{\mathbf{V}}[j,\cdot], id)^{\langle \hat{\mathbf{s}}, x\rangle} U^{\mathbf{s}[j]x[j]f(\mathbf{y},id)} && (5)\\
\mathbf{C}_4[j] &= \textstyle\prod_{i=1}^{n} \mathbf{H}[j]^{\mathbf{s}[i]x[i]}\hat{\mathbf{H}}[j]^{\hat{\mathbf{s}}[i]x[i]} = \mathbf{H}[j]^{\langle \mathbf{s}, x\rangle}\hat{\mathbf{H}}[j]^{\langle \hat{\mathbf{s}}, x\rangle}\ . && (6)
\end{aligned}
$$

Thus if $x[j] = 0$ then $(C_1, C_2, \mathbf{C}_3[j], \mathbf{C}_4[j])$ is an encryption, under our base IBE scheme, of the message 0, with coins $\langle \mathbf{s}, x\rangle \bmod p, \langle \hat{\mathbf{s}}, x\rangle \bmod p$, parameters $(g, \hat{g}, \mathbf{H}[j], \hat{\mathbf{H}}[j], \mathbf{V}[j,\cdot], \hat{\mathbf{V}}[j,\cdot])$ and identity id. The inversion algorithm will thus correctly recover $x[j] = 0$. On the other hand suppose $x[j] = 1$. Then $\mathbf{e}(C_1, \mathbf{D}_1[j])\mathbf{e}(C_2, \mathbf{D}_2[j])\mathbf{e}(\mathbf{C}_3[j], \mathbf{D}_3[j])\mathbf{e}(\mathbf{C}_4[j], \mathbf{D}_4[j]) = \mathbf{e}(U^{\mathbf{s}[j]x[j]f(\mathbf{y},id)}, \mathbf{D}_3[j])$. Now suppose $f(\mathbf{y}, id) \bmod p \neq 0$. Then $U^{\mathbf{s}[j]x[j]f(\mathbf{y},id)} \neq \mathbf{1}$ because we chose $\mathbf{s}[j]$ to be non-zero modulo p and $\mathbf{D}_3[j] \neq \mathbf{1}$ because we chose $\mathbf{r}[j]$ to be non-zero modulo p. So the result of the pairing is never $\mathbf{1}_T$, meaning the inversion algorithm will again correctly recover $x[j] = 1$. We have established that auxiliary input $\mathbf{y}$ grants invertibility, meaning induced IBTDF $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}](\mathbf{y})$ satisfies the correct inversion condition, if $f(\mathbf{y}, id) \bmod p \neq 0$ for all $id \in \mathsf{IDSp}$.

OUR IBTDF. We associate to any integers $n, \mu \ge 1$ and any identity space $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$ the IBTDF scheme induced by our E-IBTDF $\overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}]$ via auxiliary input $\mathbf{y} = (1, 0, \ldots, 0) \in \mathbb{Z}_p^{\mu+1}$, and denote this IBTDF scheme by $\overline{\mathsf{F}}[n, \mu, \mathsf{IDSp}]$. This IBTDF satisfies the correct inversion requirement because $f(\mathbf{y}, id) = \overline{id}[0] = 1 \not\equiv 0 \pmod p$ for all id. We will show that this IBTDF is selective-id secure when $\mu = 1$ and $\mathsf{IDSp} = \mathbb{Z}_p$, and adaptive-id secure when $\mathsf{IDSp} = \{0,1\}^\mu$. In the first case, it is fully lossy (i.e. 1-lossy) and in the second it is δ-lossy for appropriate δ. First we prove two technical lemmas that we will use in both cases.

4.4 Ciphertext pseudorandomness lemma

Consider games ReC, RaC of Figure 3 associated to some choice of $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$. The adversary provides the **Initialize** procedure with an auxiliary input $\mathbf{y} \in \mathbb{Z}_p^{\mu+1}$. Parameters are generated as per our base IBE scheme with the addition of U. The decryption key for id is computed as per our base IBE scheme except that the games refuse to provide it when $f(\mathbf{y}, id) = 0$. The challenge oracle, however, does not return ciphertexts of our IBE scheme. In game ReC, it returns group elements that resemble diagonal entries of the matrices in the parameters of our E-IBTDF, and in game RaC it returns random group elements. Notice that the challenge oracle does not take an identity as input. (Indeed, it has no input.) As usual it must be invoked exactly once. The following lemma says the games are indistinguishable under DLIN. The proof is in Section 4.7.

Lemma 4.1 *Let $\mu \ge 1$ be an integer and $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$. Let P be an adversary. Then there is an adversary B such that*

$$\Pr\left[\,\mathrm{ReC}^P\,\right] - \Pr\left[\,\mathrm{RaC}^P\,\right] \;\le\; (\mu + 2) \cdot \mathbf{Adv}^{\mathrm{dlin}}(B)\ . \qquad (7)$$

The running time of B is that of P plus some overhead.

proc Initialize($\mathbf{y}$) // ReC, RaC
$(pars, msk) \xleftarrow{\$} \mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Pg}$
$(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}) \leftarrow pars$
$U \xleftarrow{\$} \mathbb{G}^*$
Return $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}, U)$

proc GetDK(id) // ReC, RaC
If $f(\mathbf{y}, id) = 0$ then $dk \leftarrow \perp$
Else $dk \leftarrow \mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Kg}(pars, msk, id)$
Return dk

proc Ch() // ReC
$s \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{s} \xleftarrow{\$} \mathbb{Z}_p$; $G \leftarrow g^s$; $\hat{G} \leftarrow \hat{g}^{\hat{s}}$; $S \leftarrow H^s \hat{H}^{\hat{s}}$
For $k = 0, \ldots, \mu$ do $\mathbf{Z}[k] \leftarrow (U^{\mathbf{y}[k]} \mathbf{U}[k])^s \hat{\mathbf{U}}[k]^{\hat{s}}$
Return $(G, \hat{G}, S, \mathbf{Z})$

proc Ch() // RaC
$G, \hat{G}, S \xleftarrow{\$} \mathbb{G}$; $\mathbf{Z} \xleftarrow{\$} \mathbb{G}^{\mu+1}$
Return $(G, \hat{G}, S, \mathbf{Z})$

proc Finalize(d') // ReC, RaC
Return $(d' = 1)$

Figure 3: Games ReC ("Real Ciphertexts") and RaC ("Random Ciphertexts") associated to $\mathsf{IDSp} \subseteq \mathbb{Z}_p^{\mu}$.

proc Initialize(id)
$\mathbf{y}_0 \xleftarrow{\$} \mathsf{Aux}(id)$; $\mathbf{y}_1 \leftarrow (1, 0, \ldots, 0)$; $\textsc{Win} \leftarrow \mathsf{true}$
$g \xleftarrow{\$} \mathbb{G}^*$; $t \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{g} \leftarrow g^t$; $U \xleftarrow{\$} \mathbb{G}^*$
$\mathbf{H}, \hat{\mathbf{H}} \xleftarrow{\$} \mathbb{G}^n$; $\mathbf{V}, \hat{\mathbf{V}} \xleftarrow{\$} \mathbb{G}^{n \times (\mu+1)}$; $\mathbf{s} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$; $\hat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^n$
For $i = 1, \ldots, n$ do
  $\mathbf{G}[i] \leftarrow g^{\mathbf{s}[i]}$; $\hat{\mathbf{G}}[i] \leftarrow \hat{g}^{\hat{\mathbf{s}}[i]}$
  For $j = 1, \ldots, n$ do
    $\mathbf{J}[i, j] \leftarrow \mathbf{H}[j]^{\mathbf{s}[i]} \hat{\mathbf{H}}[j]^{\hat{\mathbf{s}}[i]}$
    For $k = 0, \ldots, \mu$ do
      If ($i = j$ and $i \le l$) then $\mathbf{W}[i, j, k] \xleftarrow{\$} \mathbb{G}$
      Else $\mathbf{W}[i, j, k] \leftarrow \mathbf{V}[j, k]^{\mathbf{s}[i]} \hat{\mathbf{V}}[j, k]^{\hat{\mathbf{s}}[i]} U^{\mathbf{s}[i] \mathbf{y}_b[k] \Delta(i,j)}$
$pars \leftarrow (g, \hat{g}, \mathbf{G}, \hat{\mathbf{G}}, \mathbf{J}, \mathbf{W}, \mathbf{H}, \hat{\mathbf{H}}, \mathbf{V}, \hat{\mathbf{V}}, U)$; $msk \leftarrow t$
$IS \leftarrow \emptyset$; $id^* \leftarrow id$
Return $pars$

proc GetDK(id)
$IS \leftarrow IS \cup \{id\}$
If $f(\mathbf{y}_0, id) = 0$ then $\textsc{Win} \leftarrow \mathsf{false}$; $dk \leftarrow \perp$
Else $dk \leftarrow \overline{\mathsf{E}}[n, \mu, \mathsf{IDSp}].\mathsf{Kg}(pars, msk, id)$
Return dk

proc Ch(id)
$id^* \leftarrow id$
If $f(\mathbf{y}_0, id) \neq 0$ then $\textsc{Win} \leftarrow \mathsf{false}$

proc Finalize(d')
Return $((d' = 1)$ and $(id^* \notin IS)$ and $\textsc{Win})$

Figure 4: Games $\mathrm{RL}_{l,b}$ ($0 \le l \le n$ and $b \in \{0, 1\}$) associated to $n, \mu, \mathsf{IDSp}, \mathsf{Aux}$ for proof of Lemma 4.2.

4.5 Proof of Lemma 4.2

Consider the games of Figure 4. Game $\mathrm{RL}_{l,b}$ makes the diagonal entries of $\mathbf{W}$ (namely all the $\mu + 1$ entries with $i = j$) random for $i \le l$ and otherwise makes them using $\mathbf{y}_b$. Game $\mathrm{RL}_{0,1}$ is the same as game RL_0 and game $\mathrm{RL}_{0,0}$ is the same as game RL_n. Games $\mathrm{RL}_{n,0}, \mathrm{RL}_{n,1}$ are identical: both make all diagonal entries of $\mathbf{W}$ (meaning, $i = j$) random, and when $i \neq j$ we have $\Delta(i, j) = 0$ so $\mathbf{y}_b(k)$ has no impact on $\mathbf{W}[i, j, k]$ in the Else statement. Thus we have

$$\Pr[\mathrm{RL}_0^A] - \Pr[\mathrm{RL}_n^A] \;=\; \big(\Pr[\mathrm{RL}_{0,1}^A] - \Pr[\mathrm{RL}_{n,1}^A]\big) + \big(\Pr[\mathrm{RL}_{n,0}^A] - \Pr[\mathrm{RL}_{0,0}^A]\big) \ .$$

We will design adversaries P_0, P_1 so that

$$\Pr[\mathrm{ReC}^{P_0}] - \Pr[\mathrm{RaC}^{P_0}] \;=\; \frac{1}{n} \cdot \big(\Pr[\mathrm{RL}_{n,0}^A] - \Pr[\mathrm{RL}_{0,0}^A]\big) \tag{8}$$

$$\Pr[\mathrm{ReC}^{P_1}] - \Pr[\mathrm{RaC}^{P_1}] \;=\; \frac{1}{n} \cdot \big(\Pr[\mathrm{RL}_{0,1}^A] - \Pr[\mathrm{RL}_{n,1}^A]\big) \ . \tag{9}$$

Adversary P picks $b \xleftarrow{\$} \{0, 1\}$ and runs P_b. This yields Equation (10). Now we present adversary P_b ($b \in \{0, 1\}$). It runs adversary A, responding to its oracle queries as follows.

When A makes query $\mathbf{Initialize}(id)$, adversary P_b begins with

$l \xleftarrow{\$} \{1,\ldots,n\}$; $\mathbf{y}_0 \xleftarrow{\$} \mathsf{Aux}(id)$; $\mathbf{y}_1 \leftarrow (1,0,\ldots,0)$; $\textsc{Win}_A \leftarrow \mathsf{true}$; $IS_A \leftarrow \emptyset$
$(g,\hat{g},H,\hat{H},\mathbf{U},\hat{\mathbf{U}},U) \xleftarrow{\$} \mathbf{Initialize}(\mathbf{y}_b)$; $(G,\hat{G},S,\mathbf{Z}) \xleftarrow{\$} \mathbf{Ch}()$.

Here P_b has called its own **Initialize** procedure with input $\mathbf{y}_b$ and then called its **Ch** procedure. Now it creates parameters *pars* for A as follows:

$\mathbf{h},\hat{\mathbf{h}} \xleftarrow{\$} \mathbb{Z}_p^n$; $\mathbf{v},\hat{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_p^{n\times(\mu+1)}$; $\mathbf{s} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$; $\hat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^n$
For $i = 1,\ldots,n$ do
  If $(i = l)$ then $\mathbf{H}[i] \leftarrow H$; $\hat{\mathbf{H}}[i] \leftarrow \hat{H}$; $\mathbf{G}[i] \leftarrow G$; $\hat{\mathbf{G}}[i] \leftarrow \hat{G}$
  If $(i \neq l)$ then $\mathbf{H}[i] \leftarrow g^{\mathbf{h}[i]}$; $\hat{\mathbf{H}}[i] \leftarrow \hat{g}^{\hat{\mathbf{h}}[i]}$; $\mathbf{G}[i] \leftarrow g^{\mathbf{s}[i]}$; $\hat{\mathbf{G}}[i] \leftarrow \hat{g}^{\hat{\mathbf{s}}[i]}$
  For $k = 0,\ldots,\mu$ do
    If $(i = l)$ then $\mathbf{V}[i,k] \leftarrow \mathbf{U}[k]$; $\hat{\mathbf{V}}[i,k] \leftarrow \hat{\mathbf{U}}[k]$
    If $(i \neq l)$ then $\mathbf{V}[i,k] \leftarrow g^{\mathbf{v}[i,k]}$; $\hat{\mathbf{V}}[i,k] \leftarrow \hat{g}^{\hat{\mathbf{v}}[i,k]}$
For $i = 1,\ldots,n$ do
  For $j = 1,\ldots,n$ do
    If $(i = l$ and $j = i)$ then $\mathbf{J}[i,j] \leftarrow S$
    If $(i = l$ and $j \neq i)$ then $\mathbf{J}[i,j] \leftarrow G^{\mathbf{h}[j]}\hat{G}^{\hat{\mathbf{h}}[j]}$
    If $(i \neq l)$ then $\mathbf{J}[i,j] \leftarrow \mathbf{H}[j]^{\mathbf{s}[i]}\hat{\mathbf{H}}[j]^{\hat{\mathbf{s}}[i]}$
    For $k = 0,\ldots,\mu$ do
      If $(i = j$ and $i \leq l-1)$ then $\mathbf{W}[i,j,k] \xleftarrow{\$} \mathbb{G}$
      If $(i = j$ and $i = l)$ then $\mathbf{W}[i,j,k] \leftarrow \mathbf{Z}[k]$
      Else $\mathbf{W}[i,j,k] \leftarrow \mathbf{V}[j,k]^{\mathbf{s}[i]}\hat{\mathbf{V}}[j,k]^{\hat{\mathbf{s}}[i]}U^{\mathbf{s}[i]\mathbf{y}_b[k]\Delta(i,j)}$
pars $\leftarrow (g,\hat{g},\mathbf{G},\hat{\mathbf{G}},\mathbf{J},\mathbf{W},\mathbf{H},\hat{\mathbf{H}},\mathbf{V},\hat{\mathbf{V}},U)$

It returns *pars* to A.

When adversary A makes query $\mathbf{GetDK}(id)$, adversary P_b proceeds as follows. In this code, **GetDK** is P_b's own oracle:

$IS_A \leftarrow IS_A \cup \{id\}$
If $f(\mathbf{y}_0,id) = 0$ then $\textsc{Win}_A \leftarrow \mathsf{false}$; $dk \leftarrow \perp$
Else
  $(D_1,D_2,D_3,D_4) \xleftarrow{\$} \mathbf{GetDK}(id)$
  $\mathbf{r}' \xleftarrow{\$} (\mathbb{Z}_p^*)^n$; $\hat{\mathbf{r}}' \xleftarrow{\$} \mathbb{Z}_p^n$
  For $i = 1,\ldots,n$ do
    If $i = l$ then $(\mathbf{D}_1[i],\mathbf{D}_2[i],\mathbf{D}_3[i],\mathbf{D}_4[i]) \leftarrow (D_1,D_2,D_3,D_4)$
    Else
      $\mathbf{D}_1[i] \leftarrow \mathcal{H}(\mathbf{V}[i,\cdot],id)^{\mathbf{r}'[i]}\mathbf{H}[i]^{\hat{\mathbf{r}}'[i]}$; $\mathbf{D}_2[i] \leftarrow g^{f(\hat{\mathbf{v}},id)\mathbf{r}'[i]}g^{\hat{\mathbf{h}}[i]\hat{\mathbf{r}}[i]}$
      $\mathbf{D}_3[i] \leftarrow g^{-\mathbf{r}'[i]}$; $\mathbf{D}_4[i] \leftarrow g^{-\hat{\mathbf{r}}'[i]}$
  $dk \leftarrow (\mathbf{D}_1,\mathbf{D}_2,\mathbf{D}_3,\mathbf{D}_4)$

It returns dk to A. Notice that P_b's invocation of **GetDK** will never return $\perp$. In the case $b = 1$ this is true because $f(\mathbf{y}_1,\cdot) = 1 \neq 0$. In the case $b = 0$ it is true because the case $f(\mathbf{y}_0,id) = 0$ was excluded by the If statement. To justify the above simulation, define $\mathbf{r},\hat{\mathbf{r}}$ by $\mathbf{r}[i] = \mathbf{r}'[i]/t$ and $\hat{\mathbf{r}}[i] = \hat{\mathbf{r}}'[i]/t$ for $i \neq l$ and $\mathbf{r}[l],\hat{\mathbf{r}}[l]$ as the randomness underlying (D_1,D_2,D_3,D_4). Then think of $\mathbf{r},\hat{\mathbf{r}}$ as the randomness used by the real key generation algorithm. Here t is the secret key, so that $\hat{g} = g^t$.

When adversary A makes query $\mathbf{Ch}(id)$, adversary P_b proceeds as follows:

$id^* \leftarrow id$
If $f(\mathbf{y}_0,id) \neq 0$ then $\textsc{Win}_A \leftarrow \mathsf{false}$.

$\underline{\textbf{proc Initialize}(id)}$ // RL_0
$\mathbf{y}_0 \xleftarrow{\$} \mathsf{Aux}(id)$; $\mathbf{y}_1 \leftarrow (1,0,\ldots,0)$
$(pars, msk) \xleftarrow{\$} \overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}].\mathsf{Pg}(\mathbf{y}_1)$
$IS \leftarrow \emptyset$; $id^* \leftarrow id$; $\textsc{Win} \leftarrow \mathsf{true}$
Return $pars$

$\underline{\textbf{proc Initialize}(id)}$ // RL_n
$\mathbf{y}_0 \xleftarrow{\$} \mathsf{Aux}(id)$; $\mathbf{y}_1 \leftarrow (1,0,\ldots,0)$
$(pars, msk) \xleftarrow{\$} \overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}].\mathsf{Pg}(\mathbf{y}_0)$
$IS \leftarrow \emptyset$; $id^* \leftarrow id$; $\textsc{Win} \leftarrow \mathsf{true}$
Return $pars$

$\underline{\textbf{proc GetDK}(id)}$ // $\mathrm{RL}_0, \mathrm{RL}_n$
$IS \leftarrow IS \cup \{id\}$
If $f(\mathbf{y}_0, id) = 0$ then $\textsc{Win} \leftarrow \mathsf{false}$; $dk \leftarrow \perp$
Else $dk \leftarrow \overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}].\mathsf{Kg}(pars, msk, id)$
Return dk

$\underline{\textbf{proc Ch}(id)}$ // $\mathrm{RL}_0, \mathrm{RL}_n$
$id^* \leftarrow id$
If $f(\mathbf{y}_0, id) \neq 0$ then $\textsc{Win} \leftarrow \mathsf{false}$

$\underline{\textbf{proc Finalize}(d')}$ // $\mathrm{RL}_0, \mathrm{RL}_n$
Return $((d' = 1)$ and $(id^* \notin IS)$ and $\textsc{Win})$

Figure 5: Games $\mathrm{RL}_0, \mathrm{RL}_n$ ("Real-to-Losssy") associated to $n, \mu, \mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$ and auxiliary input generator algorithm Aux.

Finally, A halts with output d'. Adversaries P_0, P_1 compute their output differently. Adversary P_1 returns 1 if

$$(d' = 1) \text{ and } id^* \notin IS_A \text{ and } \textsc{Win}_A$$

and 0 otherwise. Adversary P_0 does the opposite, returning 0 if the above condition is true and 1 otherwise. We obtain Equations (8), (9) as follows:

$$\begin{aligned}
\Pr[\mathrm{ReC}^{P_1}] - \Pr[\mathrm{RaC}^{P_1}] &= \frac{1}{n}\sum_{l=1}^{n} \Pr[\mathrm{RL}^A_{l-1,1}] - \Pr[\mathrm{RL}^A_{l,1}] \\
&= \Pr[\mathrm{RL}^A_{0,1}] - \Pr[\mathrm{RL}^A_{n,1}] \\
\Pr[\mathrm{ReC}^{P_0}] - \Pr[\mathrm{RaC}^{P_0}] &= \frac{1}{n}\sum_{l=1}^{n} (1 - \Pr[\mathrm{RL}^A_{l-1,0}]) - (1 - \Pr[\mathrm{RL}^A_{l,0}]) \\
&= \frac{1}{n}\sum_{l=1}^{n} \Pr[\mathrm{RL}^A_{l,0}] - \Pr[\mathrm{RL}^A_{l-1,0}] \\
&= \Pr[\mathrm{RL}^A_{n,0}] - \Pr[\mathrm{RL}^A_{0,0}] \ .
\end{aligned}$$

4.6 Real-to-lossy lemma

Consider games $\mathrm{RL}_0, \mathrm{RL}_n$ of Figure 5 associated to some choice of $n, \mu, \mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$ and auxiliary input generator Aux for $\overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}]$. The latter is an algorithm that takes input an identity in IDSp and returns an auxiliary input in $\mathbb{Z}_p^{\mu+1}$. Game RL_0 obtains an auxiliary input $\mathbf{y}_0$ via Aux but generates parameters exactly as $\overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}].\mathsf{Pg}$ with the real auxiliary input $\mathbf{y}_1$. The game will return true under the same condition as game Real but additionally requiring that $f(\mathbf{y}_0, id) \neq 0$ for all $\mathbf{GetDK}(id)$ queries and $f(\mathbf{y}_0, id) = 0$ for the $\mathbf{Ch}(id)$ query. Game RL_n generates parameters with the auxiliary input provided by Aux but is otherwise identical to game RL_0. The following lemma says it is hard to distinguish these games. We will apply this by defining Aux in such a way that its output $\mathbf{y}_0$ results in a lossy setup. The proof of the following is in Section 4.5.

Lemma 4.2 *Let $n, \mu \geq 1$ be integers and $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$. Let Aux be an auxiliary input generator for $\overline{\mathsf{E}}[n,\mu,\mathsf{IDSp}]$ and A an adversary. Then there is an adversary P such that*

$$\Pr[\mathrm{RL}^A_0] - \Pr[\mathrm{RL}^A_n] \ \leq \ 2n \cdot \left(\Pr\left[\mathrm{ReC}^P\right] - \Pr\left[\mathrm{RaC}^P\right]\right) \ . \tag{10}$$

$\underline{\textbf{proc Initialize}(\mathbf{y})}$ // $\mathrm{PC}, \mathrm{PC}_l$
$(pars, msk) \xleftarrow{\$} \mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Pg}$
$(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}) \leftarrow pars$
$U \xleftarrow{\$} \mathbb{G}^*$
Return $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}, U)$

$\underline{\textbf{proc GetDK}(id)}$ // $\mathrm{PC}, \mathrm{PC}_l$
If $f(\mathbf{y}, id) = 0$ then $dk \leftarrow \perp$
Else $dk \leftarrow \mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Kg}(pars, msk, id)$
Return dk

$\underline{\textbf{proc Ch}()}$ // PC
$s \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{s} \xleftarrow{\$} \mathbb{Z}_p$; $G \leftarrow g^s$; $\hat{G} \leftarrow \hat{g}^{\hat{s}}$; $S \leftarrow H^s \hat{H}^{\hat{s}}$
For $k = 0, \ldots, \mu$ do $\mathbf{Z}[k] \leftarrow (U^{\mathbf{y}[k]} \mathbf{U}[k])^s \hat{\mathbf{U}}[k]^{\hat{s}}$
Return $(G, \hat{G}, S, \mathbf{Z})$

$\underline{\textbf{proc Ch}()}$ // PC_l
$s \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{s} \xleftarrow{\$} \mathbb{Z}_p$; $G \leftarrow g^s$; $\hat{G} \leftarrow \hat{g}^{\hat{s}}$; $S \xleftarrow{\$} \mathbb{G}$
For $k = 0, \ldots, l-1$ do $\mathbf{Z}[k] \xleftarrow{\$} \mathbb{G}$
For $k = l, \ldots, \mu$ do $\mathbf{Z}[k] \leftarrow (U^{\mathbf{y}[k]} \mathbf{U}[k])^s \hat{\mathbf{U}}[k]^{\hat{s}}$
Return $(G, \hat{G}, S, \mathbf{Z})$

$\underline{\textbf{proc Finalize}(d')}$ // $\mathrm{PC}, \mathrm{PC}_l$
Return $(d' = 1)$

Figure 6: Games $\mathrm{PC}, \mathrm{PC}_l$ $(0 \le l \le \mu + 1)$ associated to $\mathsf{IDSp} \subseteq \mathbb{Z}_p^{\mu+1}$ for the proof of Lemma 4.1.

The running time of P is that of A plus some overhead. If A is selective-id then so is P.

The last statement allows us to use the lemma in both the selective-id and adaptive-id cases.

4.7 Proof of Lemma 4.1

Consider the games of Figure 6. Game PC is the same as game ReC. Game PC_l $(0 \le l \le \mu + 1)$ makes S random and also makes the first $l - 1$ entries of $\mathbf{Z}$ random and the rest real. Thus $\mathrm{PC}_{\mu+1}$ is the same as RaC. We will design adversaries B_1, B_2 so that

$$\mathbf{Adv}^{\mathrm{dlin}}(B_1) = \Pr[\mathrm{PC}^P] - \Pr[\mathrm{PC}_0^P] \quad (11)$$

$$\mathbf{Adv}^{\mathrm{dlin}}(B_2) = \frac{1}{\mu + 1} \left(\Pr[\mathrm{PC}_0^P] - \Pr[\mathrm{PC}_{\mu+1}^P] \right) \quad (12)$$

Adversary B will run B_1 with probability $1/(\mu + 2)$ and B_2 with probability $(\mu + 1)/(\mu + 2)$. This yields Equation (7).

On input $(g, \hat{g}, g^s, \hat{g}^{\hat{s}}, H, T)$ where T is either $H^{s+\hat{s}}$ or random, adversary B_1 runs adversary P, responding to its oracle queries as follows. When P makes query $\mathbf{Initialize}(\mathbf{y})$, adversary B_1 lets

$\mathbf{u}, \hat{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^{\mu+1}$; $u, v \xleftarrow{\$} \mathbb{Z}_p$; $\hat{H} \leftarrow H \hat{g}^v$; $U \leftarrow \hat{g}^u$
For $k = 0, \ldots, \mu$ do $\mathbf{U}[k] \leftarrow U^{-\mathbf{y}[k]} g^{\mathbf{u}[k]}$; $\hat{\mathbf{U}}[k] \leftarrow \hat{g}^{\hat{\mathbf{u}}[k]}$

It returns $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}, U)$ to P. When P makes its (single) $\mathbf{Ch}()$ query, adversary B_1 lets

$S \leftarrow T \hat{g}^{v\hat{s}}$
For $k = 0, \ldots, \mu$ do $\mathbf{Z}[k] \leftarrow g^{s\mathbf{u}[k]} \hat{g}^{\hat{s}\hat{\mathbf{u}}[k]}$

It returns $(g^s, \hat{g}^{\hat{s}}, S, \mathbf{Z})$ to P. Notice that for $0 \le k \le \mu$

$$\mathbf{Z}[k] = g^{s\mathbf{u}[k]} \hat{g}^{\hat{s}\hat{\mathbf{u}}[k]} = (U^{\mathbf{y}[k] - \mathbf{y}[k]} g^{\mathbf{u}[k]})^s \hat{g}^{\hat{s}\hat{\mathbf{u}}[k]} = (U^{\mathbf{y}[k]} \mathbf{U}[k])^s \hat{\mathbf{U}}[k]^{\hat{s}} .$$

Also if $T = H^{s+\hat{s}}$ then $S = T \hat{g}^{v\hat{s}} = H^s (H \hat{g}^v)^{\hat{s}} = H^s \hat{H}^{\hat{s}}$ as in PC while if T is random, so is S, as in PC_0. When P makes query $\mathbf{GetDK}(id)$, adversary B_1 does the following:

If $f(\mathbf{y}, id) = 0$ then $dk \leftarrow \perp$

Else

$r', \hat{r}' \xleftarrow{\$} \mathbb{Z}_p$

$D_1 \leftarrow g^{-f(\mathbf{y},id)ur'} g^{f(\mathbf{u},id)r'} H^{-f(\mathbf{u},id)\hat{r}'/f(\mathbf{y},id)}$; $D_2 \leftarrow g^{f(\hat{u},id)r'} H^{-f(\mathbf{u},id)\hat{r}'/f(\mathbf{y},id)} \hat{H}^{u\hat{r}'}$

$D_3 \leftarrow H^{\hat{r}'/f(\mathbf{y},id)} g^{-r'}$; $D_4 \leftarrow \hat{g}^{-u\hat{r}'}$; $dk \leftarrow (D_1, D_2, D_3, D_4)$

It returns dk to P. We now show this key is properly distributed. Let h be such that $H = g^h$ and let

$$r \;=\; \frac{r'}{t} - \frac{h\hat{r}'}{tf(\mathbf{y}, id)} \bmod p \quad \text{and} \quad \hat{r} \;=\; u\hat{r}' \bmod p \;.$$

Since $t, f(\mathbf{y}, uid)$ are non-zero modulo p and $r', \hat{r}'$ are random, $r, \hat{r}$ are random as well. The following computes the correct secret key components with the above randomness and shows that they are the ones of the simulation:

$$\begin{aligned}
\mathcal{H}(\mathbf{U}, id)^{tr} H^{t\hat{r}} &= \mathbf{U}[0]^{tr} \left(\textstyle\prod_{k=1}^{\mu} \mathbf{U}[k]^{id[k]tr}\right) H^{t\hat{r}} \\
&= U^{-\mathbf{y}[0]tr} g^{\mathbf{u}[0]tr} \left(\textstyle\prod_{k=1}^{\mu} U^{-\mathbf{y}[k]id[k]tr} g^{\mathbf{u}[k]id[k]tr}\right) H^{t\hat{r}} \\
&= U^{-f(\mathbf{y},id)tr} g^{f(\mathbf{u},id)tr} H^{t\hat{r}} \\
&= U^{-f(\mathbf{y},id)(r'-h\hat{r}'/f(\mathbf{y},id))} g^{f(\mathbf{u},id)(r'-h\hat{r}'/f(\mathbf{y},id))} H^{tu\hat{r}'} \\
&= \hat{g}^{-hu\hat{r}'} g^{-f(\mathbf{y},id)ur'} g^{f(\mathbf{u},id)r'} g^{-f(\mathbf{u},id)h\hat{r}'/f(\mathbf{y},id)} g^{htu\hat{r}'} \\
&= g^{-f(\mathbf{y},id)ur'} g^{f(\mathbf{u},id)r'} H^{-f(\mathbf{u},id)\hat{r}'/f(\mathbf{y},id)} \;=\; D_1 \\
\mathcal{H}(\hat{\mathbf{U}}, id)^{r} \hat{H}^{\hat{r}} &= \hat{\mathbf{U}}[0]^{r} \left(\textstyle\prod_{k=1}^{\mu} \hat{\mathbf{U}}[k]^{id[k]r}\right) \hat{H}^{\hat{r}} \;=\; \hat{g}^{\hat{u}[0]r} \left(\textstyle\prod_{k=1}^{\mu} \hat{g}^{\hat{u}[k]id[k]r}\right) \hat{H}^{\hat{r}} \\
&= \hat{g}^{f(\hat{u},id)r} \hat{H}^{\hat{r}} \;=\; g^{f(\hat{u},id)tr} \hat{H}^{\hat{r}} \\
&= g^{f(\hat{u},id)(r'-h\hat{r}'/f(\mathbf{y},id))} \hat{H}^{u\hat{r}'} \;=\; g^{f(\hat{u},id)r'} H^{-f(\mathbf{u},id)\hat{r}'/f(\mathbf{y},id)} \hat{H}^{u\hat{r}'} \;=\; D_2 \\
g^{-tr} &= g^{h\hat{r}'/f(\mathbf{y},id)-r'} \;=\; H^{\hat{r}'/f(\mathbf{y},id)} g^{-r'} \;=\; D_3 \\
g^{-t\hat{r}} &= g^{-tu\hat{r}'} \;=\; \hat{g}^{-u\hat{r}'} \;=\; D_4 \;.
\end{aligned}$$

Finally adversary P outputs d'. Adversary B_1 also outputs d', so we have Equation (11).

On input $(g, \hat{g}, g^s, \hat{g}^{\hat{s}}, \hat{U}, T)$ where T is either $\hat{U}^{s+\hat{s}}$ or random, adversary B_2 runs adversary P, responding to its oracle queries as follows. When P makes query $\mathbf{Initialize}(\mathbf{y})$, adversary B_1 lets

$l \xleftarrow{\$} \{0, \ldots, \mu\}$; $\mathbf{u}, \hat{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^{\mu+1}$; $u, h, \hat{h} \xleftarrow{\$} \mathbb{Z}_p$; $H \leftarrow \hat{g}^h$; $\hat{H} \leftarrow \hat{g}^{\hat{h}}$; $U \leftarrow g^u$

For $k = 0, \ldots, \mu$ do $\mathbf{U}[k] \leftarrow \hat{U}^{\Delta(l,k)} g^{\mathbf{u}[k]}$; $\hat{\mathbf{U}}[k] \leftarrow \hat{U}^{\Delta(l,k)} \hat{g}^{\hat{\mathbf{u}}[k]}$

It returns $(g, \hat{g}, H, \hat{H}, \mathbf{U}, \hat{\mathbf{U}}, U)$ to P. When P makes its (single) $\mathbf{Ch}()$ query, adversary B_2 lets

$S \xleftarrow{\$} \mathbb{G}$

For $k = 0, \ldots, l-1$ do $\mathbf{Z}[k] \xleftarrow{\$} \mathbb{G}$

For $k = l, \ldots, \mu$ do $\mathbf{Z}[k] \leftarrow (g^s)^{u\mathbf{y}[k]+\mathbf{u}[k]} (\hat{g}^{\hat{s}})^{\hat{\mathbf{u}}[k]} T^{\Delta(l,k)}$

It returns $(g^s, \hat{g}^{\hat{s}}, S, \mathbf{Z})$ to P. Notice that for $l+1 \le k \le \mu$

$$\mathbf{Z}[k] = (g^s)^{u\mathbf{y}[k]+\mathbf{u}[k]} (\hat{g}^{\hat{s}})^{\hat{\mathbf{u}}[k]} = U^{s\mathbf{y}[k]} \mathbf{U}[k]^s \hat{\mathbf{U}}[k]^{\hat{s}} = (U^{\mathbf{y}[k]} \mathbf{U}[k])^s \hat{\mathbf{U}}[k]^{\hat{s}} \;.$$

If $T = \hat{U}^{s+\hat{s}}$ then

$$\mathbf{Z}[l] = (g^s)^{u\mathbf{y}[l]+\mathbf{u}[l]} (\hat{g}^{\hat{s}})^{\hat{\mathbf{u}}[l]} T = U^{s\mathbf{y}[l]} (\hat{U}^{-1}\mathbf{U}[l])^s (\hat{U}^{-1}\hat{\mathbf{U}}[l])^{\hat{s}} \hat{U}^s \hat{U}^{\hat{s}} = (U^{\mathbf{y}[l]} \mathbf{U}[l])^s \hat{\mathbf{U}}[l]^{\hat{s}}$$

as in game PC_l. On the other hand if T is random then so is $\mathbf{Z}[l]$, as in game PC_{l+1}. When P makes query $\mathbf{GetDK}(id)$, adversary B_2 does the following:

If $f(\mathbf{y}, id) = 0$ then $dk \leftarrow \perp$
Else
$r, \hat{r}' \xleftarrow{\$} \mathbb{Z}_p$
$D_1 \leftarrow \hat{g}^{f(\mathbf{u},id)r}\hat{g}^{h\hat{r}'}$; $D_2 \leftarrow g^{f(\mathbf{u},id)r}\hat{U}^{id[l]r}g^{\hat{h}\hat{r}'}\hat{U}^{-\hat{h}id[l]r/h}$
$D_3 \leftarrow \hat{g}^{-r}$; $D_4 \leftarrow \hat{U}^{id[l]r/h}g^{-\hat{r}'}$; $dk \leftarrow (D_1, D_2, D_3, D_4)$

It returns dk to P. We now show this key is properly distributed. Let $\hat{u}$ be such that $\hat{U} = g^{\hat{u}}$ and let

$$\hat{r} \;=\; \frac{\hat{r}'}{t} - \frac{id[l]\hat{u}r}{th} \bmod p\ .$$

Since t is non-zero modulo p and $\hat{r}'$ is random, $\hat{r}$ is random as well. The following computes the correct secret key components with the above randomness and shows that they are the ones of the simulation:

$$\begin{aligned}
\mathcal{H}(\mathbf{U}, id)^{tr}H^{t\hat{r}} &= \mathbf{U}[0]^{tr}\left(\textstyle\prod_{k=1}^{\mu}\mathbf{U}[k]^{id[k]tr}\right)H^{t\hat{r}} \\
&= g^{\mathbf{u}[0]tr}\left(\textstyle\prod_{k=1}^{\mu}\hat{U}^{id[k]tr\Delta(l,k)}g^{\mathbf{u}[k]id[k]tr}\right)\hat{g}^{ht\hat{r}} \\
&= g^{f(\mathbf{u},id)tr}\hat{U}^{id[l]tr}\hat{g}^{ht\hat{r}} \;=\; g^{f(\mathbf{u},id)tr}\hat{U}^{id[l]tr}\hat{g}^{h(\hat{r}'-id[l]\hat{u}r/h)} \\
&= \hat{g}^{f(\mathbf{u},id)r}\hat{U}^{id[l]tr}\hat{g}^{h\hat{r}'}\hat{g}^{-id[l]\hat{u}r} \;=\; \hat{g}^{f(\mathbf{u},id)r}g^{id[l]\hat{u}rt}\hat{g}^{h\hat{r}'}\hat{g}^{-id[l]\hat{u}r} \\
&= \hat{g}^{f(\mathbf{u},id)r}\hat{g}^{h\hat{r}'} \;=\; D_1 \\
\mathcal{H}(\hat{\mathbf{U}}, id)^{r}\hat{H}^{\hat{r}} &= \hat{\mathbf{U}}[0]^{r}\left(\textstyle\prod_{k=1}^{\mu}\hat{\mathbf{U}}[k]^{id[k]r}\right)\hat{H}^{\hat{r}} \;=\; g^{\hat{\mathbf{u}}[0]r}\left(\textstyle\prod_{k=1}^{\mu}\hat{U}^{id[k]r\Delta(l,k)}g^{\hat{\mathbf{u}}[k]id[k]r}\right)\hat{g}^{\hat{h}\hat{r}} \\
&= g^{f(\hat{\mathbf{u}},id)r}\hat{U}^{id[l]r}g^{t\hat{h}\hat{r}} \;=\; g^{f(\hat{\mathbf{u}},id)r}\hat{U}^{id[l]r}g^{\hat{h}(\hat{r}'-id[l]\hat{u}r/h)} \\
&= g^{f(\hat{\mathbf{u}},id)r}\hat{U}^{id[l]r}g^{\hat{h}\hat{r}'}g^{-\hat{h}id[l]\hat{u}r/h} \;=\; g^{f(\mathbf{u},id)r}\hat{U}^{id[l]r}g^{\hat{h}\hat{r}'}\hat{U}^{-\hat{h}id[l]r/h} \;=\; D_2 \\
g^{-tr} &= \hat{g}^{-r} \;=\; D_3 \\
g^{-t\hat{r}} &= g^{\hat{u}rid[l]/h-\hat{r}'} \;=\; \hat{U}^{id[l]r/h}g^{-\hat{r}'} \;=\; D_4\ .
\end{aligned}$$

Finally adversary P outputs d'. Adversary B_2 also outputs d'. So

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dlin}}(B_2) &= \frac{1}{\mu+1}\sum_{l=0}^{\mu}\Pr[\mathrm{PC}_l^P] - \Pr[\mathrm{PC}_{l+1}^P] \\
&= \frac{1}{\mu+1}\Pr[\mathrm{PC}_0^P] - \Pr[\mathrm{PC}_{\mu+1}^P]
\end{aligned}$$

and we have Equation (12).

4.8 Selective-id security

We consider IBTDF $\overline{\mathsf{F}}[n, 1, \mathbb{Z}_p]$, the instance of our construction with $\mu = 1$ and $\mathsf{IDSp} = \mathbb{Z}_p$. We show that this IBTDF is selective-id δ-lossy for $\delta = 1$, meaning fully selective-id lossy, and hence selective-id one-way. To do this we define a sibling $\overline{\mathsf{LF}}[n, 1, \mathbb{Z}_p]$. It preserves the key-generation, evaluation and inversion algorithms of $\overline{\mathsf{F}}[n, 1, \mathbb{Z}_p]$ and alters parameter generation to

Algorithm $\overline{\mathsf{LF}}[n, 1, \mathbb{Z}_p].\mathsf{Pg}(id)$
$\mathbf{y} \leftarrow (-id, 1)$; $(pars, msk) \xleftarrow{\$} \overline{\mathsf{E}}[n, 1, \mathbb{Z}_p].\mathsf{Pg}(\mathbf{y})$; Return $(pars, msk)$

The following says that our IBTDF is 1-lossy under the DLIN assumption with lossiness $\ell = n - 2\lg(p)$.

Theorem 4.3 *Let $n > 2\lg(p)$ and let $\ell = n - 2\lg(p)$. Let $\mathsf{F} = \overline{\mathsf{F}}[n, 1, \mathbb{Z}_p]$ be the IBTDF associated by our construction to parameters n, $\mu = 1$ and $\mathsf{IDSp} = \mathbb{Z}_p$. Let $\mathsf{LF} = \overline{\mathsf{LF}}[n, 1, \mathbb{Z}_p]$ be the sibling associated to it as above. Let $\delta = 1$ and let be A a selective-id adversary. Then there is an adversary B such that*

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A) \;\le\; 2n(\mu+2)\cdot \mathbf{Adv}^{\text{dlin}}(B)\;. \tag{13}$$

The running time of B is that of A plus overhead.

Proof of Theorem 4.3: On input id, let algorithm Aux return $(-id, 1)$. Let $\mathrm{RL}_0, \mathrm{RL}_n$ be the games of Figure 5 with $\mu = 1$, $\mathsf{IDSp} = \mathbb{Z}_p$ and this Aux. Then we claim

$$\Pr\left[\,\mathrm{Real}^A_{\mathsf{F}}\,\right] = \Pr\left[\,\mathrm{RL}^A_0\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{Lossy}^A_{\mathsf{F},\mathsf{LF},\ell}\,\right] = \Pr\left[\,\mathrm{RL}^A_n\,\right]\;. \tag{14}$$

To justify this let id^* be the identity queried by A to both **Initialize** and **Ch**. (These queries are the same because A is selective-id.) Then $\mathbf{y}_0 = (-id^*, 1)$ so $f(\mathbf{y}_0, id) = id - id^*$. This is 0 iff $id = id^*$. This means that the conjunct $(id^* \notin IS) \wedge \textsc{Win}$ is always true. The claim of Equation (14) is now true because game RL_0 generates parameters with the real auxiliary input $\mathbf{y}_1 = (1, 0) \in \mathbb{Z}_p^2$ that, via $\overline{\mathsf{E}}[n, 1, \mathbb{Z}_p]$, defines F. However game RL_n generates parameters with auxiliary input $\mathbf{y}_0$. Since $f(\mathbf{y}_0, id^*) = 0$, the dependency of $\mathbf{C}_3[j]$ on $x[j]$ in Equation (5) vanishes when $id = id^*$. Examing equations (3), (4), (5), (6), we now see that with *pars* fixed, the values $\langle \mathbf{s}, x\rangle, \langle \hat{\mathbf{s}}, x\rangle$ determine the ciphertext $(C_1, C_2, \mathbf{C}_3, \mathbf{C}_4)$. Thus there are at most p^2 possible ciphertexts when $id = id^*$, and 2^n possible inputs. This means that $\lambda(\mathsf{F}.\mathsf{Ev}(pars, id^*, \cdot)) \ge n - \lg(p^2) = \ell$, which justifies the second claim of Equation (14). Recalling that $\delta = 1$, Equation (13) follows from Equation (1), Equation (14), Lemma 4.2 and Lemma 4.1. ▌

4.9 Adaptive-id Security

We consider IBTDF $\overline{\mathsf{F}}[n, \mu, \{0,1\}^\mu]$, the instance of our construction with $\mathsf{IDSp} = \{0,1\}^\mu \subset \mathbb{Z}_p^\mu$. We show that this IBTDF is adaptive-id δ-lossy for $\delta = (4(\mu+1)Q)^{-1}$ where Q is the number of key-derivation queries of the adversary. By Theorem 3.2 this means $\overline{\mathsf{F}}[n, \mu, \{0,1\}^\mu]$ is adaptive-id one-way. To do this we define a sibling $\overline{\mathsf{LF}}_Q[n, \mu, \{0,1\}^\mu]$. It preserves the key-generation, evaluation and inversion algorithms of $\overline{\mathsf{F}}[n, \mu, \{0,1\}^\mu]$ and alters parameter generation to $\overline{\mathsf{LF}}[n, \mu, \{0,1\}^\mu].\mathsf{Pg}(id)$ defined via

$$\mathbf{y} \leftarrow \mathsf{Aux}\,;\; (pars, msk) \xleftarrow{\$} \overline{\mathsf{E}}[n, \mu, \{0,1\}^\mu].\mathsf{Pg}(\mathbf{y})\,;\;\; \text{Return } (pars, msk)\;.$$

where algorithm Aux is defined via

$$\begin{array}{l}\mathbf{y}'[0] \xleftarrow{\$} \{0,\ldots,2Q-1\}\,;\; \ell \xleftarrow{\$} \{0,\ldots,\mu+1\}\,;\; \mathbf{y}[0] \leftarrow \mathbf{y}'[0] - 2\ell Q\\ \text{For } i = 1 \text{ to } \mu \text{ do } \mathbf{y}[i] \xleftarrow{\$} \{0,\ldots,2Q-1\}\\ \text{Return } \mathbf{y} \in \mathbb{Z}_p^{\mu+1}\end{array}$$

The following says that our IBTDF is δ-lossy under the DLIN assumption with lossiness $\ell = n - 2\lg(p)$.

Theorem 4.4 *Let $n > 2\lg(p)$ and let $\ell = n - 2\lg(p)$. Let $\mathsf{F} = \overline{\mathsf{F}}[n, \mu, \{0,1\}^\mu]$ be the IBTDF associated by our construction to parameters n, μ and $\mathsf{IDSp} = \{0,1\}^\mu$. Let A be an adaptive-id adversary that makes a maximal number of $Q < p/(3m)$ queries and let $\delta = (4(\mu+1)Q)^{-1}$. Let $\mathsf{LF} = \overline{\mathsf{LF}}_Q[n, \mu, \{0,1\}^\mu]$ be the sibling associated to F, A as above. Then there is an adversary B such that*

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A) \;\le\; 2n(\mu+2)\cdot \mathbf{Adv}^{\text{dlin}}(B)\;. \tag{15}$$

The running time of B is that of A plus $O(\mu^2\rho^{-1}((\mu Q\rho)^{-1}))$ overhead, where $\rho = \frac{1}{2}\cdot \mathbf{Adv}^{\delta\text{-los}}_{\mathsf{F},\mathsf{LF},\ell}(A)$.

Proof of Theorem 4.4: Our proof uses a simulation technique due to Waters [62]. We used a slightly improved analysis from [42]. Let Q be the number of queries made by A and let algorithm Aux be defined as above. Let $\mathrm{RL}_0, \mathrm{RL}_n$ be the games of Figure 5 with $\mathsf{IDSp} = \{0,1\}^\mu$ and this Aux. Let $\mathrm{E}(IS, id^*)$ denote the event that when $\underline{\mathbf{procFinalize}(d')}$ is called in RL^A_0 the flag $\textsc{Win} \leftarrow \mathsf{false}$ is set and $id^* \notin IS$. (Note that $\eta(IS, id^*)$ only depends on IS, id^* since $\mathbf{y}_0$ is exclusively used to set $\textsc{Win} \leftarrow \mathsf{false}$.) Let $\eta(IS, id^*)$ be

the probability that $\mathrm{E}(IS, id^*)$ happens. In [42, Lemma 6.2], it was shown (using purely combinatorial arguments) that $\lambda_{\mathrm{low}} := \frac{1}{4(\mu+1)Q} \leq \eta(IS, id^*) \leq \frac{1}{2Q} := \lambda_{\mathrm{up}}$. Since RL_0^A and $\mathrm{Real}_{\mathsf{F}}^A$ are only different when $\mathrm{E}(IS, id^*)$ happens, one would like to argue that $\lambda_{\mathrm{low}} \cdot \Pr\left[\mathrm{Real}_{\mathsf{F}}^A\right] = \Pr\left[\mathrm{RL}_0^A\right]$ but this is not true since $\mathrm{E}(IS, id^*)$ and $\mathrm{Real}_{\mathsf{F}}^A$ may not be independent. To get rid of this unwanted dependence we consider a modification of RL_0 and RL_n which adds some artificial abort such that in total it always sets $\textsc{Win} \leftarrow \mathsf{false}$ with probability around $1 - \lambda_{\mathrm{low}}$, independent of the view of the adversary. (Since, given IS, id^*, the exact value of $\eta(IS, id^*)$ cannot be computed efficiently, it needs to be approximated using sampling.) Concretely, games $\hat{\mathrm{RL}}_0$ and $\hat{\mathrm{RL}}_n$ are defined as RL_0 and RL_n, respectively, the only difference being **Finalize** which is defined as follows.

proc Finalize(d') // $\hat{\mathrm{RL}}_0, \hat{\mathrm{RL}}_n$
Compute an approximation $\eta'(IS, id^*)$ of $\eta(IS, id^*)$
If $\eta'(IS, id^*) > \lambda_{\mathrm{low}}$ then set $\textsc{Win} \leftarrow \mathsf{false}$ with probability $1 - \lambda_{\mathrm{low}}/\eta'(IS, id^*)$
Return $((d' = 1)$ and $(id^* \notin IS)$ and $\textsc{Win})$

We refer to [42] on details how to compute the approximation $\eta'(IS, id^*)$. Using [42, Lemma 6.3], one can show that if we use $O(\mu^2 \rho^{-1}((\mu Q \rho)^{-1}))$ samples to compute approximation $\eta'(IS, id^*)$, then

$$\Pr\left[\mathrm{Real}_{\mathsf{F}}^A\right] - \lambda_{\mathrm{low}}^{-1} \cdot \Pr\left[\hat{\mathrm{RL}}_0^A\right] = \rho. \tag{16}$$

Setting $\rho = \frac{1}{2} \cdot \Pr\left[\mathrm{Real}_{\mathsf{F}}^A\right]$ we obtain

$$\delta \cdot \Pr\left[\mathrm{Real}_{\mathsf{F}}^A\right] = \Pr\left[\hat{\mathrm{RL}}_0^A\right], \tag{17}$$

where $\delta = \lambda_{\mathrm{low}}/2$ is as in the theorem statement. As in the proof of Theorem 4.3, we can show that

$$\Pr\left[\mathrm{Lossy}_{\mathsf{F},\mathsf{LF},\ell}^A\right] = \Pr\left[\hat{\mathrm{RL}}_n^A\right]. \tag{18}$$

Now Equation (15) follows from Equations (1), (17), (18), Lemma 4.2 and (a version incorporating the artificial abort of) Lemma 4.1. ∎

We remark that we could use the proof technique of [12] which avoids the artificial abort but this increases the value of δ, making it dependent on the adversary advantage. The proof technique of [41] could be used to strengthen δ in Theorem 4.4 to $O(\sqrt{mQ})^{-1}$ which is close to the optimal value Q^{-1}.

5 IB-TDFs from Lattices

Here we give a construction of a lossy IB-TDF from lattices, specifically, the LWE assumption. We note that a one-way IB-TDF can already be derived by applying methods from [29, 2] to the LWE-based injective (not identity-based) trapdoor function from [36].

LWE is a particular type of average-case BDD/GapSVP problem. It has been recognized since [50] that GapSVP (and BDD [45]) induces a form of lossiness. So there is folklore that the GPV LWE-based TDF can be made to satisfy some meaningful notion of lossiness (specifically, for an appropriate input distribution, the output does not reveal the entire input statistically) by replacing its normally uniformly random key with an LWE (BDD/GapSVP) instance. However, a full construction and proof according to the standard notion of lossiness (which compares the domain and images sizes of the function) have not yet appeared in the literature, and there are many quantitative issues to address.

In this section we construct an (ID-based) TDF that is lossy for a natural (uniform) input distribution. We favor simplicity of analysis at the expense of tight bounds, so our construction is highly unoptimized and should be seen mainly as a proof of feasibility. Much tighter constructions and bounds can be achieved using more sophisticated machinery from the literature.

5.1 Background

For a real matrix $\mathbf{X}$, we let $s_1(\mathbf{X})$ denote its largest singular value (also known as spectral norm), i.e., $s_1(\mathbf{X}) = \max_{\mathbf{y}\neq\mathbf{0}} \|\mathbf{X}\mathbf{y}\|/\|\mathbf{y}\|$. It is easy to verify that the spectral norm satisfies the triangle inequality $s_1(\mathbf{X}+\mathbf{Y}) \leq s_1(\mathbf{X}) + s_1(\mathbf{Y})$ and $s_1(\mathbf{X}\mathbf{Y}) \leq s_1(\mathbf{X})s_1(\mathbf{Y})$. Throughout this section we let n be the main security parameter, and let $\omega(\sqrt{\log n})$ denote a *fixed* function that grows asymptotically faster than $\sqrt{\log n}$.

Probability distributions. The *discrete Gaussian* distribution with parameter $s > 0$ over the integers $\mathbb{Z}$, written $D_{\mathbb{Z},s}$, assigns probability proportional to $\exp(-\pi x^2/s^2)$ to each $x \in \mathbb{Z}$ (and probability zero elsewhere). It is extended to a product distribution over $\mathbb{Z}^n$ in the natural way, i.e., $D_{\mathbb{Z}^n,s} = D_{\mathbb{Z},s}^n$.

We say that a random variable X over $\mathbb{R}$ is *subgaussian* with parameter s if for all $t \geq 0$, we have $\Pr[|X| \geq t] \leq 2\exp(-\pi t^2/s^2)$. More generally, we say that a random vector $\mathbf{x}$ (respectively, a random matrix $\mathbf{X}$) or its distribution is subgaussian of parameter s if all its one-dimensional marginals $\langle \mathbf{x}, \mathbf{u}\rangle$ (respectively, $\mathbf{u}^t\mathbf{X}\mathbf{v}$) for unit vectors $\mathbf{u}, \mathbf{v}$ are subgaussian of parameter s. The concatenation of n independent subgaussian variables with common parameter s, interpreted as either a vector or matrix, is also subgaussian with parameter s. It is also known that $D_{\mathbb{Z},s}$ is subgaussian with parameter s (see [46, Lemma 2.8]). We need the following standard fact from random matrix theory (see, e.g., [60]).

Lemma 5.1 *For a random matrix $\mathbf{X} \in \mathbb{R}^{h\times w}$ that is subgaussian with parameter s, we have $s_1(\mathbf{X}) = s \cdot O(\sqrt{h} + \sqrt{w})$ except with probability $2^{-\Omega(h+w)}$.*

Lattices and LWE. Throughout the remainder of this section we let $q = q(n)$ denote a prime, and $\mathbb{Z}_q$ denote the ring of integers modulo q. It is possible to generalize our constructions to moduli of other forms (e.g., prime powers) using known facts from the literature (see, e.g., [46]), but this somewhat complicates the constructions and the statements of the bounds we use, so we stick with prime moduli for simplicity.

As in many recent papers, we work with a family of "q-ary" lattices (and their cosets), represented by parity-check matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. The precise definition of these lattices will not be needed in this work, so we omit it and refer the interested reader to, e.g., [36] for details. The following lemma is special case of [36, Lemma 5.3] and [46, Lemma 2.4], and the properties of the "smoothing parameter" (see [47, 36]).

Lemma 5.2 *For prime q and integer $b \geq 2$, let $\bar{m} \geq n\log_b q + \omega(\log n)$. With overwhelming probability over the uniformly random choice of $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$, the following holds: for $\mathbf{r} \leftarrow D_{\mathbb{Z},b\cdot\omega(\sqrt{\log n})}^{\bar{m}}$, the distribution of $\mathbf{A}\mathbf{r} \in \mathbb{Z}_q^n$ is negl(n)-far from uniform.*

Note that by the triangle inequality for statistical distance, the above statement also holds where $\mathbf{r}$ is replaced by $\mathbf{R} \leftarrow D_{\mathbb{Z},b\cdot\omega(\sqrt{\log n})}^{\bar{m}\times w}$, and $\mathbf{A}\mathbf{r} \in \mathbb{Z}_q^n$ with $\mathbf{A}\mathbf{R} \in \mathbb{Z}_q^{n\times w}$, for any $w = \mathrm{poly}(n)$.

The (decisional) *learning with errors* (LWE) problem [54] in dimension n with error rate $\alpha \in (0,1)$, stated in matrix form, is: given an input $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ (for any $m = \mathrm{poly}(n)$) where $\mathbf{A}$ is uniformly random, and $\mathbf{b}$ is either of the form $\mathbf{b}^t = \mathbf{x}^t \left[\begin{smallmatrix}\mathbf{I}_m \\ \mathbf{A}\end{smallmatrix}\right] \bmod q$ for $\mathbf{x} \leftarrow D_{\mathbb{Z},\alpha q}^{m+n}$, or is uniformly random and independent of $\mathbf{A}$, distinguish which is the case with non-negligible advantage.[1] By a routine hybrid argument, replacing $\mathbf{x}$ with a matrix $\mathbf{X}$ having any number $w = \mathrm{poly}(n)$ of independent columns (each drawn from $D_{\mathbb{Z},\alpha q}^{m+n}$), and replacing $\mathbf{b}^t$ with either $\mathbf{B}^t = \mathbf{X}^t \left[\begin{smallmatrix}\mathbf{I} \\ \mathbf{A}\end{smallmatrix}\right] \bmod q$ or a uniformly random $\mathbf{B}$ of the same dimension, yields an equivalent problem (up to a w factor in the adversary's advantage). When $\alpha q > 2\sqrt{n}$, this decision problem is at least as hard as approximating several problems on n-dimensional lattices in the *worst case* to within $\tilde{O}(n/\alpha)$ factors with a *quantum* algorithm [54], or via a *classical* algorithm for a subset of these problems [50].

[1] This is actually the "normal form" of the LWE problem, which is equivalent to the one from [54] in which the portion of $\mathbf{x}$ that is multiplied by $\mathbf{A}^t$ is uniformly random in $\mathbb{Z}_q^n$; see, e.g., [6]. In addition, for simplicity of analysis we use a true discrete Gaussian error distribution $D_{\mathbb{Z},\alpha q}$ instead of a "rounded" continuous Gaussian as in [54]; hardness for this error distribution is implied by the results of [51].

Trapdoors for lattices. We recall the notion and efficient construction of a (strong) trapdoor for q-ary lattices, due recently to Micciancio and Peikert [46]. This construction uses a public "gadget" matrix $\mathbf{G}$ over $\mathbb{Z}_q$, defined as

$$\mathbf{G} = \mathbf{I}_n \otimes [1, b, b^2, \ldots, b^{w-1}] \in \mathbb{Z}_q^{n \times nw} \tag{19}$$

for some integer base $b \geq 2$ and $w = \lceil \log_b q \rceil$. (Note that [46] mainly focuses on the case $b = 2$; in our constructions we will need to take b to be larger, but still constant.)

Following [46], we say that an integer matrix $\mathbf{R} \in \mathbb{Z}^{(m-nw) \times nw}$ is a *trapdoor* with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ if $\mathbf{A} \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = \mathbf{H} \cdot \mathbf{G}$. In our constructions, $\mathbf{H}$ will always be either an invertible matrix, or the zero matrix. The trapdoor generation algorithm of [46] works for any $m \geq n(\log_b q + w) + \omega(\log n)$ and generates a nearly uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, together with a trapdoor $\mathbf{R}$ (with a desired tag $\mathbf{H}$) for $\mathbf{A}$. Letting $\bar{m} = m - nw \geq n \log_b q + \omega(\log n)$, it chooses $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random, chooses $\mathbf{R} \leftarrow D_{\mathbb{Z}, b \cdot \omega(\sqrt{\log n})}^{\bar{m} \times nw}$, and lets $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H} \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$. It is clear by inspection that $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, and by Lemma 5.2 the distribution of $\mathbf{A}$ is $negl(n)$-far from uniform.

We recall two of the main operations enabled by a trapdoor: inversion of the (injective) LWE function $g_{\mathbf{A}}(\mathbf{x}) := \mathbf{x}^t \left[\begin{smallmatrix} \mathbf{I} \\ \mathbf{A} \end{smallmatrix} \right] \bmod q$ for "short" integer vectors $\mathbf{x}$, and delegation of a trapdoor for an extended parity-check matrix.

Lemma 5.3 ([46]) *Let $\mathbf{R}$ be a trapdoor with any invertible tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, using a gadget matrix $\mathbf{G}$ with base $b \geq 2$. There are efficient algorithms* Invert *and* DelTrap *that do the following:*

1. *For $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{x}) := \mathbf{x}^t \left[\begin{smallmatrix} \mathbf{I}_m \\ \mathbf{A} \end{smallmatrix} \right] \bmod q$ where $\mathbf{x} \in \mathbb{Z}^{m+n}$ is such that $\|\mathbf{x}\| \leq q/\Theta(b \cdot s_1(\mathbf{R}))$, the algorithm* $\mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ *outputs $\mathbf{x}$.*

2. *For any invertible tag $\mathbf{H}'$, matrix $\mathbf{A}' \in \mathbb{Z}_q^{n \times nw}$, and any sufficiently large $s = \Omega(b \cdot s_1(\mathbf{R})) \cdot \omega(\sqrt{\log n})$, the algorithm* $\mathsf{DelTrap}(\mathbf{R}, [\mathbf{A} \mid \mathbf{A}'], \mathbf{H}', s)$ *outputs a trapdoor $\mathbf{R}'$ with tag $\mathbf{H}'$ for $[\mathbf{A} \mid \mathbf{A}']$, where $\mathbf{R}'$ has the same distribution (up to $negl(n)$ statistical distance) for* any *trapdoor $\mathbf{R}$ satisfying the above bound on $s_1(\mathbf{R})$, and $s_1(\mathbf{R}') = O(\sqrt{m})$ with overwhelming probability.*

5.2 Our basic trapdoor function

Let $c > 1$ and integer base $b \geq 2$ be constants to be determined later in the analysis, and let $\hat{n} = cn$, $m \geq \hat{n} \log_b q = cn \log_b q$ be integers. Define $I_\beta = \{0, 1, \ldots, \beta - 1\}$ and I_γ similarly for some positive integers $\beta \geq \gamma$ to be determined later. (The analysis also goes through unchanged for $I_\beta = [-\beta, \ldots, \beta - 1)$ and I_γ defined similarly.)

1. Parameters: The public parameter *pars* is a matrix $\mathbf{A} \in \mathbb{Z}_q^{\hat{n} \times m}$ (which will be close to uniform, either statistically or computationally), and the trapdoor *msk* is a trapdoor $\mathbf{R}$ (for any invertible tag $\mathbf{H}$) for $\mathbf{A}$ with bounded $s_1(\mathbf{R})$. For a sufficiently large $m = \Omega(\hat{n} \log_b q)$, these can be created using the trapdoor generation algorithm described above, or via the DelTrap algorithm from Item 2 of Lemma 5.3.
2. Evaluate: Given parameter $\mathbf{A}$ and input $\mathbf{x} \in I_\beta^{m+n} \times I_\gamma^{\hat{n}-n}$, algorithm LWE.Ev outputs

$$b^t = g_{\mathbf{A}}(\mathbf{x}) := \mathbf{x}^t \begin{bmatrix} \mathbf{I}_m \\ \mathbf{A}^t \end{bmatrix} \bmod q.$$

3. Invert: Given parameter $\mathbf{A}$, trapdoor $\mathbf{R}$ and output $\mathbf{b}$, algorithm $\mathsf{LWE.Ev}^{-1}$ returns $\mathbf{x}$ using the inversion algorithm from Item 1 of Lemma 5.3.

The next lemma shows that when $\mathbf{A}$ has a particular non-uniform structure (*without* a trapdoor $\mathbf{R}$), the function $g_{\mathbf{A}}$ is lossy when the parameters are set appropriately; we show how to do so after the proof.

Lemma 5.4 *Suppose that $\mathbf{A} \in \mathbb{Z}_q^{\hat{n} \times m}$ is such that*

$$\begin{bmatrix} \mathbf{I}_m \\ \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{m+n} \\ \mathbf{E}^t \end{bmatrix} \begin{bmatrix} \mathbf{I}_m \\ \bar{\mathbf{A}} \end{bmatrix}$$

for some $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{E}^t \in \mathbb{Z}^{(\hat{n}-n)\times(m+n)}$*. Then for* $\mathbf{x} \in I_\beta^{m+n} \times I_\gamma^{\hat{n}-n}$*, the number of distinct output values* $g_{\mathbf{A}}(\mathbf{x})$ *is at most* $O(\beta + \gamma \cdot s_1(\mathbf{E}))^{m+n}$*.*

In particular, for large enough $\gamma^{c-1} \geq 2^{\Omega(m/n)}$ *and* $\beta \geq \gamma \cdot s_1(\mathbf{E})$*, the function* $g_{\mathbf{A}}$ *is* $\Omega(m)$*-lossy.*

Proof: Notice that

$$g_{\mathbf{A}}(\mathbf{x}) = \mathbf{x}^t \begin{bmatrix} \mathbf{I}_m \\ \mathbf{A} \end{bmatrix} = (\mathbf{x}^t \begin{bmatrix} \mathbf{I}_{m+n} \\ \mathbf{E}^t \end{bmatrix}) \begin{bmatrix} \mathbf{I}_m \\ \bar{\mathbf{A}} \end{bmatrix} \bmod q.$$

It therefore suffices to bound the number of possible values of the form $\mathbf{x}^t \left[\begin{smallmatrix} \mathbf{I} \\ \mathbf{E}^t \end{smallmatrix}\right] \in \mathbb{Z}^{m+n}$. By the triangle inequality, we have

$$\left\| \mathbf{x}^t \left[\begin{smallmatrix} \mathbf{I} \\ \mathbf{E}^t \end{smallmatrix}\right] \right\| \leq \beta\sqrt{m+n} + s_1(\mathbf{E}) \cdot \gamma\sqrt{\hat{n}-n} \leq \sqrt{m+n} \cdot (\beta + \gamma \cdot s_1(\mathbf{E})).$$

Define $N_d(r)$ to be the number of integer points in a d-dimensional Euclidean ball of radius r. For $r \geq \sqrt{d}$, from the volume of the ball and Stirling's approximation, we have $N_d(r) = O(r/\sqrt{d})^d$. Therefore, the number of possible values of the form $\mathbf{x}^t \left[\begin{smallmatrix} \mathbf{I} \\ \mathbf{E}^t \end{smallmatrix}\right] \in \mathbb{Z}^{m+n}$ is $O(\beta + \gamma \cdot s_1(\mathbf{E}))^{m+n}$, as claimed.

For lossiness, observe that for our choice of γ, the base-2 logarithm of the domain size of $g_{\mathbf{A}}$ is

$$(m+n)\lg\beta + n\lg\gamma^{c-1} \geq (m+n)\lg\beta + \Omega(m).$$

Whereas by the above, for $\beta \geq \gamma \cdot s_1(\mathbf{E})$ the base-2 logarithm of the image size of $g_{\mathbf{A}}$ is at most

$$(m+n)\lg O(\beta + \gamma \cdot s_1(\mathbf{E})) = (m+n)\lg\beta + O(m).$$

By choosing a sufficiently large universal constant in the above $\Omega(\cdot)$ expression, we have that the two quantities above differ by $\Omega(m)$, as desired. ∎

We now discuss the constraints on the parameters and show how they can be instantiated. The constant c, base b, and integer γ are chosen based on the relationship between m and n. First, we need $\gamma^{c-1} \geq 2^{\Omega(m/n)}$ as required by Lemma 5.4. In order to generate $\mathbf{A}$ with a trapdoor, we will have $m = \Theta(\hat{n}\log_b q) = \Theta(cn\log_b q)$, so we need $\gamma \geq q^{\Theta(1/\log b)\cdot c/(c-1)}$. For any desired constant $C > 1$, we can choose constants $c > 1$ and $b \geq 2$ so that $\gamma \leq q^{1/C}$. Next, we choose β: to accommodate both the upper bound that suffices for invertibility (Item 1 of Lemma 5.3), and the lower bound on β that suffices for $\Omega(m)$-lossiness (Lemma 5.4), it suffices to take

$$q^{1/C} \cdot s_1(\mathbf{E}) \quad \leq \quad \beta \quad \leq \quad q/\Theta(s_1(\mathbf{R}) \cdot \sqrt{m}). \tag{20}$$

These constraints can be satisfied for sufficiently large

$$q^{1-1/C} \quad \geq \quad \Omega(s_1(\mathbf{R}) \cdot s_1(\mathbf{E}) \cdot \sqrt{m}). \tag{21}$$

In all our instantiations, we will have (with $1 - negl(n)$ probability) $s_1(\mathbf{R}) = \mathrm{poly}(n)$ by the use of the trapdoor generation or delegation algorithms, and $s_1(\mathbf{E}) = \mathrm{poly}(n)$ by the use of LWE with error distribution $D_{\mathbb{Z},\alpha q}$ for $\alpha q = \Theta(\sqrt{n})$ to generate a pseudorandom matrix $\mathbf{A}$. Because $1 - 1/C > 0$ is a constant (which may even be chosen arbitrarily close to 1), we can choose a sufficiently large $q = \mathrm{poly}(n)$ so as to satisfy Equation (21), and can use an error rate of $\alpha = \Theta(\sqrt{n})/q = 1/\mathrm{poly}(n)$.

Remark 5.5 As a concrete (but non-identity-based) instantiation, consider a matrix $\mathbf{A}$ having the form described in Lemma 5.4, where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$ is uniformly random and the entries of $\mathbf{E}$ are chosen independently from $D_{\mathbb{Z},\alpha q}$, where $\alpha q = \Theta(\sqrt{n})$ so that we can invoke known worst-case hardness results for LWE. Then we have $s_1(\mathbf{R}) = O(\sqrt{m}) \cdot \omega(\sqrt{\log n}) = \tilde{O}(\sqrt{n})$ and $s_1(\mathbf{E}) = O(\sqrt{mn}) = \tilde{O}(n)$ with overwhelming probability, by subgaussianity of $D_{\mathbb{Z},\alpha q}$ and Lemma 5.1. Moreover, under the LWE assumption (in dimension n) with noise rate α, such an $\mathbf{A}$ is indistinguishable from uniform, which makes the lossy function $g_{\mathbf{A}}$ indistinguishable from an invertible one.

Remark 5.6 Our constructions of ID-based lossy TDFs below involve two small variations on the above example. First, the trapdoor $\mathbf{D}(id)$ for an identity will be delegated (using the $\mathsf{DelTrap}$ algorithm) from a trapdoor $\mathbf{R}(id)$, derived from the master trapdoor $\mathbf{R}$, for which $s_1(\mathbf{R}(id)) \leq \mathrm{poly}(n)$. So we will still have $s_1(\mathbf{D}(id)) \leq s_1(\mathbf{R}(id)) \cdot \mathrm{poly}(n) = \mathrm{poly}(n)$. Second, in the lossy case, the hidden matrix $\mathbf{E}$ in the structured matrix $\mathbf{A}$ will no longer be Gaussian itself, but will be the product of some Gaussian $\mathbf{E}'$ (of parameter αq) and another matrix $\mathbf{X}$ with $s_1(\mathbf{X}) = \mathrm{poly}(n)$, so we will still have $s_1(\mathbf{E}) = \mathrm{poly}(n)$ and can still instantiate all the parameters so that $q, 1/\alpha = \mathrm{poly}(n)$.

5.3 Our id-based lossy trapdoor function

SETUP. As above, let $c > 1$ and integer base $b \geq 2$ be constants to be determined later, and let $\hat{n} = cn$, $\bar{m} = \hat{n}\log_b q + \omega(\log n)$, and $m = \bar{m} + 2\hat{n}w$ where $w = \lceil \log_b q \rceil$. For integer $\mu \geq 1$, let $\mathsf{C} : \mathsf{IDSp} \to \mathbb{Z}_q^{\hat{n}\times\hat{n}} \times \{0,1\}^\mu$ denote an injective encoding of identities that will be instantiated for a specific scheme.

OUR E-IBTDF. Our E-IBTDF $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}]$ is associated with an integer $\mu \geq 1$, an identity space IDSp and an injective encoding C. It has domain $\mathsf{InSp} = I_\beta^{m+n} \times I_\gamma^{\hat{n}-n}$ and auxiliary input space $(\mathbb{Z}_q^{\hat{n}\times\hat{n}})^\mu$, and is given by the following algorithms.

1. Parameters: Given input $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}\times\bar{m}}$ and auxiliary input $\mathbf{H} = (\mathbf{H}[1], \ldots, \mathbf{H}[\mu]) \in (\mathbb{Z}_q^{\hat{n}\times\hat{n}})^\mu$, algorithm $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Pg}$ chooses $\mathbf{R} = (\mathbf{R}[1], \ldots, \mathbf{R}[\mu]) \leftarrow (D^{\bar{m}\times\hat{n}w}_{\mathbb{Z}, b\cdot\omega(\sqrt{\log n})})^\mu$, and lets $\mathbf{U} = (\mathbf{U}[1], \ldots, \mathbf{U}[\mu]) \in (\mathbb{Z}_q^{\hat{n}\times\hat{n}w})^\mu$, where
$$\mathbf{U}[i] := \mathbf{H}[i] \cdot \mathbf{G} - \mathbf{A}\mathbf{R}[i].$$
It also chooses $\mathbf{R}' \leftarrow D^{\bar{m}\times\hat{n}w}_{\mathbb{Z}, b\cdot\omega(\sqrt{\log n})}$ and lets $\mathbf{A}' = \mathbf{A}\mathbf{R}'$. It returns $pars = (\mathbf{A}, \mathbf{A}', \mathbf{U})$ as the public parameters and $msk = (\mathbf{R}, \mathbf{H})$ as the master secret key.

   Note that $\mathbf{R}[i]$ is a trapdoor with tag $\mathbf{H}[i]$ for $[\mathbf{A} \mid \mathbf{U}[i]]$. Moreover, since each $\mathbf{R}[i]$ is subgaussian with parameter $b \cdot \omega(\sqrt{\log n})$, we have (by Lemma 5.1) $s_1(\mathbf{R}[i]) = O(b\sqrt{m}) \cdot \omega(\sqrt{\log n})$ for all i, with overwhelming probability.

   For $pars = (\mathbf{A}, \mathbf{A}', \mathbf{U})$ and a user identity id with $\mathsf{C}(id) = (\mathbf{H}[0], \mathbf{c} \in \{0,1\}^\mu)$, define
$$\mathbf{A}(id) := \Big[\mathbf{A} \mid \mathbf{H}[0] \cdot \mathbf{G} + \sum_{i=1}^{\mu} \mathbf{c}[i]\mathbf{U}[i]\Big].$$
   For $\mathbf{U}$ as constructed by $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Pg}$, we have
$$\mathbf{A}(id) = \Big[\mathbf{A} \mid (\mathbf{H}[0] + \sum_{i=1}^{\mu} \mathbf{c}[i]\mathbf{H}[i]) \cdot \mathbf{G} - \mathbf{A} \cdot \sum_{i=1}^{\mu} \mathbf{c}[i]\mathbf{R}[i]\Big]. \tag{22}$$
   Define
$$\mathbf{R}(id) := \sum_i \mathbf{c}[i]\mathbf{R}[i] \quad \text{and} \quad \mathbf{H}(id) := \mathbf{H}[0] + \sum_i \mathbf{c}[i]\mathbf{H}[i],$$
   and note that $\mathbf{R}(id)$ is a trapdoor with tag $\mathbf{H}(id)$ for $\mathbf{A}(id)$. Moreover, by the above bound on $s_1(\mathbf{R}[i])$ and the triangle inequality, we have $s_1(\mathbf{R}(id)) = O(\mu b\sqrt{m}) \cdot \omega(\sqrt{\log n}) = \mathrm{poly}(n)$ for all id, with overwhelming probability. In what follows we assume that this bound holds.

2. Key generation: Given public parameters $pars = (\mathbf{A}, \mathbf{A}', \mathbf{U})$, master secret $(\mathbf{R}, \mathbf{H})$ and identity $id \in \mathsf{IDSp}$ with $\mathsf{C}(id) = (\mathbf{H}[0], \mathbf{c} \in \{0,1\}^\mu)$, algorithm $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Kg}$ proceeds as follows. It computes $\mathbf{A}(id)$, $\mathbf{R}(id)$, and $\mathbf{H}(id)$ as defined above. Define
$$\mathbf{A}'(id) := [\mathbf{A}(id) \mid \mathbf{A}'].$$
   If $\mathbf{H}(id)$ is invertible, it runs $\mathsf{DelTrap}(\mathbf{R}(id), \mathbf{A}'(id), \mathbf{H}' = \mathbf{I}, s)$ from Item 2 of Lemma 5.3 to generate a trapdoor $\mathbf{D}(id)$ with tag $\mathbf{I}$ for $\mathbf{A}'(id)$, for a sufficiently large $s = \Theta(\mu b^2\sqrt{m}) \cdot \omega(\sqrt{\log n})^2$.

proc Initialize(id) // RL_1
$\mathbf{H}_1 \xleftarrow{\$} \mathsf{Aux}_1(id)$; $\textsc{Win} \leftarrow \mathsf{true}$
$\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$; $\mathbf{E}^t \xleftarrow{\$} D_{\mathbb{Z},\alpha q}^{(\hat{n}-n)\times(\bar{m}+n)}$; $\left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{A}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{E}^t\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I}\\ \bar{\mathbf{A}}\end{smallmatrix}\right]$
$(pars, msk) \xleftarrow{\$} \overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Pg}(\mathbf{A}, \mathbf{H}_1)$
$IS \leftarrow \emptyset$; $id^* \leftarrow id$
Return $pars$

proc Initialize(id) // R_i $(i \in \{0,1\})$
$\mathbf{H}_0 \xleftarrow{\$} \mathsf{Aux}_0(id)$; $\mathbf{H}_1 \xleftarrow{\$} \mathsf{Aux}_1(id)$; $\textsc{Win} \leftarrow \mathsf{true}$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{\hat{n}\times\bar{m}}$
$(pars, msk) \xleftarrow{\$} \overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Pg}(\mathbf{A}, \mathbf{H}_i)$
$IS \leftarrow \emptyset$; $id^* \leftarrow id$
Return $pars$

proc GetDK(id) // $\mathrm{RL}_1, \mathrm{R}_0, \mathrm{R}_1$
$IS \leftarrow IS \cup \{id\}$
If either $\mathbf{H}_0(id)$ or $\mathbf{H}_1(id)$ is not invertible
  then $\textsc{Win} \leftarrow \mathsf{false}$; $dk \leftarrow \perp$
Else $dk \leftarrow \overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Kg}(pars, msk, id)$
Return dk

proc Ch(id) // $\mathrm{RL}_1, \mathrm{R}_0, \mathrm{R}_1$
$id^* \leftarrow id$
If $\mathbf{H}_0(id) \neq \mathbf{0}_{\hat{n}\times\hat{n}}$ or $\mathbf{H}_1(id) \neq \mathbf{0}_{\hat{n}\times\hat{n}}$
  then $\textsc{Win} \leftarrow \mathsf{false}$

proc Finalize(d') // $\mathrm{RL}_1, \mathrm{R}_0, \mathrm{R}_1$
Return $((d' = 1)$ and $(id^* \notin IS)$ and $\textsc{Win})$

Figure 7: Games RL_1 ("Real-to-Losssy") and $\mathrm{R}_0, \mathrm{R}_1$ associated to n, μ, IDSp and auxiliary input generator algorithms Aux_0 and Aux_1.

Note that $s = \Omega(b \cdot s_1(\mathbf{R}(id))) \cdot \omega(\sqrt{\log n})$ as required by Lemma 5.3, and that with overwhelming probability,

$$s_1(\mathbf{D}(id)) = s \cdot O(\sqrt{m}) = O(\mu b^2 m) \cdot \omega(\sqrt{\log n})^2 = \mathrm{poly}(n).$$

3. Evaluate: Given public parameters $pars = (\mathbf{A}, \mathbf{A}', \mathbf{U})$, identity $id \in \mathsf{IDSp}$ and input $\mathbf{x} \in I_\beta^{m+n} \times I_\gamma^{\hat{n}-n}$, algorithm $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Ev}$ computes $\mathbf{A}'(id) = [\mathbf{A}(id) \mid \mathbf{A}']$ as above, and outputs $\mathbf{y} = g_{\mathbf{A}'(id)}(\mathbf{x})$.
4. Invert: Given parameters $(\mathbf{A}, \mathbf{A}', \mathbf{U})$ and identity $id \in \mathsf{IDSp}$ determining $\mathbf{A}'(id)$ as above, trapdoor $\mathbf{D}_{id}$ (with tag $\mathbf{I}$) for $\mathbf{A}'(id)$, and value $\mathbf{y} = g_{\mathbf{A}'(id)}(\mathbf{x})$ as above, algorithm $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Ev}^{-1}$ returns $\mathbf{x}$ using the inversion algorithm from Item 1 of Lemma 5.3.

KEY GENERATION, INVERTIBILITY, AND LOSSINESS. The choice of auxiliary input $\mathbf{H}$ determines the ability to generate keys for identities, i.e., the induced IBTDF $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}](\mathbf{H})$ can generate a key $\mathbf{D}_{id}$ for any id such that $\mathbf{H}(id)$ is invertible. By the upper bound on β from Equation (20), inversion is correct as long as $\beta \leq q/\Theta(s_1(\mathbf{D}_{id}) \cdot \sqrt{m})$.

By contrast, suppose that the $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}\times\bar{m}}$ given to $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}].\mathsf{Pg}$ is such that $\left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{A}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{E}^t\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I}\\ \bar{\mathbf{A}}\end{smallmatrix}\right]$ for some $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{E}^t = [\mathbf{E}_1^t \mid \mathbf{E}_2^t] \in \mathbb{Z}^{(\hat{n}-n)\times\bar{m}} \times \mathbb{Z}^{(\hat{n}-n)\times n}$. (I.e., $\mathbf{A}$ is a structured matrix that satisfies the hypothesis of Lemma 5.4.) Then if $\mathbf{H}(id) = \mathbf{0}$, it can be verified that $\mathbf{A}(id)$ is such that

$$\begin{bmatrix}\mathbf{I}_{\bar{m}+\hat{n}w}\\ \mathbf{A}(id)\end{bmatrix} = \begin{bmatrix}\mathbf{I}_{\bar{m}} & & \\ & \mathbf{I}_{\hat{n}w} & \\ & & \mathbf{I}_n \\ \mathbf{E}_1^t & -\mathbf{E}_1^t \cdot \mathbf{R}(id) & \mathbf{E}_2^t\end{bmatrix}\begin{bmatrix}\mathbf{I}_{\bar{m}} & \\ & \mathbf{I}_{\hat{n}w} \\ \bar{\mathbf{A}} & -\bar{\mathbf{A}} \cdot \mathbf{R}(id)\end{bmatrix},$$

which satisfies the hypothesis of Lemma 5.4 with $[\bar{\mathbf{A}} \mid -\mathbf{A}\cdot\mathbf{R}(id)]$ in place of $\bar{\mathbf{A}}$ and $\tilde{\mathbf{E}}^t = [\mathbf{E}_1^t \mid -\mathbf{E}_1^t\cdot\mathbf{R}(id) \mid \mathbf{E}_2^t]$ in place of $\mathbf{E}^t$. Observe that by the triangle inequality, $s_1(\tilde{\mathbf{E}}) \leq s_1(\mathbf{E})(1+s_1(\mathbf{R}(id))) \leq s_1(\mathbf{E})\,\mathrm{poly}(n)$. In particular, if we have a known $\mathrm{poly}(n)$ upper bound on $s_1(\mathbf{E})$, then as described in the analysis following the proof of Lemma 5.4, we can instantiate the parameters to have correct inversion when $\mathbf{H}(id)$ is invertible, and $\Omega(m)$-lossiness when $\mathbf{H}(id) = \mathbf{0}$.

In what follows we show security of the scheme in the selective-id and adaptive models, under the LWE assumption.

5.4 Real-to-lossy lemma

Consider game RL_1 which is defined as in Figure 7, where $\mathbf{A}$ is such that $\left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{A}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{E}^t\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I}\\ \bar{\mathbf{A}}\end{smallmatrix}\right]$ for uniformly random $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{E}^t \leftarrow D_{\mathbb{Z},\alpha q}^{(\hat{n}-n)\times(\bar{m}+n)}$. Games R_0 and R_1 are defined similarly, where the distribution of $\mathbf{A}$ is uniformly random.

The following lemma says it is hard to distinguish game R_0 from RL_1. We will apply this by defining Aux_0 and Aux_1 in such a way that the output of Aux_0 results in the real scheme and the output of Aux_1 results in a lossy setup.

Lemma 5.7 *Let $n, \mu \geq 1$ be integers and IDSp. Let Aux_0 and Aux_1 be auxiliary input generators for $\overline{\mathsf{L}}[\mu, \mathsf{IDSp}, \mathsf{C}]$ and A an adversary. Then there is an adversary B such that*

$$\Pr[\mathrm{R}_0^A] - \Pr[\mathrm{RL}_1^A] \;\leq\; \mathbf{Adv}^{\mathrm{lwe}}_{n,\alpha}(B) + \mathit{negl}(n)\ . \tag{23}$$

The running time of B is that of A plus some overhead. If A is selective-id then so is B.

The last statement allows us to use the lemma in both the selective-id and adaptive-id cases.

Proof: By Remark 5.5 we have that

$$\Pr[\mathrm{R}_1^A] - \Pr[\mathrm{RL}_1^A] \;\leq\; \mathbf{Adv}^{\mathrm{lwe}}_{n,\alpha}(B)\ . \tag{24}$$

We claim that in R_0 and R_1 (where $\mathbf{A}$ is uniformly random) the values $\mathbf{H}_0$ and $\mathbf{H}_1$ are statistically hidden from A's view. By Lemma 5.2, the tuple $(\mathbf{A}, \mathbf{AR}[1], \ldots, \mathbf{AR}[\mu])$ is $\mathit{negl}(n)$-far from uniformly random. Hence the public parameters $(\bar{\mathbf{A}}, \mathbf{A}', \mathbf{U})$ are $\mathit{negl}(n)$-far from uniform for any fixed choice of the auxiliary input $\mathbf{H}$. Since the execution of the remaining game is independent of whether $\mathbf{H}$ comes from Aux_0 or Aux_1, we obtain

$$\Pr[\mathrm{R}_0^A] - \Pr[\mathrm{R}_1^A] \;\leq\; \mathit{negl}(n)\ . \tag{25}$$

which concludes the proof. ∎

5.5 Selective-id Security

We consider IBTDF $\overline{\mathsf{L}}[\mu = 1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$, the instance of our construction with identity space $\mathsf{IDSp} = \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}$, uniformly random input $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}\times\bar{m}}$, auxiliary input $\mathbf{H}_0 = \mathbf{H}_0[1] = -\mathsf{C}_{\mathrm{FRD}}(\mathbf{0}) \in \mathbb{Z}_q^{\hat{n}\times\hat{n}}$, and identity encoding $\mathsf{C}'_{\mathrm{FRD}}(id) = (\mathsf{C}_{\mathrm{FRD}}(id), 1) \in \mathbb{Z}_q^{\hat{n}\times\hat{n}} \times \{0,1\}$, where $\mathsf{C}_{\mathrm{FRD}} : \mathbb{Z}_q^{\hat{n}} \to \mathbb{Z}_q^{\hat{n}\times\hat{n}}$ is an "invertible differences" encoding as constructed in [2]. (I.e., for each $\mathbf{x} \neq \mathbf{x}'$, the matrix $\mathsf{C}_{\mathrm{FRD}}(\mathbf{x}) - \mathsf{C}_{\mathrm{FRD}}(\mathbf{x}')$ is invertible over $\mathbb{Z}_q$.)

Note that our scheme satisfies the correct inversion requirement because $\mathbf{H}_0(id) = \mathsf{C}_{\mathrm{FRD}}(id) - \mathsf{C}_{\mathrm{FRD}}(\mathbf{0})$ is invertible for all $id \in \mathsf{IDSp} = \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}$. We show that this IBTDF is selective-id δ-lossy for $\delta = 1$, meaning fully selective-id lossy, and hence selective-id one-way. To do this we define a sibling $\overline{\mathsf{LF}}[\mu = 1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$. It preserves the key-generation, evaluation and inversion algorithms of $\overline{\mathsf{L}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$ and alters parameter generation to

Algorithm $\overline{\mathsf{LF}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}].\mathsf{Pg}(id)$:
$\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$; $\mathbf{E}^t \xleftarrow{\$} D_{\mathbb{Z},\alpha q}^{(\hat{n}-n)\times(\bar{m}+n)}$; $\left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{A}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{E}^t\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I}\\ \bar{\mathbf{A}}\end{smallmatrix}\right]$
$\mathbf{H}_1[1] = -\mathsf{C}_{\mathrm{FRD}}(id)$; $(pars, msk) \xleftarrow{\$} \overline{\mathsf{L}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}].\mathsf{Pg}(\mathbf{A}, \mathbf{H_1})$; Return $(pars, msk)$.

The following says that our IBTDF is 1-lossy with lossiness $\Omega(m)$, under the LWE assumption.

Theorem 5.8 *Let $m = c_2 n > c_1 n = \hat{n}$ and $\ell = 2m$. Let $\mathsf{L} = \overline{\mathsf{L}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$ be the IBTDF associated by our construction to parameters $\mu = 1$ and $\mathsf{IDSp} = \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}$. Let $\mathsf{LF} = \overline{\mathsf{LF}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$ be the sibling associated to it as above. Let $\delta = 1$ and let be A a selective-id adversary. Then there is an adversary B such that*

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{L},\mathsf{LF},\ell}(A) \;\leq\; \mathbf{Adv}^{\mathrm{lwe}}_{n,\alpha}(B) + \mathit{negl}\ . \tag{26}$$

The running time of B is that of A plus overhead.

Proof: On input id, let algorithm Aux_0 return $-\mathsf{C}_{\mathrm{FRD}}(\mathbf{0})$ and algorithm Aux_1 return $-\mathsf{C}_{\mathrm{FRD}}(id)$. Let $\mathrm{R}_0, \mathrm{RL}_1$ be the games of Figure 7 with $\mu = 1$, $\mathsf{IDSp} = \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}$ and auxiliary input generators Aux_0 and Aux_1, respectively. Then we claim

$$\Pr\left[\,\mathrm{Real}_{\mathsf{L}}^{A}\,\right] = \Pr\left[\,\mathrm{R}_0^{A}\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{Lossy}_{\mathsf{L},\mathsf{LF},\ell}^{A}\,\right] = \Pr\left[\,\mathrm{RL}_1^{A}\,\right]. \tag{27}$$

To justify this let id^* be the identity queried by A to both **Initialize** and **Ch**. (These queries are the same because A is selective-id.) Then $\mathbf{H}_1 = -\mathsf{C}_{\mathrm{FRD}}(id^*)$ so $\mathbf{H}_1(id) = \mathsf{C}_{\mathrm{FRD}}(id) - \mathsf{C}_{\mathrm{FRD}}(id^*)$. Since $\mathsf{C}_{\mathrm{FRD}}$ is an encoding with invertible differences, this is invertible iff $id \neq id^*$. This means that the conjunct $(id^* \notin IS) \wedge \textsc{Win}$ is always true. The claim of Equation (27) is now true because game R_0 generates parameters with uniform $\mathbf{A}$ and auxiliary input $\mathbf{H}_0 = -\mathsf{C}_{\mathrm{FRD}}(\mathbf{0}) \in \mathbb{Z}_q^{\hat{n}\times\hat{n}}$ that, via $\overline{\mathsf{L}}[1, \mathbb{Z}_q^{\hat{n}} \setminus \{\mathbf{0}\}, \mathsf{C}'_{\mathrm{FRD}}]$, defines L. However game RL_1 generates parameters with auxiliary input $\mathbf{H}_1$. Since $\mathbf{H}_1(id^*) = \mathbf{0}$, the function $g_{\mathbf{A}'(id)}$ is $\Omega(m)$-lossy, as argued immediately following the description of the scheme. ∎

5.6 Full Security

We consider IBTDF $\overline{\mathsf{L}}[\mu, \{0,1\}^\mu, \mathsf{C}']$, the instance of our construction with $\mathsf{IDSp} = \{0,1\}^\mu$, uniformly random input $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}\times\bar{m}}$, auxiliary input $\mathbf{H}_0 = (\mathbf{H}_0[1], \ldots, \mathbf{H}_0[\mu]) := (\mathbf{0}_{\hat{n}\times\hat{n}}, \ldots, \mathbf{0}_{\hat{n}\times\hat{n}})$ and $\mathsf{C}'(id) = (\mathbf{1}_{\hat{n}\times\hat{n}}, \mathsf{C}_f(id))$, where $\mathsf{C}_f : \{0,1\}^\mu \to \mathbb{Z}_q^{\hat{n}\times\hat{n}}$ maps $\mathbf{x} \in \{0,1\}^\mu$ into a vector $\mathbf{X}$ of matrices such that $\mathbf{X}[i] = (-1)^{\mathbf{x}[i]} \cdot \mathbf{1}_{\hat{n}\times\hat{n}} \in \mathbb{Z}_q^{\hat{n}\times\hat{n}}$.

Note that our scheme satisfies the correct inversion requirement because $\mathbf{H}_0(id) = \mathbf{1}_{\hat{n}\times\hat{n}}$ is invertible for all $id \in \mathsf{IDSp}$. We show that this IBTDF is adaptive-id δ-lossy for $\delta = (8Q)^{-1}$ where Q is the number of key-derivation queries of the adversary. By Theorem 3.2 this means $\overline{\mathsf{L}}[\mu, \{0,1\}^\mu, \mathsf{C}']$ is adaptive-id one-way. To do this we define a sibling $\overline{\mathsf{LF}}_Q[\mu, \{0,1\}^\mu, \mathsf{C}']$. It preserves the key-generation, evaluation and inversion algorithms of $\overline{\mathsf{L}}[\mu, \{0,1\}^\mu, \mathsf{C}_f]$ and alters parameter generation to

$$\begin{array}{l}
\text{Algorithm } \overline{\mathsf{LF}}_Q[\mu, \{0,1\}^\mu, \mathsf{C}'].\mathsf{Pg}(id) : \\
\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}} \,;\, \mathbf{E}^t \xleftarrow{\$} D_{\mathbb{Z},\alpha q}^{(\hat{n}-n)\times(\bar{m}+n)} \,;\, \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{A}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{E}^t\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I}\\ \bar{\mathbf{A}}\end{smallmatrix}\right] \\
\mathbf{H}_1 \xleftarrow{\$} \mathsf{Aux}_1 \,;\, (pars, msk) \xleftarrow{\$} \overline{\mathsf{L}}[\mu, \{0,1\}^\mu, \mathsf{C}'].\mathsf{Pg}(\mathbf{A}, \mathbf{H}_1) \,;\; \text{Return } (pars, msk)\,.
\end{array}$$

where Aux_1 is a randomized algorithm from [2, 22] that generates $\mathbf{H}_1 \in (\mathbb{Z}_q^{\hat{n}\times\hat{n}})^\mu$ such that the image of $\mathbf{H}_1(\cdot)$ is either $\mathbf{0}_{\hat{n}\times\hat{n}}$ or invertible and $\mathbf{H}_1(\cdot)$ is "pairwise independent", i.e, for all $id \neq id'$, $\Pr_{\mathsf{Aux}_1}[\mathbf{H}_1(id) = \mathbf{0}_{\hat{n}\times\hat{n}} \mid \mathbf{H}_1(id') = \mathbf{0}_{\hat{n}\times\hat{n}}] = 1/(2Q)$. The following says that our IBTDF is δ-lossy under the LWE assumption with lossiness $\ell = 2m$.

Theorem 5.9 *Let $m = c_2 n > c_1 n = \hat{n}$ and $\ell = 2m$. Let $\mathsf{L} = \overline{\mathsf{L}}[\mu, \{0,1\}^\mu, \mathsf{C}']$ be the IBTDF associated by our construction to parameters μ and $\mathsf{IDSp} = \{0,1\}^\mu$. Let A be an adaptive-id adversary that makes a maximal number of Q queries and let $\delta = (8Q)^{-1}$. Let $\mathsf{LF} = \overline{\mathsf{LF}}_Q[\mu, \{0,1\}^\mu, \mathsf{C}']$ be the sibling associated to L as above. Then there is an adversary B such that*

$$\mathbf{Adv}^{\delta\text{-los}}_{\mathsf{L},\mathsf{LF},\ell}(A) \;\leq\; \mathbf{Adv}^{\mathrm{lwe}}_{n,\alpha}(B) + negl(n)\,. \tag{28}$$

The running time of B is that of A plus polynomial overhead.

Proof: (Sketch) Let Q be the number of queries made by A and let algorithm Aux be defined as above. Let $\mathrm{R}_0, \mathrm{RL}_1$ be the games of Figure 7 with $\mathsf{IDSp} = \{0,1\}^\mu$ and this Aux_0 and Aux_1. Let $\mathrm{E}(IS, id^*)$ denote the event that when $\mathbf{Finalize}(d')$ is called in R_0^A the flag $\textsc{Win} \leftarrow \mathsf{false}$ is set and $id^* \notin IS$. (Note that $\eta(IS, id^*)$ only depends on IS, id^*.) Let $\eta(IS, id^*)$ be the probability that $\mathrm{E}(IS, id^*)$ happens. In [2], it was shown that $\lambda_{\mathrm{low}} := \frac{1}{4Q} \leq \eta(IS, id^*) \leq \frac{1}{2Q} := \lambda_{\mathrm{up}}$. Since R_0^A and $\mathrm{Real}_{\mathsf{L}}^A$ are only different when $\mathrm{E}(IS, id^*)$ happens, one would like to argue that $\lambda_{\mathrm{low}} \cdot \Pr\left[\,\mathrm{Real}_{\mathsf{L}}^{A}\,\right] = \Pr\left[\,\mathrm{R}_0^{A}\,\right]$ but this is not true since $\mathrm{E}(IS, id^*)$ and $\mathrm{Real}_{\mathsf{L}}^A$ may not be independent. To get rid of this unwanted dependence we consider a modification of R_0 and RL_1 which adds some artificial abort such that in total it always sets $\textsc{Win} \leftarrow \mathsf{false}$

with probability around $1 - \lambda_{\text{low}}$, independent of the view of the adversary. (Since, given IS, id^*, the exact value of $\eta(IS, id^*)$ cannot be computed efficiently, it needs to be approximated using sampling.) Concretely, games $\hat{\text{R}}_0$ and $\hat{\text{RL}}_1$ are defined as R_0 and RL_1, respectively, the only difference being **Finalize** which is defined as follows.

proc Finalize(d') // $\hat{\text{R}}_0, \hat{\text{RL}}_1$
Compute an approximation $\eta'(IS, id^*)$ of $\eta(IS, id^*)$
If $\eta'(IS, id^*) > \lambda_{\text{low}}$ then set $\text{WIN} \leftarrow \mathsf{false}$ with probability $1 - \lambda_{\text{low}}/\eta'(IS, id^*)$
Return $((d' = 1)$ and $(id^* \notin IS)$ and $\text{WIN})$

One can again show that with a polynomial number of samples to compute approximation $\eta'(IS, id^*)$,

$$\delta \cdot \Pr\left[\text{Real}_{\mathsf{L}}^{A}\right] = \Pr\left[\hat{\text{R}}_0^{A}\right], \tag{29}$$

where $\delta = \lambda_{\text{low}}/2$ is as in the theorem statement. Similar to the proof of Theorem 5.8, we can show that

$$\Pr\left[\text{Lossy}_{\mathsf{L},\mathsf{LF},\ell}^{A}\right] = \Pr\left[\hat{\text{RL}}_1^{A}\right]. \tag{30}$$

Now Equation (28) follows from Equation (1), Equation (29), Equation (30) and Lemma 5.7. ∎

Acknowledgments

We thank Xiang Xie and the (anonymous) reviewers of Eurocrypt 2012 for their careful reading and valuable comments.

References

[1] P. Abeni, L. Bello, and M. Bertacchini. Exploiting DSA-1571: How to break PFS in SSL with EDH, July 2008. `http://www.lucianobello.com.ar/exploiting_DSA-1571/index.html`. 4

[2] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, May 2010. 2, 3, 18, 24, 25

[3] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Aug. 2010. 2

[4] M. Ajtai. Generating hard instances of the short basis problem. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, July 2099. 3

[5] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, April 2011. Preliminary version in STACS 2009. 3

[6] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Aug. 2009. 19

[7] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Aug. 2007. 1, 4, 31

[8] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249. Springer, Dec. 2009. 1, 4, 31

[9] M. Bellare, S. Halevi, A. Sahai, and S. P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 283–298. Springer, Aug. 1998. 3, 31

[10] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Apr. 2009. 1

[11] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, Jan. 2009. 1

[12] M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, Apr. 2009. 18

[13] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, May 1994. 1

[14] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. 5

[15] C. Bennet, G. Brassard, C. Crépeau, and U. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6), 1995. 4, 31

[16] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Aug. 2008. 1, 4, 31

[17] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004. 1, 2, 3, 8

[18] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Aug. 2004. 1, 2

[19] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Aug. 2004. 2

[20] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, May 2004. 4, 31

[21] D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 1, 2

[22] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, May 2010. 25

[23] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, Aug. 2006. 2, 3, 8

[24] X. Boyen and B. Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In J. Zhou and M. Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 35–52. Springer, June 2010. 1

[25] D. R. Brown. A weak randomizer attack on RSA-OAEP with e=3. IACR ePrint Archive, Report 2005/189, 2005. http://eprint.iacr.org/. 4

[26] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, May 1999. 1

[27] R. Canetti and R. R. Dakdouk. Towards a theory of extractable functions. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 595–613. Springer, Mar. 2009. 4

[28] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, May 2003. 2

[29] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, May 2010. 2, 3, 18

[30] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Dec. 2001. 1

[31] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1

[32] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 130–144. Springer, Jan. 2003. 1

[33] L. Dorrendorf, Z. Gutterman, and B. Pinkas. Cryptanalysis of the windows random number generator. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 476–485. ACM Press, Oct. 2007. 4

[34] A. Escala, J. Herranz, B. Libert, and C. Ráfols. Hierarchical identity-based (lossy) trapdoor functions, May 2012. Manuscript. 4

[35] D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295. Springer, May 2010. 1

[36] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. 1, 2, 3, 18, 19

[37] I. Goldberg and D. Wagner. Randomness in the Netscape browser. Dr. Dobb's Journal, January 1996. 4

[38] Z. Gutterman and D. Malkhi. Hold your sessions: An attack on Java session-id generation. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 44–57. Springer, Feb. 2005. 4

[39] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 4, 31

[40] B. Hemenway and R. Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. Electronic Colloquium on Computational Complexity TR09-127, 2009. 1

[41] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Aug. 2008. 18

[42] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theor. Comput. Sci.*, 410(47-49):5093–5111, 2009. 17, 18

[43] E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, May 2010. 4

[44] E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 295–313. Springer, Aug. 2010. 1

[45] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Aug. 2009. 2, 18

[46] D. Micciancio and C. Peikert. Trapdoors for lattices: simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, LNCS. Springer, 2012. 3, 19, 20

[47] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, Oct. 2004. 19

[48] M. Mueller. Debian OpenSSL predictable PRNG bruteforce SSH exploit, May 2008. `http://milw0rm.com/exploits/5622`. 4

[49] K. Ouafi and S. Vaudenay. Smashing SQUASH-0. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 300–312. Springer, Apr. 2009. 4

[50] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. 2, 18, 19

[51] C. Peikert. An efficient and parallel gaussian sampler for lattices. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Aug. 2010. 19

[52] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008. 1, 2, 3, 4, 6, 8

[53] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. 2

[54] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005. 19

[55] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978. 1

[56] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, May / June 2006. 1

[57] A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, Mar. 2009. 4

[58] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, Jan. 2000. 1, 2

[59] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Aug. 1985. 1

[60] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, January 2011. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`, last accessed 4 Feb 2011. 19

[61] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Aug. 2009. 2

[62] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005. 1, 2, 3, 17

[63] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In *IMC 2009*. ACM, 2009. 4

A Anonymous IBE

In this section we describe an IBE scheme that is similar to IBE from Section 4 with the difference that it encrypts group elements (rather than bits) and it is slightly more efficient. We associate to any integer $\mu \geq 1$ and any identity space $\mathsf{IDSp} \subseteq \mathbb{Z}_p^\mu$ an IBE scheme $\mathsf{IBE}'[\mu, \mathsf{IDSp}]$ that has message space $\mathbb{G}_T^*$ and algorithms as follows:

1. Parameters: Algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Pg}$ lets $g \xleftarrow{\$} \mathbb{G}^*$; $t, z \xleftarrow{\$} \mathbb{Z}_p^*$; $\hat{g} \leftarrow g^t$; $Z \leftarrow \mathbf{e}(g,g)^z$. It then lets $H, \hat{H}, U \xleftarrow{\$} \mathbb{G}$; $\mathbf{U} \xleftarrow{\$} \mathbb{G}^{\mu+1}$. It returns $pars = (g, \hat{g}, H, U, \hat{H}, \mathbf{U}, Z)$ as the public parameters and $msk = (t, z)$ as the master secret key.

2. Key generation: Given parameters $(g, \hat{g}, H, U, \mathbf{U}, Z)$, master secret (t, z) and identity $id \in \mathsf{IDSp}$, algorithm $\mathsf{IBE}'[\mu, \mathsf{IDSp}].\mathsf{Kg}$ returns decryption key (D_1, D_2, D_3, D_4) computed by letting $r, \hat{r} \xleftarrow{\$} \mathbb{Z}_p$ and setting

$$D_1 \leftarrow g^z \cdot \mathcal{H}(\mathbf{U}, id)^{tr} \cdot H^{t\hat{r}} \; ; \; D_2 \leftarrow U^r \cdot H^{\hat{r}} \; ; \; D_3 \leftarrow g^{-tr} \; ; \; D_4 \leftarrow g^{-t\hat{r}} \; .$$

3. Encryption: Given parameters $(g, \hat{g}, H, U, \mathbf{U}, Z)$, identity $id \in \mathsf{IDSp}$ and message $M \in \mathbb{G}_T^*$, algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Enc}$ returns ciphertext $(C_1, C_2, C_3, C_4, C_5)$ computed as follows. It lets $s, \hat{s} \xleftarrow{\$} \mathbb{Z}_p$ and

$$C_1 \leftarrow g^s \; ; \; C_2 \leftarrow \hat{g}^{\hat{s}} \; ; \; C_3 \leftarrow \mathcal{H}(\mathbf{U}, id)^s \cdot U^{\hat{s}} \; ; \; C_4 \leftarrow H^{s+\hat{s}} \; ; \; C_5 \leftarrow Z^{-s} \cdot M \; .$$

4. Decryption: Given parameters $(g, \hat{g}, H, U, \mathbf{U}, Z)$, identity $id \in \mathsf{IDSp}$, decryption key (D_1, D_2, D_4, D_4) for id and ciphertext $(C_1, C_2, C_3, C_4, C_5)$, algorithm $\mathsf{IBE}[\mu, \mathsf{IDSp}].\mathsf{Dec}$ returns

$$M = \mathbf{e}(C_1, D_1)\mathbf{e}(C_2, D_2)\mathbf{e}(C_3, D_3)\mathbf{e}(C_4, D_4)C_5 \; .$$

Compared to $\mathsf{IBE}[\mu, \mathsf{IDSp}]$ from Section 4 , the efficiency improvement consists of replacing $\mathcal{H}(\hat{\mathbf{U}}, id)$ by U in the computation of D_2 and C_3 and of setting $\hat{H} := H$. Using the techniques of the ciphertext pseudorandomness lemma (Lemma 4.1) one can show that the elements (C_1, C_2, C_3, C_4) of the ciphertext are pseudorandom. (Here the reduction knows the secret z.) In a final similar hybrid step one can also show that, under the Bilinear Diffie-Hellman assumption (which is implied by the DLIN assumption), the element C_5 is also pseudorandom. (Here is reduction knows the secret t.) As our main ID-based TDF result uses anonymous IBE techniques, the main ideas of this systems security is implicit in our main proof. A formal proof of the above stand alone system is deferred to the full version.

B Applications

We expand first on the application to achieving deterministic IBE and then on achieving hedged IBE.

D-PKE. Deterministic PKE (D-PKE) cannot achieve IND-CPA security. Bellare, Boldyreva and O'Neill [7] defined a target notion PRIV for it that captures the best possible security under the condition that encryption is deterministic. D-PKE provides a way to do fast (logarithmic time) search on encrypted data. PEKS [20] offers higher security but takes linear time, and trading some security for a significant increase in searching speed is attractive for large databases.

Achieving PRIV for D-PKE has been (and remains) a challenge. It is possible in the RO model [7]. The best results without ROs are due to Boldyreva, Fehr and O'Neill [16], who show how to achieve PRIV without random oracles for message sequences which are blocksources, meaning each message has some min-entropy even given the previous ones. Using the Leftover Hash Lemma (LHL) [15, 39], they show that any LTDF is a D-PKE scheme that is PRIV-secure for blocksources as long as the lossy branch is a universal hash function.

D-IBE. We introduce deterministic IBE (D-IBE). The PRIV definition is easily extended to this setting. D-IBE offers, over D-PKE, the same advantages that IBE offers over PKE, for example that there are no certificates and encryption depends only on the identity of the receiver. Again, D-IBE can be achieved in the RO model by setting the coins of an IBE scheme to the RO-hash of the message. (This is how PKE is turned into D-PKE in the RO model in [9, 7].) We ask what can be done without ROs.

We show that our constructions of DLIN-based lossy IB-TDFs have the properties necessary to obtain PRIV-secure D-IBE schemes for blocksources under the paradigm of [16] in the selective case. We start by observing that the lossy branches are universal hash functions. This can be seen from Equations (3), (4), (5) and (6). In the lossy case, $f(\mathbf{y}, id) = 0$, and the function has a range R of size p^2. Now if x_1, x_2 are distinct inputs, then the outputs of the function on them collide exactly when $(\langle \mathbf{s}, x_1 \rangle, \langle \hat{\mathbf{s}}, x_1 \rangle) = (\langle \mathbf{s}, x_2 \rangle, \langle \hat{\mathbf{s}}, x_2 \rangle)$. The probability that this happens when $\mathbf{s}, \hat{\mathbf{s}}$ are chosen at random from $\mathbb{Z}_p^n$ is $1/p^2 = 1/|R|$.

Hedged IBE. The definitions and methods of [8] can be extended to the identity-based setting in a straightforward way in the selective setting once we have universal lossy IB-TDFs. There are two approaches. One is generic composition of an IBE scheme with a IB-TDF. The other is to first pad the message with randomness and then apply the IB-TDF.

Adaptive setting. It remains open to achieve deterministic or hedged IBE in the adaptive security setting.

A Toolkit for Ring-LWE Cryptography

Vadim Lyubashevsky* Chris Peikert† Oded Regev‡

May 16, 2013

Abstract

Recent advances in lattice cryptography, mainly stemming from the development of ring-based primitives such as ring-LWE, have made it possible to design cryptographic schemes whose efficiency is competitive with that of more traditional number-theoretic ones, along with entirely new applications like fully homomorphic encryption. Unfortunately, realizing the full potential of ring-based cryptography has so far been hindered by a lack of practical algorithms and analytical tools for working in this context. As a result, most previous works have focused on very special classes of rings such as power-of-two cyclotomics, which significantly restricts the possible applications.

We bridge this gap by introducing a toolkit of fast, modular algorithms and analytical techniques that can be used in a wide variety of ring-based cryptographic applications, particularly those built around ring-LWE. Our techniques yield applications that work in *arbitrary* cyclotomic rings, with *no loss* in their underlying worst-case hardness guarantees, and very little loss in computational efficiency, relative to power-of-two cyclotomics. To demonstrate the toolkit's applicability, we develop a few illustrative applications: two variant public-key cryptosystems, and a "somewhat homomorphic" symmetric encryption scheme. Both apply to arbitrary cyclotomics, have tight parameters, and very efficient implementations.

1 Introduction

The past few years have seen many exciting developments in lattice-based cryptography. Two such trends are the development of schemes whose efficiency is competitive with traditional number-theoretic ones (e.g., [Mic02] and follow-ups), and the breakthrough work of Gentry [Gen09b, Gen09a] (followed by others) on fully homomorphic encryption. While these two research threads currently occupy opposite ends of the efficiency spectrum, they are united by their use of algebraically structured *ideal lattices* arising from polynomial rings. The most efficient and advanced systems in both categories rely on the ring-LWE problem [LPR10], an analogue of the standard *learning with errors* problem [Reg05]. Informally (and a bit inaccurately), in a ring $R = \mathbb{Z}[X]/(f(X))$ for monic irreducible $f(X)$ of degree n, and for an integer

*INRIA and École Normale Supérieure, Paris. Part of this work was performed while at Tel Aviv University and also while visiting Georgia Tech. Partially supported by a European Research Council (ERC) Starting Grant.

†School of Computer Science, College of Computing, Georgia Institute of Technology. This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by DARPA under agreement number FA8750-11-C-0096, and by the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA or the U.S. Government, or the Sloan Foundation. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

‡Courant Institute, New York University. Supported by a European Research Council (ERC) Starting Grant. Part of the work done while the author was with the CNRS, DI, ENS, Paris.

modulus q defining the quotient ring $R_q := R/qR = \mathbb{Z}_q[X]/(f(X))$, the ring-LWE problem is to distinguish pairs $(a_i, b_i = a_i \cdot s + e_i) \in R_q \times R_q$ from uniformly random pairs, where $s \in R_q$ is a random secret (which stays fixed over all pairs), the $a_i \in R_q$ are uniformly random and independent, and the error (or "noise") terms $e_i \in R$ are independent and "short."

In all applications of ring-LWE, and particularly those related to homomorphic encryption, a main technical challenge is to control the sizes of the noise terms when manipulating ring-LWE samples under addition, multiplication, and other operations. For correct decryption, q must be chosen large enough so that the final accumulated error terms do not "wrap around" modulo q and cause decryption error. On the other hand, the *error rate* (roughly, the ratio of the noise magnitude to the modulus q) of the original published ring-LWE samples and the dimension n trade off to determine the theoretical and concrete hardness of the ring-LWE problem. Tighter control of the noise growth therefore allows for a larger initial error rate, which permits a smaller modulus q and dimension n, which leads to smaller keys and ciphertexts, and faster operations for a given level of security.

Regarding the choice of ring, the class of *cyclotomic* rings $R \cong \mathbb{Z}[X]/\Phi_m(X)$, where $\Phi_m(X)$ is the mth cyclotomic polynomial (which has degree $n = \varphi(m)$ and is monic and irreducible over the rationals), has many attractive features that have proved very useful in cryptography. For example, the search/decision equivalence for ring-LWE in arbitrary cyclotomics [LPR10] relies on their special algebraic properties, as do many recent works that aim for more efficient fully homomorphic encryption schemes (e.g., [SV11, BGV12, GHS12a, GHS12b, GHPS12]). In particular, *power-of-two* cyclotomics, i.e., where the index $m = 2^k$ for some $k \geq 1$, are especially nice to work with, because (among other reasons) $n = m/2$ is also a power of two, $\Phi_m(X) = X^n + 1$ is maximally sparse, and polynomial arithmetic modulo $\Phi_m(X)$ can be performed very efficiently using just a slight tweak of the classical n-dimensional FFT (see, e.g., [LMPR08]). Indeed, power-of-two cyclotomics have become the dominant and preferred class of rings in almost all recent ring-based cryptographic schemes (e.g., [LMPR08, LM08, Lyu09, Gen09b, Gen10, LPR10, SS11, BV11b, BGV12, GHS12a, GHS12b, Lyu12, BPR12, MP12, GLP12, GHPS12]), often to the exclusion of all other rings.

While power-of-two cyclotomic rings are very convenient to use, there are several reasons why it is essential to consider other cyclotomics as well. The most obvious, practical reason is that powers of two are sparsely distributed, and the desired concrete security level for an application may call for a ring dimension much smaller than the next-largest power of two. So restricting to powers of two could lead to key sizes and runtimes that are at least twice as large as necessary. A more fundamental reason is that certain applications, such as the above-mentioned works that aim for more efficient (fully) homomorphic encryption, *require* the use of non-power-of-two cyclotomic rings. This is because power-of-two cyclotomics lack the requisite algebraic properties needed to implement features like SIMD operations on "packed" ciphertexts, or plaintext spaces isomorphic to finite fields of characteristic two (other than $\mathbb{F}_2$ itself). A final important reason is diversification of security assumptions. While some results are known [GHPS12] that relate ring-LWE in cyclotomic rings when one index m divides the other, no other connections appear to be known. So while we might conjecture that ring-LWE and ideal lattice problems are hard in *every* cyclotomic ring (of sufficiently high dimension), some rings might turn out to be significantly easier than others.

Unfortunately, working in non-power-of-two cyclotomics is rather delicate, and the current state of affairs is unsatisfactory in several ways. Unlike the special case where m is a power of two, in general the cyclotomic polynomial $\Phi_m(X)$ can be quite "irregular" and dense, with large coefficients. While in principle, polynomial arithmetic modulo $\Phi_m(X)$ can still be done in $O(n \log n)$ scalar operations (on high-precision complex numbers), the generic algorithms for achieving this are rather complex and hard to implement, with large constants hidden by the $O(\cdot)$ notation.

Geometrically, the non-power-of-two case is even more problematic. If one views $\mathbb{Z}[X]/(\Phi_m(X))$ as the set of polynomial residues of the form $a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$, and uses the naïve "coefficient embedding" that views them as vectors $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$ to define geometric quantities like the ℓ_2 norm, then both the concrete and theoretical security of cryptographic schemes depend heavily on the form of $\Phi_m(X)$. This stems directly from the fact that multiplying two polynomials with small norms can result in a polynomial residue having a *much* larger norm. The growth can be quantified by the "expansion factor" [LM06] of $\Phi_m(X)$, which unfortunately can be very large, up to $n^{\Omega(\log n)}$ in the case of highly composite m [Erd46]. Later works [GHS12a] circumvented such large expansion by using tricks like lifting to the larger-dimensional ring $\mathbb{Z}[X]/(X^m - 1)$, but this still involves a significant loss in the tolerable noise rates as compared with the power-of-two case.

In [PR07, LPR10] a different geometric approach was used, which avoided any dependence on the form of the polynomial modulus $\Phi_m(X)$. In these works, the norm of a ring element is instead defined according to its *canonical embedding* into $\mathbb{C}^n$, a classical concept from algebraic number theory. This gives a much better way of analyzing expansion, since both addition and multiplication in the canonical embedding are simply coordinate-wise. Working with the canonical embedding, however, introduces a variety of practical issues, such as how to efficiently generate short noise terms having appropriate distributions over the ring. More generally, the focus of [LPR10] was on giving an abstract mathematical definition of ring-LWE and proving its hardness under worst-case ideal lattice assumptions; in particular, it did not deal with issues related to practical efficiency, bounding noise growth, or designing applications in non-power-of-two cyclotomics.

1.1 Contributions

Our main contribution is a toolkit of modular algorithms and analytical techniques that can be used in a wide variety of ring-based cryptographic applications, particularly those built around ring-LWE. The high-level summary is that using our techniques, one can design applications to work in *arbitrary* cyclotomic rings, with *no loss* in their underlying worst-case hardness guarantees, and very little loss in computational efficiency, relative to the best known techniques in power-of-two cyclotomics. In fact, our analytical techniques even improve the state of the art for the power-of-two case.

In more detail, our toolkit includes fast, specialized algorithms for all the main cryptographic operations in arbitrary cyclotomic rings. Among others, these include: addition, multiplication, and conversions among various useful representations of ring elements; generation of noise terms under probability distributions that guarantee both worst-case and concrete hardness; and decoding of noise terms as needed in decryption and related operations. Our algorithms' efficiency and quality guarantees stem primarily from our use of simple but non-obvious representations of ring elements, which differ from their naïve representations as polynomial residues modulo $\Phi_m(X)$. (See the second part of Section 1.2 for more details.) On the analytical side, we give tools for tightly bounding noise growth under operations like addition, multiplication, and round-off/discretization. (Recall that noise growth is the main factor determining an application's parameters and noise rates, and hence its key sizes, efficiency, and concrete security.)
Some attractive features of the toolkit include:

- All the algorithms for arbitrary cyclotomics are simple, modular, and highly parallel, and work by elementary reductions to the (very simple) prime-index case. In particular, they do not require any polynomial reductions modulo $\Phi_m(X)$ – in fact, they never need to compute $\Phi_m(X)$ at all! The algorithms work entirely on vectors of dimension $n = \varphi(m)$, and run in $O(n \log n)$ or even $O(nd)$ scalar operations (with small hidden constants), where d is the number of distinct primes dividing m. With the exception of continuous noise generation, all scalar operations are low precision, i.e., they

involve small integers. In summary, the algorithms are very amenable to practical implementation. (Indeed, we have implemented all the algorithms from scratch, which will be described in a separate work.)

- Our algorithm for decoding noise, used primarily in decryption, is fast (requiring $O(n \log n)$ or fewer small-integer operations) and correctly recovers from optimally large noise rates. (See the last part of Section 1.2 for details.) This improves upon prior techniques, which in general have worse noise tolerance by anywhere between $m/2$ and super-polynomial $n^{\omega(1)}$ factors, and are computationally slower and more complex due to polynomial reduction modulo $\Phi_m(X)$, among other operations.

- Our bounds on noise growth under ring addition and multiplication are exactly the same in *all* cyclotomic rings; no ring-dependent "expansion factor" is incurred. (For discretizing continuous noise distributions, our bounds are the same up to very small $1 + o(1)$ factors, depending on the primes dividing m.) This allows applications to use essentially the same underlying noise rate as a function of the ring dimension n, and hence be based on the same worst-case approximation factors, for all cyclotomics. Moreover, our bounds improve upon the state of the art even for power-of-two cyclotomics: e.g., our (average-case, high probability) expansion bound for ring multiplication improves upon the (worst-case) expansion-factor bound by almost a $\sqrt{n}$ factor.

To illustrate the toolkit's applicability, in Section 8 we develop the following illustrative applications:

1. A simple adaptation of the "dual" LWE-based public-key cryptosystem of [GPV08], which can serve as a foundation for (hierarchical) identity-based encryption. (See Section 8.1.)

2. An efficient and compact public-key cryptosystem, which is essentially the "two element" system outlined in [LPR10], but generalized to arbitrary cyclotomics, and with tight parameters. (See Section 8.2.)

3. A "somewhat homomorphic" symmetric encryption scheme, which follows the template of the Brakerski-Vaikuntanathan [BV11a] and Brakerski-Gentry-Vaikuntanathan [BGV12] schemes in power-of-two cyclotomics, but generalized to arbitrary cyclotomics and with much tighter noise analysis. This application exercises all the various parts of the toolkit more fully, especially in its modulus-reduction and key-switching procedures. (See Section 8.3.)

A final contribution of independent interest is a new "regularity lemma" for arbitrary cyclotomics, i.e., a bound on the smoothing parameter of random q-ary lattices over the ring. Such a lemma is needed for porting many applications of standard LWE (and the related "short integer solution" SIS problem) to the ring setting, including SIS-based signature schemes [GPV08, CHKP10, Boy10, MP12], the "primal" [Reg05] and "dual" [GPV08] LWE cryptosystems (as in Section 8.1), chosen ciphertext-secure encryption schemes [Pei09, MP12], and (hierarchical) identity-based encryption schemes [GPV08, CHKP10, ABB10]. In terms of generality and parameters, our lemma essentially subsumes a prior one of Micciancio [Mic02] for the ring $\mathbb{Z}[X]/(X^n - 1)$, and an independent one of Stehlé et al. [SSTX09] for power-of-two cyclotomics. See Section 7 for further discussion.

Following the preliminary publication of this work, our toolkit has also been used centrally in the "ring-switching" technique for homomorphic encryption [GHPS12], and to give efficient "bootstrapping" algorithms for fully homomorphic encryption [AP13].

1.2 Techniques

The tools we develop in this work involve several novel applications of classical notions from algebraic number theory. In summary, our results make central use of: (1) the *canonical embedding* of a number field, which endows the field (and its subrings) with a nice and easy-to-analyze geometry; (2) the decomposition of arbitrary cyclotomics into the *tensor product* of prime-power cyclotomics, which yields both simpler and faster algorithms for computing in the field, as well as geometrically nicer bases; and (3) the "*dual*" ideal $R^\vee$ and its "*decoding*" basis $\vec{d}$, for fast noise generation and optimal noise tolerance in decryption and related operations. We elaborate on each of these next.

The canonical embedding. As in the previous works [PR07, LPR10], our analysis relies heavily on using the *canonical embedding* $\sigma\colon K \to \mathbb{C}^n$ (rather than, say, the naïve coefficient embedding) for defining all geometric quantities, such as Euclidean norms and inner products. For example, under the canonical embedding, the "expansion" incurred when multiplying by an element $a \in K$ is characterized exactly by $\|\sigma(a)\|_\infty$, its ℓ_∞ norm under the canonical embedding; no (worst-case) ring-dependent "expansion factor" is needed. So in the average-case setting, where the multiplicands are random elements from natural noise distributions, for each multiplication we get at least a $\tilde{\Omega}(\sqrt{n})$ factor improvement over using the expansion factor in *all* cyclotomics (including those with power-of-two index), and up to a super-polynomial $n^{\omega(1)}$ factor improvement in cyclotomics having highly composite indices. In our analysis of the noise tolerance of decryption, we also get an additional $\tilde{\Omega}(\sqrt{n})$ factor savings over more simplistic analyses that only use norm information, by using the notion of *subgaussian* random variables. These behave under linear transformations in essentially the same way as Gaussians do, and have Gaussian tails. (Prior works that use subgaussianity in lattice cryptography include [AP09, MP12].)

Tensorial decomposition. An important fact at the heart of this work is that the mth cyclotomic number field $K = \mathbb{Q}(\zeta_m) \cong \mathbb{Q}[X]/(\Phi_m(X))$ may instead be viewed as (i.e., is isomorphic to) the *tensor product* of prime-power cyclotomics:

$$K \cong \bigotimes\nolimits_\ell K_\ell = \mathbb{Q}(\zeta_{m_1}, \zeta_{m_2}, \ldots),$$

where $m = \prod_\ell m_\ell$ is the prime-power factorization of m and $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$. Equivalently, in terms of polynomials we may view K as the multivariate field

$$K \cong \mathbb{Q}[X_1, X_2, \ldots]/(\Phi_{m_1}(X_1), \Phi_{m_2}(X_2), \ldots), \tag{1.1}$$

where there is one indeterminant X_ℓ and modulus $\Phi_{m_\ell}(X_\ell)$ per prime-power divisor of m. Similar decompositions hold for the ring of integers $R \cong \mathbb{Z}[X]/\Phi_m(X)$ and other important objects in K, such as the dual ideal $R^\vee$ (described below).

Adopting the polynomial interpretation of K from Equation (1.1) for concreteness, notice that a natural $\mathbb{Q}$-basis is the set of multinomials $\prod_\ell X_\ell^{j_\ell}$ for each choice of $0 \le j_\ell < \varphi(m_\ell)$. We call this set the "powerful" basis of K (and of R). Interestingly, for non-prime-power m, under the field isomorphism with $\mathbb{Q}[X]/(\Phi_m(X))$ that maps each $X_\ell \to X^{m/m_\ell}$, the powerful basis does *not* coincide with the standard "power" basis $1, X, X^2, \ldots, X^{\varphi(m)-1}$ usually used to represent the univariate field. It turns out that in general, the powerful basis has much nicer computational and geometric properties than the power basis, as we outline next.

Computationally, the tensorial decomposition of K (with the powerful basis) allows us to modularly reduce operations in K (or R, or powers of $R^\vee$) to their counterparts in much simpler prime-power cyclotomics (which themselves easily reduce to the prime-index case). We can therefore completely avoid all the

many algorithmic complications associated with working with polynomials modulo $\Phi_m(X)$. In particular, we obtain novel, simple and fast algorithms, similar to the FFT, for converting between the multivariate "polynomial" representation (i.e., the powerful basis) and the "evaluation" or "Chinese remainder" representation, in which addition and multiplication are essentially linear time. Similarly, we obtain linear-time (or nearly so) algorithms for switching between the polynomial representation and the "decoding" representation used in decryption (described below), and for generating noise terms in the decoding representation. A final advantage of the tensorial representation is that it yields trivial linear-time algorithms for computing the *trace* function to cyclotomic subfields of K.

The tensorial representation also comes with important geometrical advantages. In particular, under the canonical embedding the powerful basis is better-conditioned than the power basis, i.e., the ratio of its maximal and minimal singular values can be much smaller. This turns out to be important when bounding the additional error introduced when discretizing (rounding off) field elements in noise-generation and modulus-reduction algorithms, among others.

The dual ideal $R^\vee$ and its decoding basis. Under the canonical embedding, the cyclotomic ring R of index m embeds as a lattice which, unlike $\mathbb{Z}^n$, is in general not self-dual. Instead, its dual lattice corresponds to a fractional ideal $R^\vee \subset K$ satisfying $R \subseteq R^\vee \subseteq m^{-1}R$, where the latter inclusion is nearly an equality. (In fact, $R^\vee$ is a scaling of R exactly when m is a power of two, in which case $R = (m/2)R^\vee$.) In [LPR10] it is shown that the "right" definition of the ring-LWE distribution, which arises naturally from the worst-case to average-case reduction, involves the dual ideal $R^\vee$: the secret belongs to the quotient $R_q^\vee = R^\vee/qR^\vee$ (or just $R^\vee$), and ring-LWE samples are of the form $(a, b = a \cdot s + e \bmod qR^\vee)$ for uniformly random $a \in R_q$ and error e which is essentially spherical in the canonical embedding.

While it is possible [DD12] to simplify the ring-LWE distribution by replacing every instance of $R^\vee$ with R, while retaining essentially spherical error (but scaled up by about m, corresponding to the approximate ratio of R to $R^\vee$), in this work we show that *it is actually advantageous to retain $R^\vee$ and expose it in applications*.[1] The reason is that in general, $R^\vee$ supports correct bounded-distance decoding—which is the main operation performed in decryption—under a larger error rate than R does.[2] In fact, the error tolerance of $R^\vee$ is *optimal* for the simple, fast lattice decoding algorithm used implicitly in essentially all decryption procedures, namely Babai's "round-off" algorithm [Bab85]. The reason is that when decoding a lattice Λ using some basis $\{\mathbf{b}_i\}$, the error tolerance depends inversely on the Euclidean lengths of the vectors dual to $\{\mathbf{b}_i\}$. For $R^\vee$, there is a particular "*decoding*" basis whose dual basis is optimally short (relative to the determinant of R), whereas for R no such basis exists in general.[3] In fact, the decoding basis of $R^\vee$ is simply the dual of the (conjugate of the) powerful basis described above!

In addition to its optimal error tolerance, we also show that the decoding basis has good computational properties. In particular, there are linear-time (or nearly so) algorithms for converting to the decoding basis from the other bases of $R^\vee$ or $R_q^\vee$ that are more appropriate for other computational tasks. And Gaussian errors, especially spherical ones, can be sampled in essentially linear time in the decoding basis.

[1] This is unless m is a power of two, in which case nothing is lost by simply scaling up by exactly $m/2$ to replace $R^\vee$ with R.

[2] By "error rate" here we mean the ratio of the error (in, say, ℓ_2 norm) to the dimension-normalized determinant $\det(\Lambda)^{1/n}$ of the lattice Λ, so exact scaling has no effect on the error rate.

[3] We note that decoding by "lifting" R to the larger-dimensional ring $\mathbb{Z}[X]/(X^m - 1)$, as done in [GHS12a], still leads to at least an $m/2$ factor loss in error tolerance overall, because some inherent loss is already incurred when replacing $R^\vee$ with R, and a bit more is lost in the lifting procedure.

Notation	Description	See
m, $n = \varphi(m)$, $\hat{m}$	The cyclotomic *index*, a positive integer having prime-power factorization $m = \prod_\ell m_\ell$, so that $n = \prod_\ell \varphi(m_\ell)$. Also, $\hat{m} = m/2$ if m is even, otherwise $\hat{m} = m$.	
$K = \mathbb{Q}(\zeta_m)$ $\cong \mathbb{Q}[X]/(\Phi_m(X))$ $\cong \bigotimes_\ell \mathbb{Q}(\zeta_{m_\ell})$	The mth *cyclotomic number field*, where ζ_m denotes an abstract element having order m over $\mathbb{Q}$. (Here $\Phi_m(X) \in \mathbb{Z}[X]$ is the mth cyclotomic polynomial, the minimal polynomial of ζ_m, which has degree n.) It is best viewed as the tensor product of the cyclotomic subfields $\mathbb{Q}(\zeta_{m_\ell})$.	§2.5.1
$\sigma\colon K \to \mathbb{C}^n$	The *canonical embedding* of K, which endows K with a geometry, e.g., $\|a\|_2 := \|\sigma(a)\|_2$ for $a \in K$. Both addition and multiplication in K correspond to their coordinate-wise counterparts in $\mathbb{C}^n$, yielding tight bounds on "expansion" under ring operations.	§2.5.2
$R = \mathbb{Z}[\zeta_m]$ $\cong \mathbb{Z}[X]/(\Phi_m(X))$ $\cong \bigotimes_\ell \mathbb{Z}[\zeta_{m_\ell}]$	The *ring of integers* of K. It is best viewed as a tensor product of subrings $R_\ell = \mathbb{Z}[\zeta_{m_\ell}]$.	§2.5.3
$R^\vee = \langle t^{-1} \rangle$, $g, t \in R$	The *dual* fractional ideal of R, generated by $t^{-1} = g/\hat{m}$, so $R \subseteq R^\vee \subseteq \hat{m}^{-1}R$. Each of $R^\vee$, g, and t can be seen as the tensor products of their counterparts in the subfields $\mathbb{Q}(\zeta_{m_\ell})$.	§2.5.4
$\vec{p} \subset R$	The "*powerful*" $\mathbb{Z}$-basis of R, defined as the tensor product of the power $\mathbb{Z}$-bases of each $\mathbb{Z}[\zeta_{m_\ell}]$. For non-prime-power m, it differs from the power $\mathbb{Z}$-basis $\{\zeta_m^0, \zeta_m^1, \ldots, \zeta_m^{n-1}\}$ often used to represent $\mathbb{Z}[\zeta_m]$, and has better computational and geometric properties.	§4
$\vec{c} \subset R_q$	The "*Chinese remainder*" (CRT) $\mathbb{Z}_q$-basis of $R_q = R/qR$, for any prime $q = 1 \bmod m$. It yields linear-time addition and multiplication in R_q, and there is an $O(n \log n)$-time algorithm for converting between $\vec{c}$ and $\vec{p}$ (as a $\mathbb{Z}_q$-basis of R_q).	§2.5.5, §5
$\vec{d} \subset R^\vee$	The "*decoding*" $\mathbb{Z}$-basis of $R^\vee$, defined as the dual of the (conjugate of the) powerful basis $\vec{p}$. It is used for optimal decoding of $R^\vee$ and its powers, and for efficiently sampling Gaussians.	§6

Figure 1: *Dramatis Personæ*.

1.3 Organization

We draw the reader's attention to Figure 1, which provides a glossary of the main algebraic objects and notation used in this work, and pointers to further discussion of their properties. The rest of the paper is organized as follows:

Section 2 Covers background on our (unusual, but useful) notation for vectors, matrices and tensors; Gaussian and subgaussian random variables; lattices and basic decoding/discretization algorithms; algebraic number theory; and ring-LWE. For the reader with some background in algebraic number theory, we draw attention to the lesser-known material in Section 2.5.1 on the tensorial decomposition into prime-power cyclotomics, and Section 2.5.4 on duality ($R^\vee$, dual bases, etc.).

Section 3 Recalls a "sparse decomposition" of the discrete Fourier transform (DFT) matrix, and develops a novel sparse decomposition for a closely related one that we call the "Chinese remainder transform," which plays a central role in many of our fast algorithms.

Section 4 Defines the "powerful" $\mathbb{Z}$-basis $\vec{p}$ of R and describes its algebraic and geometric properties.

Section 5 Defines the "Chinese remainder" $\mathbb{Z}_q$-basis $\vec{c}$ of R_q, gives its connection to the powerful basis, and describes how it enables fast ring operations.

Section 6 Defines the "decoding" basis $\vec{d}$ of $R^\vee$, gives its connection to the powerful basis, describes how it is used for decoding with optimal noise tolerance, and shows how to efficiently generate (continuous) Gaussians as represented in the decoding basis.

Section 7 Gives a regularity lemma for random lattices over arbitrary cyclotomics. This is needed for only one of our applications, as well as for adapting prior signature schemes and LWE-based (hierarchical) identity-based encryption schemes to the ring setting.

Section 8 Gives some applications of the toolkit: two basic public-key encryption schemes, and a "somewhat homomorphic" symmetric-key encryption scheme.

Acknowledgments. We thank Markus Püschel for his help with the sparse decomposition of the "Chinese remainder transform," and Damien Stehlé for useful discussions.

2 Preliminaries

For a positive integer k, we let $[k]$ denote the set $\{0, \ldots, k-1\}$. For any $\bar{a} \in \mathbb{R}/\mathbb{Z}$, we let $[\![\bar{a}]\!] \in \mathbb{R}$ denote the unique representative $a \in (\bar{a} + \mathbb{Z}) \cap [-1/2, 1/2)$. Similarly, for $\bar{a} \in \mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ we let $[\![\bar{a}]\!]$ denote the unique representative $a \in (\bar{a} + q\mathbb{Z}) \cap [-q/2, q/2)$. We extend $[\![\cdot]\!]$ entrywise to vectors and matrices. The *radical* of a positive integer m, denoted $\mathrm{rad}(m)$, is the product of all primes dividing m.

For a vector $\mathbf{x}$ over $\mathbb{R}$ or $\mathbb{C}$, define the ℓ_2 norm as $\|\mathbf{x}\|_2 = (\sum_i |x_i|^2)^{1/2}$, and the ℓ_∞ norm as $\|\mathbf{x}\|_\infty = \max_i |x_i|$. For an n-by-n matrix M we denote by $s_1(M)$ its largest singular value (also known as the spectral or operator norm), and by $s_n(M)$ its smallest singular value.

2.1 Vectors, Matrices, and Tensors

Throughout this paper, the entries of a vector over a domain D are always indexed (in no particular order) by some finite set S, and we write D^S to denote the set of all such vectors. When the domain is $\mathbb{Z}_q$ or a subset of the complex numbers, we usually denote vectors using bold lower-case letters (e.g., $\mathbf{a}$), otherwise we use arrow notation (e.g., $\vec{a}$). Similarly, the rows and columns of an "R-by-C matrix" over D are indexed by some finite sets R and C, respectively. We write $D^{R \times C}$ for the set of all such matrices, and typically use upper-case letters to denote individual matrices (e.g., A). The R-by-R identity matrix I_R has 1 as its (i,i)th entry for each $i \in R$, and 0 elsewhere. All the standard matrix and vector operations are defined in the natural way, for objects having compatible domains and index sets.

In particular, the Kronecker (or tensor) product $M = A \otimes B$ of an R_0-by-C_0 matrix A with an R_1-by-C_1 matrix B is the $(R_0 \times R_1)$-by-$(C_0 \times C_1)$ matrix M with entries $M_{(i_0,i_1),(j_0,j_1)} = A_{i_0,j_0} \cdot B_{i_1,j_1}$. The Kronecker product of two vectors, or of a matrix with a vector, is defined similarly. For positive integers n_0, n_1, we often implicitly identify the index set $[n_0] \times [n_1]$ with $[n_0 n_1]$, using the bijective correspondence

$(i_0, i_1) \leftrightarrow i = i_0 n_1 + i_1$; note that this matches the traditional Kronecker product for *ordered* rows and columns. Similarly, when $m = \prod_\ell m_\ell$ for a set of pairwise coprime positive integers m_ℓ, we often identify the index sets $\mathbb{Z}_m^*$ and $\prod_\ell \mathbb{Z}_{m_\ell}^*$ via the bijection induced by the Chinese remainder theorem. In other settings we reindex a set using another correspondence, which will be described in context.

An important fact about the Kronecker product is the *mixed-product property*: $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$. Using the mixed-product property, a tensor product $A = \bigotimes_\ell A_\ell$ of several matrices can be written as

$$A = \prod_\ell (I \otimes \cdots \otimes I \otimes A_\ell \otimes I \otimes \cdots \otimes I), \tag{2.1}$$

where the identity matrices have the appropriate induced index sets. In particular, if each A_ℓ is a square matrix of dimension n_ℓ, then A is square of dimension $n = \prod_\ell n_\ell$, and multiplication by A reduces to n/n_ℓ parallel multiplications by A_ℓ, in sequence for each value of ℓ (in any order).

2.2 The Space H

When working with cyclotomic number fields and ideal lattices under the canonical embedding (see Section 2.5.2 below), it is convenient to use a subspace $H \subseteq \mathbb{C}^{\mathbb{Z}_m^*}$ (for some integer $m \geq 2$), defined as

$$H = \{\mathbf{x} \in \mathbb{C}^{\mathbb{Z}_m^*} \ : \ x_i = \overline{x_{m-i}}, \ \forall \ i \in \mathbb{Z}_m^*\}.$$

Letting $n = \varphi(m)$, it is not difficult to verify that H (with the inner product induced on it by $\mathbb{C}^{\mathbb{Z}_m^*}$) is isomorphic to $\mathbb{R}^{[n]}$ as an inner product space. For $m = 2$ this is trivial, and for $m > 2$ this can be seen via the $\mathbb{Z}_m^*$-by-$[n]$ unitary basis matrix $B = \frac{1}{\sqrt{2}} \begin{pmatrix} I & \sqrt{-1}J \\ J & -\sqrt{-1}I \end{pmatrix}$ of H, where the $\mathbb{Z}_m^*$-indexed rows are shown in increasing order according to their representatives in $\{1, \ldots, m-1\}$, the $[n]$-indexed columns are shown in increasing order by index, I is the identity matrix, and J is the reversal matrix (obtained by reversing the columns of I).

We equip H with the ℓ_2 and ℓ_∞ norms induced on it from $\mathbb{C}^{\mathbb{Z}_m^*}$. Namely, for $\mathbf{x} \in H$ we have $\|\mathbf{x}\|_2 = \sum_i (|x_i|^2)^{1/2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$, and $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

Gram-Schmidt orthogonalization. For an ordered set $B = \{\mathbf{b}_j\}_{j \in [n]} \subset H$ of linearly independent vectors, the Gram-Schmidt orthogonalization $\widetilde{B} = \{\widetilde{\mathbf{b}}_j\}$ is defined iteratively as follows: $\widetilde{\mathbf{b}}_0 = \mathbf{b}_0$, and for $j = 1, 2, \ldots, n-1$, $\widetilde{\mathbf{b}}_j$ is the component of $\mathbf{b}_j$ orthogonal to the linear span of $\mathbf{b}_0, \ldots, \mathbf{b}_{j-1}$:

$$\widetilde{\mathbf{b}}_j = \mathbf{b}_j - \sum_{k \in [j]} \widetilde{\mathbf{b}}_k \cdot \langle \mathbf{b}_j, \widetilde{\mathbf{b}}_k \rangle / \langle \widetilde{\mathbf{b}}_k, \widetilde{\mathbf{b}}_k \rangle.$$

Viewing B as a matrix whose columns are the vectors $\mathbf{b}_j$, its orthogonalization corresponds to the unique factorization $B = QDU$, where Q is unitary with columns $\widetilde{\mathbf{b}}_j / \|\widetilde{\mathbf{b}}_j\|_2$; D is real diagonal with positive diagonal entries $\|\widetilde{\mathbf{b}}_j\|_2 > 0$; and U is real upper unitriangular with entries $w_{k,j} = \langle \mathbf{b}_j, \widetilde{\mathbf{b}}_k \rangle / \langle \widetilde{\mathbf{b}}_k, \widetilde{\mathbf{b}}_k \rangle$.[4] The Gram-Schmidt orthogonalization is $\widetilde{B} = QD$, and so $B = \widetilde{B}U$. The real positive definite Gram matrix of B is $B^*B = U^T D^2 U$. Because U is upper unitriangular, this is exactly the Cholesky decomposition of B^*B, which is unique; it therefore determines the matrices D, U in the Gram-Schmidt orthogonalization of B. One can also verify from the definitions that D^2 and U are both rational if the Gram matrix is rational.

[4] This is often referred to as the "QR" factorization, though here we have also factored out the diagonal entries of the upper-triangular matrix R into D, making U unitriangular.

2.3 Gaussians and Subgaussian Random Variables

For $s > 0$, define the Gaussian function $\rho_s \colon H \to (0, 1]$ as $\rho_s(\mathbf{x}) = \exp(-\pi\langle \mathbf{x}, \mathbf{x}\rangle/s^2) = \exp(-\pi\|\mathbf{x}\|_2^2/s^2)$. By normalizing this function we obtain the *continuous* Gaussian probability distribution D_s of parameter s, whose density is given by $s^{-n} \cdot \rho_s(\mathbf{x})$.

For much of our analysis it is convenient to use the standard notion of *subgaussian* random variables, relaxed slightly as in [MP12]. (For further details and full proofs, see, e.g., [Ver11].) For any $\delta \geq 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is δ-*subgaussian* with parameter $s > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies

$$\mathbb{E}[\exp(2\pi t X)] \leq \exp(\delta) \cdot \exp(\pi s^2 t^2).$$

Notice that the $\exp(\pi s^2 t^2)$ term on the right is exactly the (scaled) moment-generating function of the one-dimensional Gaussian distribution of parameter s over $\mathbb{R}$. It is easy to see that if X is δ-subgaussian with parameter s, then cX is δ-subgaussian with parameter $|c|s$ for any real c. In addition, by Markov's inequality, the tails of X are dominated by those of a Gaussian of parameter s, i.e., for all $t \geq 0$,

$$\Pr[|X| \geq t] \leq 2\exp(\delta - \pi t^2/s^2). \tag{2.2}$$

Using the inequality $\cosh(x) \leq \exp(x^2/2)$, it can be shown that any B-bounded centered random variable X (i.e., $\mathbb{E}[X] = 0$ and $|X| \leq B$ always) is 0-subgaussian with parameter $B\sqrt{2\pi}$.

The sum of independent subgaussian variables is easily seen to be subgaussian. Here we observe that the same holds even in a martingale-like setting.

Claim 2.1. *Let $\delta_i, s_i \geq 0$ and X_i be random variables for $i = 1, \ldots, k$. Suppose that for every i, when conditioning on any values of $X_1, \ldots, X_{i-1}$, the random variable X_i is δ_i-subgaussian with parameter s_i. Then $\sum X_i$ is $(\sum \delta_i)$-subgaussian with parameter $(\sum s_i^2)^{1/2}$.*

Proof. It suffices to prove the claim for $k = 2$; the general case follows by induction, since X_k is subgaussian conditioned on any value of $\sum_{i=1}^{k-1} X_i$. Indeed,

$$\mathbb{E}\big[\exp(2\pi t(X_1 + X_2))\big] = \mathbb{E}_{X_1}\Big[\exp(2\pi t X_1)\,\mathbb{E}_{X_2}\big[\exp(2\pi t X_2) \mid X_1\big]\Big] \leq \exp(\delta_1 + \delta_2)\exp(\pi(s_1^2 + s_2^2)t^2).$$

□

We also have the following bound on the tail of a sum of squares of independent subgaussian variables.

Lemma 2.2. *Let X be a δ-subgaussian random variable with parameter s. Then, for any $t \in (0, 1/(2s^2))$,*

$$\mathbb{E}\big[\exp(2\pi t X^2)\big] \leq 1 + 2\exp(\delta)\left(\frac{1}{2ts^2} - 1\right)^{-1}.$$

Moreover, if $X_1, \ldots, X_k$ are random variables, each of which is δ-subgaussian with parameter s conditioned on any values of the previous ones, then for any $r > k's^2/\pi$ where $k' = 2k\exp(\delta)$ we have that

$$\Pr\Big[\sum_i X_i^2 > r\Big] \leq \exp\Big(k'\Big(2\Big(\frac{\pi r}{k's^2}\Big)^{1/2} - \frac{\pi r}{k's^2} - 1\Big)\Big).$$

In particular, using the inequality $2\alpha^{1/2} - \alpha - 1 \leq -\alpha/4$ valid for all $\alpha \geq 4$, we obtain that for any $r \geq 4k's^2/\pi$,

$$\Pr\Big[\sum_i X_i^2 > r\Big] \leq \exp\Big(-\frac{\pi r}{4s^2}\Big).$$

Proof. Using integration by parts and (2.2),

$$\begin{aligned}\mathbb{E}\big[\exp(2\pi tX^2)\big] &= 1 + \int_0^\infty \Pr[|X| \geq r] \cdot 4\pi tr \exp(2\pi tr^2)\mathrm{d}r \\ &\leq 1 + 8\pi t \exp(\delta) \int_0^\infty r \exp(-\pi r^2/s^2 + 2\pi tr^2)\mathrm{d}r \\ &= 1 + 2\exp(\delta)\Big(\frac{1}{2ts^2} - 1\Big)^{-1} \\ &\leq \exp\Big(2\exp(\delta)\Big(\frac{1}{2ts^2} - 1\Big)^{-1}\Big),\end{aligned}$$

where the last equality uses that for every $a > 0$, $\int_0^\infty r\exp(-ar^2)\mathrm{d}r = (2a)^{-1}$. This completes the first part of the lemma. For the second part, notice that by the above, if $X_1, \ldots, X_k$ are as in the statement, we have for any $t \in (0, 1/(2s^2))$,

$$\mathbb{E}\Big[\exp(2\pi t\sum_i X_i^2)\Big] \leq \exp\Big(2k\exp(\delta)\Big(\frac{1}{2ts^2} - 1\Big)^{-1}\Big),$$

and hence by Markov's inequality, for all $r > 0$ and $t \in (0, 1/(2s^2))$,

$$\Pr\Big[\sum_i X_i^2 > r\Big] \leq \exp\Big(2k\exp(\delta)\Big(\frac{1}{2ts^2} - 1\Big)^{-1} - 2\pi tr\Big).$$

Letting $x = 2s^2t \in (0,1)$ and $A = \pi r/(s^2k') > 1$, the expression inside the exponent is

$$2k\exp(\delta)\Big(\Big(\frac{1}{x} - 1\Big)^{-1} - Ax\Big).$$

The lemma follows using the fact that for any $A > 1$, the minimum over $x \in (0,1)$ of the expression inside the parenthesis is $2\sqrt{A} - A - 1$ (obtained at $1 - 1/\sqrt{A}$). □

We extend the notion of subgaussianity to random vectors in $\mathbb{R}^n$ (or equivalently, in H). Specifically, we say that a random vector X in $\mathbb{R}^n$ is δ-subgaussian with parameter s if for all unit vectors $\mathbf{u} \in \mathbb{R}^n$, the random variable $\langle X, \mathbf{u}\rangle$ is δ-subgaussian with parameter s. It follows from Claim 2.1 that if the coordinates of a random vector in $\mathbb{R}^n$ are independent, and each is δ-subgaussian with parameter s, then the random vector is $n\delta$-subgaussian with the same parameter s.

Sums of subgaussian random vectors are again easily seen to be subgaussian, even in the martingale setting as in Claim 2.1 above. We summarize this in the following corollary, which considers the more general setting in which we apply a (possibly different) linear transformation to each subgaussian random vector.

Corollary 2.3. *Let $\delta_i, s_i \geq 0$ and X_i be random vectors in $\mathbb{R}^n$ (or in H), and let A_i be $n \times n$ matrices for $i = 1, \ldots, k$. Suppose that for every i, when conditioning on any values of $X_1, \ldots, X_{i-1}$, the random vector X_i is δ_i-subgaussian with parameter s_i. Then $\sum A_iX_i$ is $(\sum \delta_i)$-subgaussian with parameter $\lambda_{max}(\sum s_i^2 A_i A_i^T)^{1/2}$, where λ_{max} denotes the largest eigenvalue.*

Proof. For any vector $\mathbf{u} \in \mathbb{R}^n$,

$$\Big\langle \sum_i A_iX_i, \mathbf{u}\Big\rangle = \sum_i \langle A_iX_i, \mathbf{u}\rangle = \sum_i \langle X_i, A_i^T\mathbf{u}\rangle,$$

which is a sum of random variables satisfying that for each i, the ith variable is δ_i-subgaussian with parameter $s_i\|A_i^T\mathbf{u}\|_2$ conditioned on any value of the previous ones. By Claim 2.1, this sum is $(\sum \delta_i)$-subgaussian with parameter

$$\Big(\sum_i s_i^2\|A_i^T\mathbf{u}\|_2^2\Big)^{1/2} = \Big(\mathbf{u}^T\big(\sum_i s_i^2 A_i A_i^T\big)\mathbf{u}\Big)^{1/2},$$

whose maximum over all unit vectors $\mathbf{u}$ is $\lambda_{\max}(\sum_i s_i^2 A_i A_i^T)^{1/2}$. □

By applying Corollary 2.3 with the linear transformation induced by coordinate-wise multiplication in $H \subset \mathbb{C}^{\mathbb{Z}_m^*}$ we obtain the following.

Claim 2.4. *If X is a δ-subgaussian with parameter s in H, and $\mathbf{z} \in H$ is any element, then the coordinate-wise multiplication $\mathbf{z} \odot X \in H$ is δ-subgaussian with parameter $\|\mathbf{z}\|_\infty \cdot s$. More generally, if $X_j \in H$ are random vectors satisfying the property in Corollary 2.3 for some $\delta_j, s_j \geq 0$ (respectively), then for any $\mathbf{z}_j \in H$, we have that $\sum_j \mathbf{z}_j \odot X_j \in H$ is $(\sum \delta_j)$-subgaussian with parameter $\max_{i\in\mathbb{Z}_m^*}(\sum_j s_j^2|(\mathbf{z}_j)_i|^2)^{1/2}$.*

2.4 Lattice Background

We define a *lattice* as a discrete additive subgroup of H. We deal here exclusively with full-rank lattices, which are generated as the set of all integer linear combinations of some set of n linearly independent *basis* vectors $B = \{\mathbf{b}_j\} \subset H$:

$$\Lambda = \mathcal{L}(B) = \Big\{\sum\nolimits_j z_j\mathbf{b}_j \ : \ z_j \in \mathbb{Z}\Big\}.$$

Two bases B, B' generate the same lattice if and only if there exists a unimodular matrix U (i.e., integer matrix with determinant ± 1) such that $BU = B'$. The *determinant* of a lattice $\mathcal{L}(B)$ is defined as $|\det(B)|$, which is independent of the choice of basis B. The *minimum distance* $\lambda_1(\Lambda)$ of a lattice Λ (in the Euclidean norm) is the length of a shortest nonzero lattice vector: $\lambda_1(\Lambda) = \min_{\mathbf{0}\neq\mathbf{x}\in\Lambda}\|\mathbf{x}\|_2$.

The *dual lattice* of $\Lambda \subset H$ is defined as $\Lambda^\vee = \{\mathbf{y} \in H : \forall\, \mathbf{x} \in \Lambda,\ \langle\mathbf{x}, \overline{\mathbf{y}}\rangle = \sum_i x_i y_i \in \mathbb{Z}\}$. Notice that this is actually the *complex conjugate* of the dual lattice as usually defined in $\mathbb{C}^n$; our definition corresponds more naturally to the notion of duality in algebraic number theory (see Section 2.5.4). All of the properties of the dual lattice that we use also hold for the conjugate dual. In particular, $\det(\Lambda^\vee)$ is $\det(\Lambda)^{-1}$.

It is easy to see that $(\Lambda^\vee)^\vee = \Lambda$. If $B = \{\mathbf{b}_j\} \subset H$ is a set of linearly independent vectors (i.e., an $\mathbb{R}$-basis of H), its *dual basis* $D = \{\mathbf{d}_j\}$ is characterized by $\langle\mathbf{b}_j, \overline{\mathbf{d}_k}\rangle = \delta_{jk}$, where δ_{jk} is the Kronecker delta. It is easy to verify that $\mathcal{L}(D) = \mathcal{L}(B)^\vee$.

Micciancio and Regev [MR04] introduced a lattice quantity called the *smoothing parameter*, and related it to various lattice quantities.

Definition 2.5. *For a lattice Λ and positive real $\varepsilon > 0$, the* smoothing parameter $\eta_\varepsilon(\Lambda)$ *is the smallest s such that $\rho_{1/s}(\Lambda^\vee\backslash\{\mathbf{0}\}) \leq \varepsilon$.*

Lemma 2.6 ([MR04, Lemma 3.2]). *For any n-dimensional lattice Λ, we have $\eta_{2^{-2n}}(\Lambda) \leq \sqrt{n}/\lambda_1(\Lambda^\vee)$.*[5]

Lemma 2.7 ([Reg05, Claim 3.8]). *For any lattice Λ, real $\varepsilon > 0$ and $s \geq \eta_\varepsilon(\Lambda)$, and $\mathbf{c} \in H$, we have $\rho_s(\Lambda + \mathbf{c}) \in [1 \pm \varepsilon] \cdot s^n \det(\Lambda)^{-1}$.*

[5] Note that we are using $\varepsilon = 2^{-2n}$ instead of 2^{-n} as in [MR04], but the stronger bound holds by the same proof.

For a lattice coset $\Lambda + \mathbf{c}$ and real $s > 0$, define the *discrete Gaussian* probability distribution over $\Lambda + \mathbf{c}$ with parameter s as

$$D_{\Lambda+\mathbf{c},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\Lambda + \mathbf{c})} \quad \forall\, \mathbf{x} \in \Lambda + \mathbf{c}. \tag{2.3}$$

It is known to satisfy the following concentration bound.

Lemma 2.8 ([Ban93, Lemma 1.5(i)]). *For any n-dimensional lattice Λ and $s > 0$, a point sampled from $D_{\Lambda,s}$ has Euclidean norm at most $s\sqrt{n}$, except with probability at most 2^{-2n}.*

Gentry, Peikert, and Vaikuntanathan [GPV08] showed how to efficiently sample from a discrete Gaussian, using any lattice basis consisting of sufficiently short orthogonalized vectors.

Lemma 2.9 ([GPV08, Theorem 4.1]). *There is an efficient algorithm that samples to within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda+\mathbf{c},s}$, given $\mathbf{c} \in H$, a basis B of Λ, and a parameter $s \geq \max_j \|\widetilde{\mathbf{b}}_j\| \cdot \omega(\sqrt{\log n})$, where $\widetilde{B} = \{\widetilde{\mathbf{b}}_j\}$ is the Gram-Schmidt orthogonalization of B.*

We make a few remarks on the implementation of the algorithm from Lemma 2.9. It is a randomized variant of Babai's "nearest plane" algorithm [Bab85] (a related variant was also considered by Klein [Kle00] for a different problem). On input $\mathbf{c} \in H$, and a basis B and parameter s satisfying the above constraint, it does the following: for $j = n-1, \ldots, 0$, let $\mathbf{c} \leftarrow \mathbf{c} - z_j \mathbf{b}_j$, where $z_j \leftarrow c'_j + D_{\mathbb{Z}-c'_j,s_j}$ for $c'_j = \langle \mathbf{c}, \widetilde{\mathbf{b}}_j \rangle / \langle \widetilde{\mathbf{b}}_j, \widetilde{\mathbf{b}}_j \rangle$ and $s_j = s/\|\widetilde{\mathbf{b}}_j\|_2$. Output the final value of $\mathbf{c}$.

In practice, the above algorithm is usually invoked on a fixed basis B whose Gram matrix B^*B is rational. It is best implemented by precomputing the rational matrices D^2, U associated with $\widetilde{B}$ and B^*B (see Section 2.2), and by representing the input and intermediate values $\mathbf{c}$ using rational coefficient vectors with respect to B. Then each value $c'_j = \langle \mathbf{c}, \widetilde{\mathbf{b}}_j \rangle / \langle \widetilde{\mathbf{b}}_j, \widetilde{\mathbf{b}}_j \rangle$ can be computed simply as the inner product of $\mathbf{c}$'s coefficient vector with the jth row of U.

2.4.1 Decoding

In many applications we need to perform the following algorithmic task, which is essentially a bounded-distance decoding. Let Λ be a known fixed lattice, and let $\mathbf{x} \in H$ be an unknown short vector. The goal is to recover $\mathbf{x}$, given $\mathbf{t} = \mathbf{x} \bmod \Lambda$. Although there are several possible algorithms for this task, here we focus on a slight extension of the so-called "round-off" algorithm originally due to Babai [Bab85]. This is due to its high efficiency and because for our purposes it performs optimally (or nearly so). The algorithm is very simple: let $\{\mathbf{v}_i\}$ be a fixed set of n linearly independent (and typically short) vectors in the dual lattice $\Lambda^\vee$. Denote the dual basis of $\{\mathbf{v}_i\}$ by $\{\mathbf{b}_i\}$, and let $\Lambda' \supseteq \Lambda$ be the superlattice generated by $\{\mathbf{b}_i\}$. Given an input $\mathbf{t} = \mathbf{x} \bmod \Lambda$, we express $\mathbf{t} \bmod \Lambda'$ in the basis $\{\mathbf{b}_i\}$ as $\sum_i c_i \mathbf{b}_i$, where $c_i \in \mathbb{R}/\mathbb{Z}$ (so $c_i = \langle \mathbf{x}, \overline{\mathbf{v}_i} \rangle \bmod 1$), and output $\sum_i \lfloor\!\lceil c_i \rfloor\!\rceil \mathbf{b}_i \in H$.

Claim 2.10. *Let $\Lambda \subset H$ be a lattice, let $\{\mathbf{v}_i\} \subset \Lambda^\vee$ be a set of n linearly independent vectors in its dual, and let $\{\mathbf{b}_i\} \subset \Lambda$ denote the dual basis of $\{\mathbf{v}_i\}$. The above round-off algorithm, given input $\mathbf{x} \bmod \Lambda$, outputs $\mathbf{x}$ if and only if all the coefficients $a_i = \langle \mathbf{x}, \overline{\mathbf{v}_i} \rangle \in \mathbb{R}$ in the expansion $\mathbf{x} = \sum_i a_i \mathbf{b}_i$ are in $[-1/2, 1/2)$.*

We remark that in Babai's round-off algorithm one often assumes that $\{\mathbf{v}_i\}$ is a *basis* of $\Lambda^\vee$ (and hence $\{\mathbf{b}_i\}$ is a basis of Λ), whereas here we consider the more general case where $\{\mathbf{v}_i\}$ can be an arbitrary set of linearly independent vectors in $\Lambda^\vee$. For some lattices (including those appearing in our applications) this can make a big difference. Consider for instance the lattice of all points in $\mathbb{Z}^n$ whose coordinates sum to an even

number. The dual of this lattice is $\mathbb{Z}^n \cup (\mathbb{Z}^n + (1, \ldots, 1)/2)$, and clearly any basis of this dual must contain a vector of length at least $\sqrt{n}/2$. As a result, when limited to using a basis, the round-off algorithm can fail for vectors of length greater than $1/\sqrt{n}$. However, the dual lattice clearly has a set of n linearly independent vectors of length 1, allowing us to decode up to length $1/2$.

2.4.2 Discretization

We now consider another algorithmic task related to the one in the previous subsection. This task shows up in applications, such as when converting a continuous Gaussian into a discrete Gaussian-like distribution. Given a lattice $\Lambda = \mathcal{L}(B)$ represented by a "good" basis $B = \{\mathbf{b}_i\}$, a point $\mathbf{x} \in H$, and a point $\mathbf{c} \in H$ representing a lattice coset $\Lambda + \mathbf{c}$, the goal is to discretize $\mathbf{x}$ to a point $\mathbf{y} \in \Lambda + \mathbf{c}$, written $\mathbf{y} \leftarrow \lfloor \mathbf{x} \rceil_{\Lambda+\mathbf{c}}$, so that the length (or subgaussian parameter) of $\mathbf{y} - \mathbf{x}$ is not too large. To do this, we sample a relatively short offset vector $\mathbf{f}$ from the coset $\Lambda + \mathbf{c}' = \Lambda + (\mathbf{c} - \mathbf{x})$ in one of a few natural ways described below, and output $\mathbf{y} = \mathbf{x} + \mathbf{f}$. We require that the method used to choose $\mathbf{f}$ be efficient and depend only on the desired coset $\Lambda + \mathbf{c}'$, not on the particular representative used to specify it; we call such a procedure (or the induced discretization) *valid*.

Note that for a valid discretization, $\lfloor \mathbf{z} + \mathbf{x} \rceil_{\Lambda+\mathbf{c}}$ and $\mathbf{z} + \lfloor \mathbf{x} \rceil_{\Lambda+\mathbf{c}}$ are identically distributed for any $\mathbf{z} \in \Lambda$. Therefore, for any sublattice $\Lambda' \subseteq \Lambda$, a valid discretization also induces a well-defined discretization from any *coset* $\bar{\mathbf{x}} = \Lambda' + \mathbf{x}$ to $\bar{\mathbf{y}} = \bar{\mathbf{x}} + \mathbf{f} = \Lambda' + \mathbf{y}$, where $\mathbf{y} \in \Lambda + \mathbf{c}$.

There are several valid ways of sampling $\mathbf{f}$, offering tradeoffs between efficiency and output guarantees:

- A particularly simple and efficient method is "coordinate-wise randomized rounding:" given a coset $\Lambda + \mathbf{c}'$, we represent $\mathbf{c}'$ in the basis B as $\mathbf{c}' = \sum_i a_i \mathbf{b}_i \bmod \Lambda$ for some coefficients $a_i \in [0, 1)$, then randomly and independently choose each f_i from $\{a_i - 1, a_i\}$ to have expectation zero, and output $\mathbf{f} = \sum_i f_i \mathbf{b}_i \in \Lambda + \mathbf{c}'$. The validity of this procedure is immediate, since any representative of $\Lambda + \mathbf{c}'$ induces the same a_i values. Because each f_i has expectation zero and is bounded by 1 in magnitude, it is 0-subgaussian with parameter $\sqrt{2\pi}$ (see Section 2.3), and hence so is the entire vector of f_i values. By Corollary 2.3 (applied with just one random vector), we conclude that $\mathbf{f}$ is 0-subgaussian with parameter $\sqrt{2\pi} \cdot s_1(B)$.

- In some settings we can use a *deterministic* version of the above method, where we instead compute coefficients $a_i \in [-1/2, 1/2)$ and simply output $\mathbf{f} = \sum_i a_i \mathbf{b}_i$. When, for example, $\mathbf{x}$ comes from a sufficiently wide continuous Gaussian, this method yields $\mathbf{y} = \mathbf{x} + \mathbf{f}$ having a (very slightly) better subgaussian parameter than the randomized method. However, the analysis is a bit more involved, and we omit it.

- If $\mathbf{x}$ has a continuous or discrete Gaussian distribution, then using more sophisticated rounding methods it is possible to make $\mathbf{y}$ also be distributed according to a true discrete Gaussian (of some particular covariance), which is needed in some applications (though not any we develop in this paper). By [Pei10, Theorem 3.1], under mild conditions it suffices for $\mathbf{f}$ to be distributed as a discrete Gaussian over $\Lambda + \mathbf{c}'$, and the covariance parameter of $\mathbf{y}$ will be the sum of those of $\mathbf{x}$ and $\mathbf{f}$. Using the algorithm from Lemma 2.9, we can sample a discrete Gaussian $\mathbf{f}$ with parameter bounded by $\max_j \|\widetilde{\mathbf{b}}_j\| \cdot \omega(\sqrt{\log n})$. Alternatively, a simpler and more efficient randomized round-off algorithm obtains a parameter bounded by $s_1(B) \cdot \omega(\sqrt{\log n})$ [Pei10]. Both of these methods are easily seen to be valid, though note that they yield slightly worse Gaussian parameters than the two simpler methods described above.

2.5 Algebraic Number Theory Background

Algebraic number theory is the study of *number fields*. Here we review the necessary background, specialized to the case of *cyclotomic* number fields, which are the only kind we use in this work. More background and complete proofs can be found in any introductory book on the subject, e.g., [Ste04, Lan94], and especially the latter reference for material related to the tensorial decomposition.

2.5.1 Cyclotomic Number Fields and Polynomials

For a positive integer m, the mth *cyclotomic number field* is a field extension $K = \mathbb{Q}(\zeta_m)$ obtained by adjoining an element ζ_m of order m (i.e., a primitive mth root of unity) to the rationals. (Note that we view ζ_m as an abstract element, and not, for example, as any particular value in $\mathbb{C}$.) The minimal polynomial of ζ_m is the mth *cyclotomic polynomial*

$$\Phi_m(X) = \prod_{i \in \mathbb{Z}_m^*} (X - \omega_m^i) \in \mathbb{Z}[X], \tag{2.4}$$

where $\omega_m \in \mathbb{C}$ is any primitive mth root of unity in $\mathbb{C}$, e.g., $\omega_m = \exp(2\pi\sqrt{-1}/m)$. Therefore, there is a natural isomorphism between K and $\mathbb{Q}[X]/(\Phi_m(X))$, given by $\zeta_m \mapsto X$. Since $\Phi_m(X)$ has degree $n = |\mathbb{Z}_m^*| = \varphi(m)$, we can view K as a vector space of degree n over $\mathbb{Q}$, which has $(\zeta_m^j)_{j \in [n]} = (1, \zeta_m, \ldots, \zeta_m^{n-1}) \in K^{[n]}$ as a basis. This is called the *power basis* of K.

We recall two useful facts about cyclotomic polynomials, which can be verified by examining the roots of both sides of each equation.

Fact 2.11. *For any m, we have $X^m - 1 = \prod_{d|m} \Phi_d(X)$, where d runs over all the positive divisors of m. In particular, $\Phi_p(X) = 1 + X + X^2 + \cdots + X^{p-1}$ for any prime p.*

Fact 2.12. *For any m, we have $\Phi_m(X) = \Phi_{\mathrm{rad}(m)}(X^{m/\mathrm{rad}(m)})$, where recall that $\mathrm{rad}(m)$ is the product of all distinct primes dividing m. In particular, if m is a power of a prime p, then $\Phi_m(X) = \Phi_p(X^{m/p})$.*

For instance, $\Phi_8(X) = 1 + X^4$ and $\Phi_{25}(X) = 1 + X^5 + X^{10} + X^{15} + X^{20}$.

For any m' dividing m, it is often convenient to view $K' = \mathbb{Q}(\zeta_{m'})$ as a subfield of $K = \mathbb{Q}(\zeta_m)$, by identifying $\zeta_{m'}$ with $\zeta_m^{m/m'}$.

Non-prime-power cyclotomics. Not all cyclotomic polynomials are "regular"-looking or have 0-1 (or even small) coefficients. Generally speaking, the irregularity and range of coefficients grows with the number of prime divisors of m. For example, $\Phi_6(X) = X^2 - X + 1$; $\Phi_{3\cdot5\cdot7}(X)$ has 33 monomials with coefficients $-2, -1$, and 1; and $\Phi_{3\cdot5\cdot7\cdot11\cdot13}(X)$ has coefficients of magnitude up to 22. Fortunately, the form of $\Phi_m(X)$ for non-prime-power m will never be a concern in this work, due to an alternative way of viewing $K = \mathbb{Q}(\zeta_m)$ by reducing to the case of prime-power cyclotomics.

To do this we first need to briefly recall the notion of a *tensor product* of fields. Let K, L be two field extensions of $\mathbb{Q}$. Then the field tensor product $K \otimes L$ is defined as the set of all $\mathbb{Q}$-linear combinations of *pure tensors* $a \otimes b$ for $a \in K, b \in L$, where $\otimes$ is $\mathbb{Q}$-bilinear and satisfies the mixed-product property, i.e.,

$$\begin{aligned}
(a_1 \otimes b) + (a_2 \otimes b) &= (a_1 + a_2) \otimes b \\
(a \otimes b_1) + (a \otimes b_2) &= a \otimes (b_1 + b_2) \\
e(a \otimes b) &= (ea) \otimes b = a \otimes (eb) \\
(a_1 \otimes b_1)(a_2 \otimes b_2) &= (a_1 a_2) \otimes (b_1 b_2)
\end{aligned}$$

for all $e \in \mathbb{Q}$. These properties define addition and multiplication in $K \otimes L$, and though the result is not always a field (because it may lack multiplicative inverses), it will always be one whenever we take the tensor product of two cyclotomic fields in this work. It is straightforward to verify that if A, B are $\mathbb{Q}$-bases of K, L respectively, then the Kronecker product $A \otimes B$ is a $\mathbb{Q}$-basis of $K \otimes L$. Later on we also consider tensor products of rings, or more generally of $\mathbb{Z}$-modules. These are defined in the same way, except that they are made up of only the $\mathbb{Z}$-linear combinations of pure tensors. This always yields a ring or $\mathbb{Z}$-module, respectively, with $\mathbb{Z}$-bases obtained by tensoring $\mathbb{Z}$-bases of the original objects.

A key fact from algebraic number theory is the following.

Proposition 2.13. *Let m have prime-power factorization $m = \prod_\ell m_\ell$, i.e., the m_ℓ are powers of distinct primes. Then $K = \mathbb{Q}(\zeta_m)$ is isomorphic to the tensor product $\bigotimes_\ell K_\ell$ of the fields $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$, via the correspondence $\prod_\ell a_\ell \leftrightarrow (\otimes_\ell a_\ell)$, where on the left we implicitly embed each $a_\ell \in K_\ell$ into K.*

2.5.2 Embeddings and Geometry

Here we describe the *embeddings* of a cyclotomic number field, which induce a 'canonical' geometry on it.

The mth cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ of degree $n = \varphi(m)$ has exactly n ring homomorphisms (embeddings) $\sigma_i \colon K \to \mathbb{C}$ that fix every element of $\mathbb{Q}$. Concretely, for each $i \in \mathbb{Z}_m^*$ there is an embedding σ_i defined by $\sigma_i(\zeta_m) = \omega_m^i$, where $\omega_m \in \mathbb{C}$ is some fixed primitive mth root of unity. Clearly, the embeddings come in pairs of complex conjugates, i.e., $\sigma_i = \overline{\sigma_{m-i}}$. The *canonical embedding* $\sigma \colon K \to \mathbb{C}^{\mathbb{Z}_m^*}$ is defined as

$$\sigma(a) = (\sigma_i(a))_{i \in \mathbb{Z}_m^*}.$$

Due to the conjugate pairs, σ actually maps into $H \subset \mathbb{C}^{\mathbb{Z}_m^*}$, defined in Section 2.2. Note that σ is a ring homomorphism from K to H, where multiplication and addition in H are both component-wise.

By identifying K with its canonical embedding into H, we endow K with a canonical geometry. Recalling that norms on H are just those induced from $\mathbb{C}^{\mathbb{Z}_m^*}$, we see that for any $a \in K$, the ℓ_2 norm of a is simply $\|a\|_2 = \|\sigma(a)\|_2 = (\sum_i |\sigma_i(a)|^2)^{1/2}$, and the ℓ_∞ norm is $\max_i |\sigma_i(a)|$. Because multiplication of embedded elements is component-wise, for any $a, b \in K$ we have

$$\|a \cdot b\| \leq \|a\|_\infty \cdot \|b\|, \tag{2.5}$$

where $\|\cdot\|$ denotes either the ℓ_2 or ℓ_∞ norm (or indeed, any ℓ_p norm). Thus the ℓ_∞ norm acts as an "absolute value" for K that bounds how much an element expands any other by multiplication. For example, note that for any power ζ of ζ_m, each $\sigma_i(\zeta)$ must be a root of unity in $\mathbb{C}$, and hence $\|\zeta\|_2 = \sqrt{n}$ and $\|\zeta\|_\infty = 1$.

The *trace* $\mathrm{Tr} = \mathrm{Tr}_{K/\mathbb{Q}} \colon K \to \mathbb{Q}$ can be defined as the sum of the embeddings: $\mathrm{Tr}(a) = \sum_i \sigma_i(a)$. Clearly, the trace is $\mathbb{Q}$-linear: $\mathrm{Tr}(a + b) = \mathrm{Tr}(a) + \mathrm{Tr}(b)$ and $\mathrm{Tr}(c \cdot a) = c \cdot \mathrm{Tr}(a)$ for all $a, b \in K$ and $c \in \mathbb{Q}$. Also notice that

$$\mathrm{Tr}(a \cdot b) = \sum_i \sigma_i(a)\sigma_i(b) = \langle \sigma(a), \overline{\sigma(b)} \rangle,$$

so $\mathrm{Tr}(a \cdot b)$ is a symmetric bilinear form akin to the inner product of the embeddings of a and b. The (field) *norm* $\mathrm{N} = \mathrm{N}_{K/\mathbb{Q}} \colon K \to \mathbb{Q}$ can be defined as the product of all the embeddings: $\mathrm{N}(a) = \prod_i \sigma_i(a)$. Clearly, the norm is multiplicative: $\mathrm{N}(a \cdot b) = \mathrm{N}(a) \cdot \mathrm{N}(b)$.

When taking $K \cong \bigotimes_\ell K_\ell$ as in Proposition 2.13, it follows directly from the definitions that σ is the tensor product of the canonical embeddings $\sigma^{(\ell)}$ of K_ℓ, i.e.,

$$\sigma(\otimes_\ell a_\ell) = \bigotimes_\ell \sigma^{(\ell)}(a_\ell). \tag{2.6}$$

(Here the index set of σ is $\prod_\ell \mathbb{Z}_{m_\ell}^*$, which corresponds bijectively to $\mathbb{Z}_m^*$ via the Chinese remainder theorem.) This decomposition of σ in turn implies that the trace decomposes as

$$\mathrm{Tr}_{K/\mathbb{Q}}(\otimes_\ell a_\ell) = \prod\nolimits_\ell \mathrm{Tr}_{K_\ell/\mathbb{Q}}(a_\ell). \tag{2.7}$$

Using the canonical embedding also allows us to think of the Gaussian distribution D_r over H as a distribution over K, or more accurately, over the field tensor product $K_\mathbb{R} = K \otimes \mathbb{R}$, which is isomorphic as a real vector space to H via σ. For our purposes it is usually helpful to ignore the distinction between K and $K_\mathbb{R}$, and to approximate the latter by the former using sufficient precision.

2.5.3 Ring of Integers and Its Ideals

Let $R \subset K$ denote the set of all algebraic integers in a number field K. This set forms a ring (under the usual addition and multiplication operations in K), called the *ring of integers* of K. Note that the trace and norm of an algebraic integer are rational integers (i.e., in $\mathbb{Z}$), so we have the induced functions $\mathrm{Tr}, \mathrm{N}\colon R \to \mathbb{Z}$.

For the mth cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ of degree $n = \varphi(m)$, the ring of integers happens to be $R = \mathbb{Z}[\zeta_m] \cong \mathbb{Z}[X]/\Phi_m(X)$, and hence has the power basis $\{\zeta_m^j\}_{j\in[n]}$ as a $\mathbb{Z}$-basis. Alternatively—and this is the view we adopt throughout the paper—we can view $R \cong \bigotimes_\ell R_\ell$ as a tensor product of the rings of integers R_ℓ in $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$, where $m = \prod_\ell m_\ell$ is the prime-power factorization of m.

The (absolute) *discriminant* Δ_K of K is a measure of the geometric sparsity of its ring of integers, defined as $\Delta_K = \det(\sigma(R))^2$, the squared determinant of the lattice $\sigma(R)$.[6] The discriminant of the mth cyclotomic number field is

$$\Delta_K = \left(\frac{m}{\prod\limits_{\text{prime } p \mid m} p^{1/(p-1)}} \right)^n \leq n^n, \tag{2.8}$$

where the product in the denominator runs over all primes p dividing m. The above inequality is tight exactly when m is a power of two.

An *(integral) ideal* $\mathcal{I} \subseteq R$ is a nontrivial (i.e., $I \neq \emptyset$ and $\mathcal{I} \neq \{0\}$) additive subgroup that is closed under multiplication by R, i.e., $r \cdot a \in \mathcal{I}$ for any $r \in R$ and $a \in \mathcal{I}$.[7] A *principal* ideal $\mathcal{I}$ is one that is generated by a single element, i.e., $\mathcal{I} = uR$ for some $u \in R$ which is unique up to multiplication by units in R; we sometimes write $\mathcal{I} = \langle u \rangle$. An ideal $\mathcal{I}$ always has a $\mathbb{Z}$-basis of cardinality n, which is not unique; if $\mathcal{I} = \langle u \rangle$ and B is any $\mathbb{Z}$-basis of R, then uB is a $\mathbb{Z}$-basis of $\mathcal{I}$. A *fractional ideal* $\mathcal{I} \subset K$ is a set such that $d\mathcal{I} \subseteq R$ is an integral ideal for some $d \in R$, and is principal if it equals uR for some $u \in K$. Any fractional ideal $\mathcal{I}$ embeds under σ as a lattice $\sigma(\mathcal{I})$ in H, which we call an *ideal lattice*. We identify $\mathcal{I}$ with this lattice and associate with $\mathcal{I}$ all the usual lattice quantities (determinant, minimum distance, etc.).

The norm of an ideal $\mathcal{I}$ is its index as an additive subgroup of R, i.e., $\mathrm{N}(\mathcal{I}) = |R/\mathcal{I}|$. This notion of norm generalizes the field norm, in that $\mathrm{N}(\langle a \rangle) = |\mathrm{N}(a)|$ for any $a \in R$, and $\mathrm{N}(\mathcal{I}\mathcal{J}) = \mathrm{N}(\mathcal{I})\,\mathrm{N}(\mathcal{J})$. The norm of a fractional ideal $\mathcal{I}$ is defined as $\mathrm{N}(\mathcal{I}) = \mathrm{N}(d\mathcal{I})/|\mathrm{N}(d)|$, where $d \in R$ is such that $d\mathcal{I} \subseteq R$. It follows that the determinant of an ideal lattice $\mathcal{I}$ is

$$\det(\sigma(\mathcal{I})) = \mathrm{N}(\mathcal{I}) \cdot \sqrt{\Delta_K}. \tag{2.9}$$

The following lemma gives upper and lower bounds on the minimum distance of an ideal lattice. The upper bound is an immediate consequence of Minkowski's first theorem; the lower bound follows from the arithmetic mean/geometric mean inequality, and the fact that $|\mathrm{N}(a)| \geq \mathrm{N}(\mathcal{I})$ for any nonzero $a \in \mathcal{I}$.

[6] Some texts define the discriminant as a signed quantity, but in this work we only care about its magnitude.

[7] Some texts also define the trivial set $\{0\}$ as an ideal, but in this work it is more convenient to exclude it.

Lemma 2.14. *For any fractional ideal $\mathcal{I}$ in a number field K of degree n,*

$$\sqrt{n} \cdot \mathrm{N}^{1/n}(\mathcal{I}) \quad \leq \quad \lambda_1(\mathcal{I}) \quad \leq \quad \sqrt{n} \cdot \mathrm{N}^{1/n}(\mathcal{I}) \cdot \sqrt{\Delta_K^{1/n}}.$$

The sum $\mathcal{I} + \mathcal{J}$ of two ideals is the set of all $a + b$ for $a \in \mathcal{I}$, $b \in \mathcal{J}$, and the product ideal $\mathcal{I}\mathcal{J}$ is the set of all finite sums of terms ab for $a \in \mathcal{I}$, $b \in \mathcal{J}$. Multiplication extends to fractional ideals in the obvious way, and the set of fractional ideals forms a group under multiplication; in particular, every fractional ideal $\mathcal{I}$ has a (multiplicative) inverse ideal, written $\mathcal{I}^{-1}$.

Two ideals $\mathcal{I}, \mathcal{J} \subseteq R$ are *coprime* if $\mathcal{I} + \mathcal{J} = R$. An ideal $\mathfrak{p} \subsetneq R$ is *prime* if whenever $ab \in \mathfrak{p}$ for some $a, b \in R$, then $a \in \mathfrak{p}$ or $b \in \mathfrak{p}$ (or both). An ideal $\mathfrak{p}$ is prime if and only if it is *maximal*, i.e., if the only proper superideal of $\mathfrak{p}$ is R itself, which implies that the quotient ring $R/\mathfrak{p}$ is a finite field. The ring R has unique factorization of ideals, i.e., every ideal $\mathcal{I}$ can be expressed uniquely as a product of powers of prime ideals.

2.5.4 Duality

Here we recall the notion of a dual ideal and explain its close connection to both the inverse ideal and the dual lattice. For more details, see [Con09] as an accessible reference.

For any fractional ideal $\mathcal{I}$ in K, its *dual* is defined as

$$\mathcal{I}^\vee = \{a \in K \ : \ \mathrm{Tr}(a\mathcal{I}) \subseteq \mathbb{Z}\}.$$

It is easy to verify that $(\mathcal{I}^\vee)^\vee = \mathcal{I}$, that $\mathcal{I}^\vee$ is a fractional ideal, and that $\mathcal{I}^\vee$ embeds under σ as the (conjugate) dual lattice of $\mathcal{I}$, as defined in Section 2.4.

For any $\mathbb{Q}$-basis $B = \{b_j\}$ of K, we denote its dual basis by $B^\vee = \{b_j^\vee\}$, which is characterized by $\mathrm{Tr}(b_i \cdot b_j^\vee) = \delta_{ij}$, the Kronecker delta. It is immediate that $(B^\vee)^\vee = B$, and if B is a $\mathbb{Z}$-basis of some fractional ideal $\mathcal{I}$, then $B^\vee$ is a $\mathbb{Z}$-basis of its dual ideal $\mathcal{I}^\vee$. An important fact is that if $a = \sum_j a_j \cdot b_j$ for $a_j \in \mathbb{R}$ is the unique representation of $a \in K_\mathbb{R}$ in basis B, then $a_j = \mathrm{Tr}(a \cdot b_j^\vee)$ by linearity of trace.

Suppose that $K \cong \bigotimes_\ell K_\ell$ as in Proposition 2.13. Then by linearity and the tensorial decomposition of the trace (Equation (2.7)), taking the dual commutes with tensoring, i.e., $(\bigotimes_\ell B_\ell)^\vee = \bigotimes_\ell B_\ell^\vee$ for any $\mathbb{Q}$-bases B_ℓ of K_ℓ. In particular, this implies that $(\bigotimes_\ell \mathcal{I}_\ell)^\vee = \bigotimes_\ell \mathcal{I}_\ell^\vee$ for any fractional ideals $\mathcal{I}_\ell$ in K_ℓ.

Except in the trivial number field $K = \mathbb{Q}$, the ring of integers R is not self-dual, nor are an ideal and its inverse dual to each other. However, an ideal and its inverse *are* related by multiplication with the dual ideal $R^\vee$ of the ring: for any fractional ideal $\mathcal{I}$, its dual is $\mathcal{I}^\vee = \mathcal{I}^{-1} \cdot R^\vee$. The factor $R^\vee$ is often called the *codifferent*, and its inverse $(R^\vee)^{-1}$ the *different*, which is in fact an ideal in R. By Equation (2.9) and the fact that $\det(\sigma(R)) = \det(\sigma(R^\vee))^{-1}$, we have

$$\mathrm{N}(R^\vee) = \Delta_K^{-1}. \tag{2.10}$$

The codifferent $R^\vee$ plays an important role in ring-LWE and its applications. The following material shows that $R^\vee$ is a principal ideal with a particularly simple generator, and that $(R^\vee)^{-1} \subseteq R$ is an integral ideal. We include proofs for completeness. We start with a useful lemma characterizing the traces of the powers of ζ_m.

Lemma 2.15. *Let m be a power of a prime p and $m' = m/p$, and j be an integer. Then*

$$\mathrm{Tr}(\zeta_m^j) = \begin{cases} \varphi(p) \cdot m' & \text{if } j = 0 \bmod m \\ -m' & \text{if } j = 0 \bmod m', j \neq 0 \bmod m \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The first case is immediate, since $\zeta_m^j = 1$. Otherwise, let $d = \gcd(j, m)$ and $\tilde{m} = m/d$, so $\mathrm{Tr}(\zeta_m^j) = d \cdot \mathrm{Tr}_{\mathbb{Q}(\zeta_{\tilde{m}})/\mathbb{Q}}(\zeta_{\tilde{m}}^{j/d})$. Because j/d is coprime with $\tilde{m}$, the latter trace is the sum of all complex primitive $\tilde{m}$th roots of unity, which is -1 when $\tilde{m} = p$, and 0 otherwise. □

Lemma 2.16. *Let m be a power of a prime p and $m' = m/p$, and let $g = 1 - \zeta_p \in R = \mathbb{Z}[\zeta_m]$. Then $R^\vee = \langle g/m \rangle$; $p/g \in R$; and $\langle g \rangle$ and $\langle p' \rangle$ are coprime for every prime integer $p' \neq p$.*

Proof. To prove the first claim, we first show that $g/m \in R^\vee$. Since the power basis is a $\mathbb{Z}$-basis of R, it is necessary and sufficient to show that $\mathrm{Tr}(\zeta_m^j \cdot g/m) = \mathrm{Tr}(\zeta_m^j - \zeta_m^{j+m'})/m$ is an integer for every $j \in [\varphi(m)]$. By Lemma 2.15, it is $(\varphi(p) + 1)m'/m = 1$ for $j = 0$, and 0 for all other j. Now to show that $R^\vee = \langle g/m \rangle$, it suffices to show that $\mathrm{N}(g/m) = \mathrm{N}(R^\vee)$, the latter of which is $p^{m/p}/m^{\varphi(m)}$ by Equations (2.10) and (2.8). Now $\mathrm{N}(m) = m^{\varphi(m)}$, and $\mathrm{N}(1 - \zeta_p) = \mathrm{N}_{\mathbb{Q}(\zeta_p)/\mathbb{Q}}(1 - \zeta_p)^{m/p}$. Because the roots of $\Phi_p(X)$ are exactly the complex primitive pth roots of unity, the latter norm is exactly $\Phi_p(1) = p$, as desired.

To prove that $p/g \in R$, using $1 + \zeta_p + \zeta_p^2 + \cdots + \zeta_p^{p-1} = 0$ one may verify that

$$p = (1 - \zeta_p)\big((p-1) + (p-2)\zeta_p + \cdots + \zeta_p^{p-2}\big).$$

To prove the third claim, recall again that the norm of $\langle g \rangle$ is a power of p. Therefore, the norm of $\langle g \rangle + \langle p' \rangle$, being a divisor of both a power of p and of p', must be 1, implying that $\langle g \rangle$ and $\langle p' \rangle$ are coprime.□

Definition 2.17. *For $R = \mathbb{Z}[\zeta_m]$, define $g = \prod_p (1 - \zeta_p) \in R$, where p runs over all* odd *primes dividing m. Also define $t = \hat{m}/g \in R$, where $\hat{m} = m/2$ if m is even, otherwise $\hat{m} = m$.*

Notice that $\hat{m}/g \in R$ because $(1 - \zeta_2) = 2$, so $\hat{m}/g = m/\prod_p (1 - \zeta_p) \in R$, where here p runs over *all* primes dividing m.

Corollary 2.18. *Adopt the notation from Definition 2.17. Then $R^\vee = \langle g/\hat{m} \rangle = \langle t^{-1} \rangle$, and $\langle g \rangle$ is coprime with $\langle p' \rangle$ for every prime integer p' except those* odd *primes dividing m.*

Proof. Letting $m = \prod_\ell m_\ell$ be the prime-power factorization of m, where each m_ℓ is a power of some prime p_ℓ, and using the ring isomorphism $R \cong \bigotimes_\ell R_\ell$ where $R_\ell = \mathbb{Z}[\zeta_{m_\ell}]$, we can equivalently express g as $g = (\hat{m}/m)(\otimes_\ell g_\ell)$, where $g_\ell = (1 - \zeta_{p_\ell})$. Then by Lemma 2.16,

$$\Big(\bigotimes\nolimits_\ell R_\ell\Big)^\vee = \bigotimes\nolimits_\ell (R_\ell^\vee) = \bigotimes\nolimits_\ell (g_\ell/m_\ell) R_\ell = (g/\hat{m}) \cdot \Big(\bigotimes\nolimits_\ell R_\ell\Big),$$

as desired.

For the coprimality claim, the norm of g is a product of powers of the odd primes dividing m, and the claim follows by the same reasoning as in Lemma 2.16. □

2.5.5 Prime Splitting and Chinese Remainder Theorem

For an integer prime $p \in \mathbb{Z}$, the factorization of the principal ideal $\langle p \rangle \subset R = \mathbb{Z}[\zeta_m]$ is as follows. Let $d \geq 0$ be the largest integer such that p^d divides m, let $h = \varphi(p^d)$, and let $f \geq 1$ be the multiplicative order of p modulo m/p^d. Then $\langle p \rangle = \mathfrak{p}_1^h \cdots \mathfrak{p}_g^h$, where $g = n/(hf)$ and the $\mathfrak{p}_i$ are distinct prime ideals each of norm p^f.

A particular case of interest for us is the factorization of an integer prime $q = 1 \bmod m$, and the form of its prime ideal factors. Here the order of q modulo m is 1, and so $\langle q \rangle$ "splits completely" into n distinct prime ideals of norm q. Notice that the field $\mathbb{Z}_q$ has a primitive root of unity ω_m, because the multiplicative

group of $\mathbb{Z}_q$ is cyclic with order $q-1$. Indeed, there are $n = \varphi(m)$ distinct such roots of unity $\omega_m^i \in \mathbb{Z}_q$, for $i \in \mathbb{Z}_m^*$, and the prime ideal factors of $\langle q \rangle$ are simply $\mathfrak{q}_i = \langle q \rangle + \langle \zeta_m - \omega_m^i \rangle$. Therefore, each quotient ring $R/\mathfrak{q}_i$ is isomorphic to the field $\mathbb{Z}_q$, via the map $\zeta_m \mapsto \omega_m^i$.

The Chinese Remainder Theorem says that if $\mathfrak{p}_i$ are pairwise coprime ideals in R, then the natural ring homomorphism from $R/\prod_i \mathfrak{p}_i$ to the product ring $\prod_i (R/\mathfrak{p}_i)$ is in fact an isomorphism. To support efficient operations in $R_q = R/qR$, we will use the following special case, which we use to define a special $\mathbb{Z}_q$-basis of R_q (see Section 5 for details).

Lemma 2.19. *Let $q = 1 \bmod m$ be prime, and let $\omega_m \in \mathbb{Z}_q$ and ideals $\mathfrak{q}_i$ be as above. Then the natural ring homomorphism $R/\langle q \rangle \to \prod_{i \in \mathbb{Z}_m^*} (R/\mathfrak{q}_i) \cong (\mathbb{Z}_q)^n$ is an isomorphism.*

2.6 Ring-LWE

We now provide the formal definition of the ring-LWE problem and describe the worst-case hardness result shown in [LPR10]. We remark that our definition here differs very slightly from the one used in [LPR10]: we scale the b component by a factor of q, so that it is an element of $K_{\mathbb{R}}/qR^\vee$ and not $K_{\mathbb{R}}/R^\vee$ as in [LPR10]. This is done for convenience when later discretizing the b component, and the two definitions are easily seen to be equivalent.

Definition 2.20 (Ring-LWE Distribution). *For a "secret" $s \in R_q^\vee$ (or just $R^\vee$) and a distribution ψ over $K_{\mathbb{R}}$, a sample from the ring-*LWE *distribution $A_{s,\psi}$ over $R_q \times (K_{\mathbb{R}}/qR^\vee)$ is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \psi$, and outputting $(a, b = a \cdot s + e \bmod qR^\vee)$.*

Definition 2.21 (Ring-LWE, Average-Case Decision). *The* average-case decision *version of the ring-*LWE *problem, denoted R-*DLWE*$_{q,\psi}$, is to distinguish with non-negligible advantage between independent samples from $A_{s,\psi}$, where $s \leftarrow R_q^\vee$ is uniformly random, and the same number of uniformly random and independent samples from $R_q \times (K_{\mathbb{R}}/qR^\vee)$.*

Theorem 2.22. *Let K be the mth cyclotomic number field having dimension $n = \varphi(m)$ and $R = \mathcal{O}_K$ be its ring of integers. Let $\alpha = \alpha(n) > 0$, and let $q = q(n) \geq 2$, $q = 1 \bmod m$ be a $\mathrm{poly}(n)$-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. Then there is a polynomial-time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices in K to the problem of solving R-*DLWE*$_{q,\psi}$ given only ℓ samples, where ψ is the Gaussian distribution $D_{\xi q}$ for $\xi = \alpha \cdot (n\ell/\log(n\ell))^{1/4}$.*

Note that the above worst-case hardness result deteriorates with the number of samples ℓ. Since most applications only require a small (or even a constant) number of samples, this is not a serious issue. In cases where a large number of samples is needed, one can use two alternative hardness theorems proven in [LPR10]. The first assumes hardness of the search problem for spherical Gaussian error, which as yet lacks a reduction from a worst-case problem. The second is a reduction from a worst-case problem, and it allows an arbitrary number of samples without any deterioration in the approximation factor; it does, however, require the error distribution to be non-spherical and chosen in a specific way, which makes it somewhat less convenient in implementations. We refer to [LPR10] for additional information.

In applications it is often useful to work with a version of ring-LWE whose error distribution is discrete. This leads naturally to a definition of $A_{s,\chi}$ for a discrete error distribution χ over $R^\vee$, with b being an element of $R_q^\vee$. We similarly modify Definition 2.21 by letting R-DLWE$_{q,\chi}$ be the problem of distinguishing between $A_{s,\chi}$ and uniform samples from $R_q \times R_q^\vee$. As we show next, for a wide family of discrete error distributions, the hardness of the discrete version follows from that of the continuous one. In more detail, the lemma

below implies that if R-DLWE$_{q,\psi}$ is hard with some number ℓ of samples, then so is R-DLWE$_{q,\chi}$ with the same number of samples, where the error distribution χ is $\lfloor p \cdot \psi \rceil_{w+pR^\vee}$ for some integer p coprime to q, $\lfloor \cdot \rceil$ is any valid discretization to (cosets of) $pR^\vee$, and w is an arbitrary element in $R_p^\vee$ that can vary from sample to sample (even adaptively and adversarially). In particular, for $p = 1$ we get hardness with error distribution $\lfloor \psi \rceil_{R^\vee}$.

Lemma 2.23. *Let p and q be positive coprime integers, and $\lfloor \cdot \rceil$ be a valid discretization to (cosets of) $pR^\vee$. There exists an efficient transformation that on input $w \in R_p^\vee$ and a pair in $(a', b') \in R_q \times K_\mathbb{R}/qR^\vee$, outputs a pair $(a = pa' \bmod qR, b) \in R_q \times R_q^\vee$ with the following guarantees: if the input pair is uniformly distributed then so is the output pair; and if the input pair is distributed according to the ring-*LWE *distribution $A_{s,\psi}$ for some (unknown) $s \in R^\vee$ and distribution ψ over $K_\mathbb{R}$, then the output pair is distributed according to $A_{s,\chi}$, where $\chi = \lfloor p \cdot \psi \rceil_{w+pR^\vee}$.*

Proof. Given w and a sample $(a', b') \in R_q \times K_\mathbb{R}/qR^\vee$, the transformation discretizes $pb' \in K_\mathbb{R}/pqR^\vee$ to $\lfloor pb' \rceil_{w+pR^\vee} \in (w + pR^\vee) + pqR^\vee$. It then lets $a = pa' \bmod qR$ and $b = \lfloor pb' \rceil_{w+pR^\vee} \bmod qR^\vee$, and outputs the sample $(a, b) \in R_q \times R_q^\vee$.

If the distribution of (a', b') is $A_{s,\psi}$, then $pb' = (pa') \cdot s + pe' \bmod pqR^\vee$ for $e' \leftarrow \psi$. Because $(pa') \cdot s \in pR^\vee/pqR^\vee$, by validity of the discretization we have that $\lfloor pb' \rceil_{w+pR^\vee}$ and $(pa') \cdot s + \lfloor pe' \rceil_{w+pR^\vee}$ are identically distributed. Because p and q are coprime, $a = pa' \bmod qR$ is uniformly random over R_q, so (a, b) has distribution $A_{s,\chi}$.

On the other hand, if (a', b') is uniformly random, then a is uniform over R_q. Moreover, since the uniform distribution over $K_\mathbb{R}/pqR^\vee$ is invariant under shifts by $pR^\vee$, then by validity so is the distribution of $b = \lfloor pb' \rceil_{w+pR^\vee} \bmod qR^\vee$, for any $w \in R^\vee$. Then because p and q are coprime, b is uniformly random over $R_q^\vee$ and independent of a, as desired. □

Finally, another important variant of ring-LWE, known as the "normal form," is the one in which the secret, instead of being uniformly distributed, is chosen from the error distribution (discretized to $R^\vee$, or a coset of $pR^\vee$ as in Lemma 2.23 above). This modification makes the secret short, which is very useful in some applications. We now show that this variant of ring-LWE is as hard as the original one, closely following the technique of [ACPS09].

Lemma 2.24. *Let p and q be positive coprime integers, $\lfloor \cdot \rceil$ be a valid discretization to (cosets of) $pR^\vee$, and w be an arbitrary element in $R_p^\vee$. If R-DLWE$_{q,\psi}$ is hard given some number ℓ of samples, then so is the variant of R-DLWE$_{q,\psi}$ in which the secret is sampled from $\chi := \lfloor p \cdot \psi \rceil_{w+pR^\vee}$, given $\ell - 1$ samples.*

Proof. We show how to solve the former problem given an oracle for the latter. Start by drawing one sample from the unknown distribution and apply the transformation from Lemma 2.23 (with p, w, and $\lfloor \cdot \rceil$) to it. Let $(a_0, b_0) \in R_q \times R_q^\vee$ be the result. If a_0 is not in R_q^*, abort and reject. Otherwise, let $a_0^{-1} \in R_q^*$ denote its inverse. Draw $\ell - 1$ additional samples $(a_i, b_i) \in R_q \times K_\mathbb{R}/qR^\vee$ $(i = 1, \ldots, \ell - 1)$ from the unknown distribution, and return the oracle's output when applied to the pairs

$$(a_i' = -a_0^{-1} a_i \ , \ b_i' = b_i + a_i' b_0) \in R_q \times K_\mathbb{R}/qR^\vee.$$

To prove this gives a valid distinguisher, notice first that by Claim 2.25 below, it suffices to show a noticeable distinguishing gap conditioned on a_0 being invertible. Next, observe that if the input distribution is uniform, then so is the distribution of the pairs (a_i', b_i'). Finally, if the input distribution is $A_{s,\psi}$ for

some $s \in R^\vee$, then we have $b_0 = a_0 \cdot s + e_0$ where e_0 is distributed according to χ. Therefore, for each $i = 1, \ldots, \ell - 1$,

$$b'_i = (a_i \cdot s + e_i) - a_0^{-1} a_i (a_0 \cdot s + e_0) = e_i + a'_i e_0,$$

where the e_i are distributed according to ψ, and so the input to the oracle consists of independent samples from $A_{e_0,\psi}$, as required. □

Claim 2.25. *Consider the mth cyclotomic field of degree $n = \varphi(m)$ for some $m \geq 2$. Then for any $q \geq 2$, the fraction of invertible elements in R_q is at least $1/\operatorname{poly}(n, \log q)$.*

When $q = 1 \bmod m$ is a prime (as in Theorem 2.22), we have by Lemma 2.19 that the fraction of invertible elements in R_q is $(1 - 1/q)^n \geq (1 - 1/(n+1))^n \geq e^{-1}$. This uses the inequality $1 - 1/(\alpha + 1) \geq e^{-1/\alpha}$ for $\alpha > 0$, which we will use again in the proof below.

Proof. We first observe that for any integer $r \geq 1$ and prime ideal $\mathfrak{p}$, an element $a \in R$ is invertible modulo $\mathfrak{p}^r$ if and only if $a \neq 0 \bmod \mathfrak{p}$, and therefore the fraction of uninvertible elements in $R/\mathfrak{p}^r$ is $1/\operatorname{N}(\mathfrak{p})$. One direction is obvious: if $a = 0 \bmod \mathfrak{p}$, then so is $a \cdot b$ for any $b \in R$, so a is uninvertible (because $1 \notin \mathfrak{p}$). For the other direction, if $a \neq 0 \bmod \mathfrak{p}$, then $\mathfrak{p} \nmid \langle a \rangle$, and so $\langle a \rangle, \mathfrak{p}^r$ are coprime, i.e., $\langle a \rangle + \mathfrak{p}^r = R$. Therefore, there exists $b \in R$ such that $ab \in 1 + \mathfrak{p}^r$.

Using the factorization of the ideals $\langle p \rangle$ given in Section 2.5.5 and the Chinese remainder theorem, we get that the fraction of invertible elements in R_q is

$$\prod_{\text{prime } p | q} (1 - p^{-f_p})^{n/(f_p \varphi(p^{d_p}))} \geq \prod_{\text{prime } p | q} (1 - p^{-f_p})^{n/\varphi(p^{d_p})}, \tag{2.11}$$

where d_p is the largest integer such that p^{d_p} divides m and f_p is the multiplicative order of p modulo m/p^{d_p}. For any prime p we clearly have $p^{f_p} > m \geq m/p^{d_p}$, and therefore

$$\begin{aligned}(1 - p^{-f_p})^{n/\varphi(p^{d_p})} &= (1 - p^{-f_p})^{\varphi(m/p^{d_p})} \\ &\geq (1 - p^{-f_p})^{m/p^{d_p}} \geq e^{-1}.\end{aligned}$$

As a result, the product in (2.11), restricted to primes p dividing m, of which there are at most $\log_2 m$, is at least $1/\operatorname{poly}(m)$. It therefore suffices to bound from below the product in (2.11) restricted to primes p not dividing m. For such primes p we have $d_p = 0$, and the expression simplifies to

$$\prod_{p | q, p \nmid m} (1 - p^{-f_p})^n, \tag{2.12}$$

where f_p is the multiplicative order of p modulo m. Notice that the values p^{f_p} are distinct for distinct p. Moreover, they are all 1 modulo m. Therefore, since the product in (2.12) includes at most $\log_2 q$ terms, we can bound it from below by

$$\prod_{k=1}^{\log_2 q} \left(1 - \frac{1}{km+1}\right)^n \geq \prod_{k=1}^{\log_2 q} e^{-n/km} \geq \prod_{k=1}^{\log_2 q} e^{-1/k} \geq e^{-1} \prod_{k=2}^{\log_2 q} \left(1 - \frac{1}{k}\right) = (e \cdot \log_2 q)^{-1}.$$

□

3 Sparse Decompositions of DFT and CRT

Here we give structured (or "sparse") decompositions of two important linear transformations, which lead to fast algorithms for applying them. We follow the algebraic framework of [PM08].

Definition 3.1. *Let m be a prime power and let $\mathcal{R}$ denote any commutative ring containing some element ω_m of multiplicative order m, i.e., a primitive mth root of unity.*

- *The* discrete Fourier transform DFT_m *over $\mathcal{R}$ is the $\mathbb{Z}_m$-by-$\mathbb{Z}_m$ matrix whose (i,j)th entry is ω_m^{ij}.*
- *The* Chinese remainder transform CRT_m *over $\mathcal{R}$ is the (square) submatrix of DFT_m obtained by restricting to the rows indexed by $\mathbb{Z}_m^*$ and the columns indexed by $[\varphi(m)]$.*

For an arbitrary positive integer m having prime-power factorization $m = \prod_\ell m_\ell$, where $\mathcal{R}$ has an mth root of unity (and hence has primitive m_ℓth roots of unity for each m_ℓ), the DFT and CRT matrices are

$$\mathrm{DFT}_m = \bigotimes_\ell \mathrm{DFT}_{m_\ell} \quad \textit{and} \quad \mathrm{CRT}_m = \bigotimes_\ell \mathrm{CRT}_{m_\ell}.$$

We identify the matrices DFT_m and CRT_m with the linear transforms they represent.

For a prime power m, applying DFT_m corresponds with evaluating a polynomial in $\mathcal{R}[X]$ of degree less than m (represented by its vector of coefficients in the natural order) at all the mth roots of unity $\omega_m^i \in \mathcal{R}$ for $i \in [m]$. Similarly, CRT_m corresponds with evaluating a polynomial of degree less than $\varphi(m)$ at all the *primitive* mth roots of unity ω_m^i for $i \in \mathbb{Z}_m^*$. (This interpretation, and its connection with Lemma 2.19, explains our choice of the name "Chinese remainder transform.")

For m with prime-power factorization $m = \prod_\ell m_\ell$, it can be shown using the Good-Thomas decomposition that DFT_m again corresponds with polynomial evaluation at all mth roots of unity, but under some permutations of the input and output vectors. For CRT_m, the correspondence with polynomial evaluation is different, because the columns of CRT_m typically do not correspond to powers $0, \ldots, \varphi(m)-1$ of a primitive mth root of unity ω_m. Instead, CRT_m corresponds with evaluation of a multivariate polynomial (with one variable per factor m_ℓ) at all input tuples in which the ℓth element is a primitive m_ℓth root of unity. We adopt the tensorial form of CRT_m because it corresponds directly with the tensorial (or multivariate) decomposition of the mth cyclotomic number field, and admits a finer-grained decomposition and more efficient algorithms than the univariate perspective.

Decomposition of DFT_m. Let m be a power of some prime p, and let $m' = m/p$. Using the Cooley-Tukey decomposition we can express DFT_m in terms of smaller DFTs of dimensions p and m', and by iterating, in terms of DFT_p alone. Reindex the columns of DFT_m by pairs $(j_0, j_1) \in [p] \times [m']$, using the standard correspondence $j = m'j_0 + j_1 \in [m]$. Similarly, reindex the rows by pairs $(i_0, i_1) \in [p] \times [m']$, this time using the (nonstandard) correspondence $i = pi_1 + i_0 \in [m]$.[8] We then have the decomposition

$$\mathrm{DFT}_m = (I_{[p]} \otimes \mathrm{DFT}_{m'}) \cdot T_m \cdot (\mathrm{DFT}_p \otimes I_{[m']}), \tag{3.1}$$

where all three terms are $([p] \times [m'])$-by-$([p] \times [m'])$ matrices, and T_m is the diagonal "twiddle" matrix having entry $\omega_m^{i_0 i_1}$ in its (i_0, i_1)th diagonal entry. Therefore, applying DFT_m reduces to m' parallel applications of DFT_p, followed by m parallel scalar multiplications by twiddle factors, followed by p parallel applications

[8] This relabeling corresponds with the "bit-reversal" or related "stride" output permutation in the standard decimation-in-frequency FFT algorithm. In an implementation, the permutation can be omitted because the output does not need to be in any particular order.

of $\mathrm{DFT}_{m'}$. Of course, each $\mathrm{DFT}_{m'}$ can be further decomposed in the same way, down to the DFT_p base case. Using any of the Rader, Winograd, or Bluestein FFT algorithms, we can apply such base cases in $O(p \log p)$ time, which implies that DFT_m can be applied in $O(n \log n)$ time, where $n = \varphi(m)$.

To verify Equation (3.1), it suffices by linearity to compare the action of both sides on the standard basis vectors. Take any $(j_0, j_1) \in [p] \times [m']$ and consider the vector with 1 in location (j_0, j_1) and zero elsewhere. Applying $\mathrm{DFT}_p \otimes I_{[m']}$ to it yields the vector that is $\omega_p^{i_0 j_0}$ in locations (i_0, j_1) for $i_0 \in [p]$ and zero elsewhere. The matrix T_m changes these nonzero entries to $\omega_p^{i_0 j_0} \omega_m^{i_0 j_1}$, and finally, $I_{[p]} \otimes \mathrm{DFT}_{m'}$ yields the vector with

$$\omega_p^{i_0 j_0} \cdot \omega_m^{i_0 j_1} \cdot \omega_{m'}^{i_1 j_1} = \omega_m^{m' i_0 j_0 + i_0 j_1 + p i_1 j_1} = \omega_m^{(p i_1 + i_0)(m' j_0 + j_1)}$$

in any location $(i_0, i_1) \in [p] \times [m']$, as required.

Decomposition of CRT_m. Letting m, p, and m' be as above, notice that $\varphi(m) = \varphi(p) \cdot m'$. Moreover, with the above reindexing of rows and columns, CRT_m is the submatrix of DFT_m restricted to rows $\mathbb{Z}_p^* \times [m']$ and columns $[\varphi(p)] \times [m']$. By appropriately restricting the matrices in Equation (3.1), we obtain the decomposition (which can be verified in the same way as above)

$$\mathrm{CRT}_m = (I_{\mathbb{Z}_p^*} \otimes \mathrm{DFT}_{m'}) \cdot \hat{T}_m \cdot (\mathrm{CRT}_p \otimes I_{[m']}), \tag{3.2}$$

where $\hat{T}_m$ is the diagonal twiddle matrix T_m from above, restricted to the rows and columns indexed by $\mathbb{Z}_p^* \times [m']$. Applying CRT_m therefore reduces to m' parallel applications of CRT_p, followed by $\varphi(m)$ parallel scalar multiplications by twiddle factors, followed by $\varphi(p)$ parallel applications of $\mathrm{DFT}_{m'}$.

Inversion. Using the inversion rules for matrix multiplication and the Kronecker product, the inverse DFT and CRT decompose as

$$\mathrm{DFT}_m^{-1} = (\mathrm{DFT}_p^{-1} \otimes I_{[m']}) \cdot T_m^{-1} \cdot (I_{[p]} \otimes \mathrm{DFT}_{m'}^{-1}) \tag{3.3}$$

$$\mathrm{CRT}_m^{-1} = (\mathrm{CRT}_p^{-1} \otimes I_{[m']}) \cdot (\hat{T}_m)^{-1} \cdot (I_{\mathbb{Z}_p^*} \otimes \mathrm{DFT}_{m'}^{-1}), \tag{3.4}$$

and can be applied at exactly the same cost as their forward counterparts. Note that the row and column index sets of CRT_m are different (as they are for CRT_p and $\hat{T}_m$ as well), so $\mathrm{CRT}_m^{-1} \cdot \mathrm{CRT}_m$ and $\mathrm{CRT}_m \cdot \mathrm{CRT}_m^{-1}$ are "different" matrices, although they are both still identity matrices over the appropriate index sets.

Arbitrary m. For m that may have more than one prime divisor, the tensorial form of CRT_m leads immediately to a fast algorithm. Specifically, if m has prime-power factorization $m = \prod_\ell m_\ell$, then by the mixed-product property, applying $\mathrm{CRT}_m = \bigotimes_\ell \mathrm{CRT}_{m_\ell}$ reduces to $\varphi(m/m_\ell)$ parallel applications of CRT_{m_ℓ}, in sequence for each ℓ (see the end of Section 2.1). Since each CRT_{m_ℓ} can be applied in $O(m_\ell \log m_\ell)$ time and $O(\log m_\ell)$ parallel depth, the total runtime and parallel depth are $O(m \log m)$ and $O(\log m)$, respectively.

4 The Powerful Basis

In this section (and Section 6) we study certain $\mathbb{Z}$-bases of certain fractional ideals $\mathcal{I}$ in $K = \mathbb{Q}(\zeta_m)$, which are therefore $\mathbb{Z}_q$-bases of the quotients $\mathcal{I}_q = \mathcal{I}/q\mathcal{I}$ for any positive integer q. Fixing such a basis $\vec{b}$ and viewing it as a (column) vector over $\mathcal{I}$, we can represent any $a \in \mathcal{I}$ (respectively, $a \in \mathcal{I}_q$) uniquely as

$a = \langle \vec{b}, \mathbf{a} \rangle = \vec{b}^T \cdot \mathbf{a}$ for some coefficient vector $\mathbf{a}$ over $\mathbb{Z}$ (respectively, $\mathbb{Z}_q$) having the same index set as $\vec{b}$. Our algorithms simply store and operate on these coefficient vectors, while also keeping track of the corresponding basis, which will be one of the few we define below. Notice that by linearity, if we have some $a \in \mathcal{I}$ represented by coefficient vector $\mathbf{a}$ in basis $\vec{b}$, then $\mathbf{a}$ is also the representation of $ra \in r\mathcal{I}$ in the basis $r\vec{b}$ for any $r \in K$, so we can switch between the two values at essentially no cost.

Here we define a certain useful $\mathbb{Z}$-basis of R, and hence $\mathbb{Q}$-basis of K. We call it the "powerful" basis, due to its decomposition in terms of the power bases of prime-power cyclotomics, and the fast algorithms associated with it.[9]

Definition 4.1. *The* powerful *basis $\vec{p}$ of $K = \mathbb{Q}(\zeta_m)$ and $R = \mathbb{Z}[\zeta_m]$ is defined as follows:*

- *For a prime power m, define $\vec{p}$ to be the power basis $(\zeta_m^j)_{j \in [\varphi(m)]}$, treated as a vector over $R \subset K$.*
- *For m having prime-power factorization $m = \prod_\ell m_\ell$, define $\vec{p} = \bigotimes_\ell \vec{p_\ell}$, the tensor product of the power(ful) bases $\vec{p_\ell}$ of each $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$.*

For any power $\mathcal{I} = (R^\vee)^k$ of $R^\vee = \langle t^{-1} \rangle$, define the powerful basis of $\mathcal{I}$ to be $t^{-k} \cdot \vec{p}$.

By definition of the tensor product, $\vec{p}$ is a vector with index set $\prod_\ell [\varphi(m_\ell)]$. So to specify an entry of $\vec{p}$ we need one index $j_\ell \in [\varphi(m_\ell)]$ per prime divisor of m, and the specified entry is $p_{(j_\ell)} = \prod_\ell \zeta_{m_\ell}^{j_\ell}$. Note that because $\zeta_{m_\ell} = \zeta_m^{m/m_\ell} \in K$, it is possible to "flatten" the index set to a size-$\varphi(m)$ subset of $[m]$, where index tuple (j_ℓ) maps to $j = \sum_\ell (m/m_\ell) j_\ell \bmod m$, and $p_j = \zeta_m^j$. We note that unless m is a prime power, the flattened index set is *not* equal to $[\varphi(m)]$, so the powerful basis differs from the power basis, although it still consists of powers of ζ_m. For instance, for $m = 15$ and $\zeta = \zeta_{15}$, the powerful basis consists of $\zeta^0, \zeta^3, \zeta^5, \zeta^6, \zeta^8, \zeta^9, \zeta^{11}$, and ζ^{14}. Because the flattened indices tend to be a somewhat irregular subset of $[m]$, it is usually preferable to maintain the structured index set.

Observe that $\vec{p}^T$ is a row vector (over K) with columns indexed by $\prod_\ell [\varphi(m_\ell)]$. Applying the canonical embedding σ entry-wise to obtain column vectors indexed by $\mathbb{Z}_m^*$ (or equivalently, $\prod_\ell \mathbb{Z}_{m_\ell}^*$), by Equation (2.6) we obtain the complex matrix $\sigma(\vec{p}^T) = \mathrm{CRT}_m$. With this fact in mind, we now prove two basic facts about the geometry of the powerful basis. The first says that all its elements are short (and in fact, by Lemma 2.14 they are shortest nonzero elements of R), and the second statement says essentially that the elements are close to orthogonal.

Claim 4.2. *The length of each element p_j of $\vec{p}$ in ℓ_∞ norm is $\|p_j\|_\infty = 1$, and in ℓ_2 norm is $\|p_j\|_2 = \sqrt{\varphi(m)} = \sqrt{n}$.*

Proof. Each entry in the CRT_m matrix is a root of unity, hence it has magnitude 1, and so the ℓ_∞ and ℓ_2 norms of each column are 1 and $\sqrt{\varphi(m)}$, respectively. □

Lemma 4.3. *The largest singular value of $\sigma(\vec{p}^T)$ (or equivalently, of CRT_m) is $s_1(\vec{p}) = \sqrt{\hat{m}}$, and the smallest singular value is $s_n(\vec{p}) = \sqrt{m/\operatorname{rad}(m)}$.*

Notice that the ratio of $s_1(\vec{p})$ to $\sqrt{\varphi(m)}$ (i.e., the ℓ_2 norm of each basis element) is just $\sqrt{\hat{m}/\varphi(m)} = \left(\prod_p p/(p-1)\right)^{1/2} = O(\sqrt{\log \log m})$, where the product runs over all odd primes dividing m.

[9] Although we define the powerful basis in a different way, it can be seen that it coincides with what Bosma [Bos90] calls the "canonical" basis of R. Bosma's work is the only one we know of that explicitly considers this basis.

Proof. It suffices to prove the statement when m is a prime power, due to the tensor structure of CRT_m, and the fact that the vector of singular values of $A \otimes B$ is the tensor product of the two vectors of singular values of A and B. So let m be a power of a prime p, and let $m' = m/p$. By Equation (3.2),

$$\mathrm{CRT}_m = (\sqrt{m'}Q) \cdot (\mathrm{CRT}_p \otimes I_{[m']})$$

for some unitary matrix Q, because $\mathrm{DFT}_{m'}/\sqrt{m'}$ is unitary for any m', and so is the twiddle matrix $\hat{T}_m$. The lemma then follows immediately from the fact that the $\varphi(p) = p - 1$ eigenvalues of the Gram matrix

$$\mathrm{CRT}_p^* \cdot \mathrm{CRT}_p = (pI_{[\varphi(p)]} - \mathbf{1} \cdot \mathbf{1}^T) \tag{4.1}$$

are $p, \ldots, p$ ($p-2$ times) and 1, where the asterisk denotes the conjugate transpose, $\mathbf{1} \in \mathbb{R}^{[\varphi(p)]}$ is the all-ones vector, and the equality is by the fact that CRT_p is obtained by removing the all-1s row and one column from DFT_p, which is a unitary matrix scaled up by a $\sqrt{p}$ factor. □

We conclude this section by characterizing the Gram-Schmidt orthogonalization $\widetilde{\mathrm{CRT}_m}$ of the powerful basis (under the canonical embedding), in Lemma 4.4 below. This orthogonalization is used in the nearest-plane [Bab85] and Klein/GPV [GPV08] algorithms (see Lemma 2.9), which we use for sampling from discrete Gaussians over R. The lemma implies that the orthogonalization is structured so that these algorithms can be executed in substantially less time, and using much less precision, than is required for an arbitrary basis. This is because the U matrix associated with the orthogonalization is block diagonal with $m/\operatorname{rad}(m)$ identical square blocks of dimension $\operatorname{rad}(m)$, which allows an implementation to make $m/\operatorname{rad}(m)$ parallel and independent calls to a quadratic-time subroutine on dimension $\operatorname{rad}(m)$, for $O(m \operatorname{rad}(m))$ scalar operations in total. Moreover, each row of U has a small (common) denominator, allowing an implementation to compute inner products with the rows of U using low-precision integers (see the discussion following Lemma 2.9).

Recalling from Section 2.2 the matrix form of the Gram-Schmidt orthogonalization, it follows by the mixed-product property that $\widetilde{A \otimes B} = \widetilde{A} \otimes \widetilde{B}$. By the tensor structure of CRT_m, it therefore suffices to consider the case where m is a prime power.

Lemma 4.4. *Let m be a power of a prime p and $m' = m/p$. Then*

$$\mathrm{CRT}_m = Q_m \cdot (\sqrt{m'}D_p \otimes I_{[m']}) \cdot (U_p \otimes I_{[m']}),$$

where Q_m is unitary, D_p is the real diagonal $[\varphi(p)]$-by-$[\varphi(p)]$ matrix with $\sqrt{(p-1) - j/(p-j)}$ in its jth diagonal entry, and U_p is the upper unitriangular $[\varphi(p)]$-by-$[\varphi(p)]$ matrix with $-1/(p-i-1)$ in its (i,j)th entry, for $0 \le i < j < \varphi(p)$.

Proof. By Equation (3.2) and the fact that $\hat{T}_m$ and $\mathrm{DFT}_{m'}/\sqrt{m'}$ are unitary matrices, we have

$$\mathrm{CRT}_m = \sqrt{m'}Q' \cdot (\mathrm{CRT}_p \otimes I_{[m']})$$

for some unitary Q'. Thus it suffices to show that $\mathrm{CRT}_p = Q_p \cdot D_p \cdot U_p$ for some unitary Q_p.

Let $G = \mathrm{CRT}_p^* \cdot \mathrm{CRT}_p$ be the Gram matrix of CRT_p and recall from Equation (4.1) that G has diagonal entries $p-1$, and -1 entries elsewhere. As discussed in Section 2.2, by the uniqueness of the Cholesky decomposition it suffices to show that

$$G = U_p^T \cdot D_p^2 \cdot U_p.$$

This equality can be verified by an elementary calculation, as follows. For $k \geq 1$, define

$$T(k) := \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \cdots + \frac{1}{(k-1) \cdot k}.$$

It is easy to see (by induction, or by noticing that adding $1/k$ to the above collapses the sum) that $T(k) = 1 - 1/k$. For any $i \in [\varphi(p)]$, the ith diagonal entry in $U_p^T \cdot D_p^2 \cdot U_p$ is

$$p - 1 - \frac{i}{p-i} + \sum_{k=0}^{i-1} \frac{1}{(p-k-1)^2}\Big(p - 1 - \frac{k}{p-k}\Big).$$

The summation in the above expression is

$$p \sum_{k=0}^{i-1} \frac{1}{(p-k)(p-k-1)} = p\big(T(p) - T(p-i)\big) = p\Big(1 - \frac{1}{p} - 1 + \frac{1}{p-i}\Big) = \frac{i}{p-i},$$

and so the ith diagonal entry is $p - 1$, as required. The off-diagonal entries are calculated in essentially the same way. □

5 The Chinese Remainder Basis and Fast Ring Operations

When working in K or R, we can perform ring operations efficiently by representing elements under the canonical embedding σ. Recall that σ is the ring embedding from $K = \mathbb{Q}(\zeta_m)$ into the product ring $H \subset \mathbb{C}^{\mathbb{Z}_m^*}$ that maps ζ_m to each power $\omega_m^i \in \mathbb{C}$ for $i \in \mathbb{Z}_m^*$, where ω_m is a primitive complex mth root of unity. Under the canonical embedding, addition and multiplication simply apply coordinate-wise on each complex coordinate. Converting to the embedding representation from the powerful basis $\vec{p}$ is done simply by multiplying (with sufficient precision) by the complex matrix $\mathrm{CRT}_m = \sigma(\vec{p}^T)$, i.e., if $a = \langle \vec{p}, \mathbf{a} \rangle \in K$ for some rational vector $\mathbf{a}$ then $\sigma(a) = \mathrm{CRT}_m \cdot \mathbf{a}$.

In ring-LWE and its applications, we often work in R_q and $R_q^\vee$, and sometimes in $\mathcal{I}_q$ for $\mathcal{I} = (R^\vee)^k$, where q is a prime integer congruent to 1 modulo m.[10] While using the canonical embedding as above lets us perform ring operations relatively efficiently in these quotients (by using an arbitrary set of representatives), here we describe more efficient and practical algorithms that only use arithmetic in $\mathbb{Z}_q$, rather than on high-precision complex numbers. These algorithms are facilitated by what we call the *Chinese remainder* (CRT) basis for $\mathcal{I}_q$, defined next.

Recalling that $R \cong \bigotimes_\ell R_\ell$ where $m = \prod_\ell m_\ell$ is the prime-power factorization of m and R_ℓ is the m_ℓth cyclotomic ring, it is easy to verify that the quotient ring $R_q \cong \bigotimes_\ell (R_\ell / qR_\ell)$. Therefore we may focus on the case of prime-power m. Also recall from Section 2.5.5 the prime ideal factorization $\langle q \rangle = \prod_{i \in \mathbb{Z}_m^*} \mathfrak{q}_i$ in R, where $\mathfrak{q}_i = \langle q \rangle + \langle \zeta_m - \omega_m^i \rangle$ is prime in R and ω_m is some fixed element of order m in $\mathbb{Z}_q$.

Definition 5.1. *For a positive integer m, the Chinese remainder (or CRT) $\mathbb{Z}_q$-basis $\vec{c}$ of R_q is as follows:*

- *For a prime power m, $\vec{c} = (c_i)_{i \in \mathbb{Z}_m^*}$ is characterized by $c_i = 1 \bmod \mathfrak{q}_i$ and $c_i = 0 \bmod \mathfrak{q}_j$ for $i \neq j$. (Its existence is guaranteed by Lemma 2.19, the Chinese remainder theorem.)*

[10] The modulus q may also be a product of several primes $q_i = 1 \bmod m$, in which case we can use the Chinese Remainder Theorem to decompose R_q into the product of rings R_{q_i}.

- *For m having prime-power factorization $m = \prod_\ell m_\ell$, define $\vec{c} = \bigotimes_i \vec{c_\ell}$, the tensor product of the CRT bases $\vec{c_\ell}$ of each R_ℓ/qR_ℓ.*

For any power $\mathcal{I} = (R^\vee)^k$ of $R^\vee = \langle t^{-1} \rangle$, the CRT $\mathbb{Z}_q$-basis of $\mathcal{I}_q$ is $t^{-k} \cdot \vec{c}$.

Note that $\vec{c}$ is a vector over R_q having as its index set the Cartesian product $\prod_\ell \mathbb{Z}^*_{m_\ell}$, which may be flattened to the set $\mathbb{Z}^*_m$ using the bijective correspondence $(j_\ell) \leftrightarrow j = \sum_\ell (m/m_\ell) \cdot j_\ell \in \mathbb{Z}^*_m$. But for our purposes it is usually more convenient to retain the structured index set.

Working in the CRT basis yields very fast arithmetic operations. Suppose that m is a prime power. Since $c_i^2 = c_i \in R_q$ and $c_i \cdot c_{i'} = 0 \in R_q$ for distinct $i, i' \in \mathbb{Z}^*_m$, the CRT basis has the property that if $a, b \in R_q$ have coefficient vectors $\mathbf{a}, \mathbf{b}$ (respectively) over $\mathbb{Z}_q$ in the CRT basis—i.e., $a = \langle \vec{c}, \mathbf{a} \rangle$ and $b = \langle \vec{c}, \mathbf{b} \rangle$—then the coefficient vector of $a \cdot b \in R_q$ is the componentwise product $\mathbf{a} \odot \mathbf{b}$ over $\mathbb{Z}_q$. (Addition is componentwise as well, by linearity.) Moreover, this extends immediately to powers of $R^\vee$: if $\mathbf{a}, \mathbf{b}$ are the respective coefficient vectors of $a \in (R^\vee)^{k_1}_q$, $b \in (R^\vee)^{k_2}_q$ in the respective CRT bases $t^{-k_1} \cdot \vec{c}$ and $t^{-k_2} \cdot \vec{c}$, then $\mathbf{a} \odot \mathbf{b}$ is the coefficient vector of $a \cdot b \in (R^\vee)^k_q$ in the CRT basis $t^{-k} \cdot \vec{c}$, where $k = k_1 + k_2$.

Still treating m as a prime power, using the field isomorphisms $R/\mathfrak{q}_i \cong \mathbb{Z}_q$ given by $\zeta_m \mapsto \omega_m^i$, we see that the CRT basis $\vec{c}$ and powerful basis $\vec{p} = (\zeta_m^j)_{j \in [\varphi(m)]}$ of R_q are related by

$$\vec{p}^T = \vec{c}^T \cdot \mathrm{CRT}_m, \tag{5.1}$$

where the matrix CRT_m is over $\mathbb{Z}_q$. So if $a \in R_q$ has coefficient vector $\mathbf{a} \in \mathbb{Z}_q^{[\varphi(m)]}$ in the powerful basis—i.e., $a = \langle \vec{p}, \mathbf{a} \rangle$—then its coefficient vector in the CRT basis is $\mathrm{CRT}_m \cdot \mathbf{a} \in \mathbb{Z}_q^{\mathbb{Z}^*_m}$—i.e., $a = \langle \vec{c}, \mathrm{CRT}_m \cdot \mathbf{a} \rangle$—and similarly for $\mathcal{I}_q$ by linearity. Using the sparse decomposition of CRT_m and its inverse from Section 3, we can therefore switch efficiently between the power and Chinese remainder bases.

Finally, for arbitrary m, by the tensorial decomposition of R_q, multiplication is still componentwise in the CRT basis. Moreover, by the definitions of $\vec{p}$, $\vec{c}$, and CRT_m as tensor products and the mixed-product property, it immediately follows that Equation (5.1) holds as well.

6 The Decoding Basis of $R^\vee$

When working with ring-LWE we need to perform a variety of operations over $R^\vee = \langle t^{-1} \rangle$ or $R^\vee_q$. For certain operations it is best to use a certain $\mathbb{Z}$-basis of $R^\vee$ (and $\mathbb{Z}_q$-basis of $R^\vee_q$), defined below.

Let τ be the automorphism (and involution) of K that maps ζ_m to $\zeta_m^{-1} = \zeta_m^{m-1}$. We refer to τ as the conjugation map, since under the canonical embedding it corresponds to complex conjugation: $\sigma(\tau(a)) = \overline{\sigma(a)}$. Notice that for any m' dividing m, τ also maps $\zeta_{m'} = \zeta_m^{m/m'}$ to $\zeta_{m'}^{-1} = \zeta_m^{-m/m'}$. Also note that $\tau(\vec{p})$ is a $\mathbb{Z}$-basis of R, since τ is an automorphism and hence fixes R.

Definition 6.1. *The* decoding basis *of $R^\vee$ is $\vec{d} = \tau(\vec{p})^\vee$, the dual of the conjugate of the powerful basis $\vec{p}$.*[11]

The decoding basis therefore has the same index set as $\vec{p}$. When m is a prime power, $\vec{d}$ is simply the dual of the conjugate power basis $\tau(\vec{p}) = (\zeta_m^{-j})_{j \in [\varphi(m)]}$ of R. For general m, because $\tau(\vec{p})$ is the tensor product

[11] Note that unlike the powerful and CRT bases, we do not define a decoding basis for any other power of $R^\vee$; see Section 6.2 for discussion. Also, there is some flexibility in the choice of $\vec{d}$, and other definitions may be nearly as good, e.g., $\vec{d} = \vec{p}^\vee$ (without conjugation). We adopt the above definition because it corresponds to the adjoint of $\sigma(\vec{p}^T)$, and yields a particularly simple connection between $\vec{d}$ and the powerful basis $t^{-1}\vec{p}$ of $R^\vee$ (see Lemma 6.3).

of the conjugate power bases for prime-power cyclotomics R_ℓ, and $(\vec{a} \otimes \vec{b})^\vee = (\vec{a}^\vee \otimes \vec{b}^\vee)$, it follows that $\vec{d}$ is the tensor product of the decoding bases for each $R_\ell^\vee$.

We start with some basic facts about the decoding basis. Any $a \in K_\mathbb{R}$ can be represented in the decoding basis as $a = \langle \vec{d}, \mathbf{a} \rangle$ for some vector $\mathbf{a}$ of real coefficients, given by

$$a_j = \mathrm{Tr}(a \cdot d_j^\vee) = \mathrm{Tr}(a \cdot \tau(p_j)) = \langle \sigma(a), \sigma(p_j) \rangle \quad \iff \quad \mathbf{a} = \mathrm{CRT}_m^* \cdot \sigma(a). \tag{6.1}$$

Since $\vec{d}$ is the dual basis of $\tau(\vec{p})$, which embeds as $\sigma(\tau(\vec{p}^T)) = \overline{\mathrm{CRT}_m}$ over $\mathbb{C}$, we have that $\vec{d}$ embeds as

$$\sigma(\vec{d}^T) = (\mathrm{CRT}_m^*)^{-1}.$$

Lemma 4.3, and the fact that complex conjugation leaves singular values unchanged, implies the following geometric fact about the decoding basis.

Lemma 6.2. *The spectral norm of $\vec{d}$ is $s_1(\vec{d}) = \sqrt{\mathrm{rad}(m)/m}$.*

We point out that $s_1(\vec{d})$ can be as large as 1 (in the extreme case where m is square free), which, unlike for $\vec{p}$ (see Lemma 4.3), is much larger than the normalized determinant $\det(R^\vee)^{1/n} = \Delta_K^{-1/(2n)} \approx 1/\sqrt{n}$. Fortunately, the decoding basis is still always a good choice for discretizing a continuous ring-LWE error distribution (while increasing the subgaussian parameter only slightly), because the input error distribution needs to have Gaussian parameter at least $\omega(\sqrt{\log n})$ for provable worst-case hardness (see Theorem 2.22). We also point out that if $\vec{d}$ were instead defined as the dual of the *power* basis (or its conjugate), then its spectral norm could be much larger: e.g., for $m = 1155 = 3 \cdot 5 \cdot 7 \cdot 11$ we would have $s_1(\vec{d}) \approx 22.6$.

In the next few subsections, we prove several important and useful properties of the decoding basis, summarized as follows:

- There are very fast linear transformations (requiring $O(nd)$ scalar operations with small hidden constant, where d is the number of prime divisors of m) for converting between the decoding basis $\vec{d}$ and the powerful basis $t^{-1}\vec{p}$ of $R^\vee$ (see Section 6.1).
- Short elements (as always, in the sense of the canonical embedding) of K have optimally small coefficients with respect to $\vec{d}$, making it a best choice for decoding $R^\vee$. Moreover, $\vec{d}$ also yields (nearly) optimal decoding in higher powers of $R^\vee$. (See Section 6.2.)
- Continuous Gaussians (especially spherical ones) as represented in the decoding basis can be sampled very simply and efficiently (see Section 6.3).

The first fact, combined with the fast CRT transformation, means that we can efficiently convert among the decoding, power, and CRT bases of $R^\vee$ (or $R_q^\vee$) as needed. The latter two facts mean that the decoding basis is an excellent choice for generating and decoding error terms (e.g., in encryption and decryption, respectively). By contrast, the power basis and other natural bases of R or $R^\vee$ do not typically enjoy the above properties (except when m is a power of 2), and while they can in principle be used for all the same tasks, it would come at a potentially large loss in tightness and/or computational efficiency.

6.1 Relation to the Powerful Basis

Recall that both $\vec{d}$ and $t^{-1}\vec{p}$ are $\mathbb{Z}$-bases of $R^\vee$, so there is a unimodular transformation that relates them, which is given in the following lemma.

Lemma 6.3. *Let m be a power of a prime p and let $m' = m/p$, so $\varphi(m) = \varphi(p) \cdot m'$. Then*

$$\vec{d}^T = t^{-1}\vec{p}^T \cdot (L_p \otimes I_{[m']}), \tag{6.2}$$

where $L_p \in \mathbb{Z}^{[\varphi(p)]\times[\varphi(p)]}$ is the lower-triangular matrix with 1s throughout its lower-left triangle, i.e., its (i,j)th entry is 1 for $i \geq j$, and 0 otherwise.

Proof. First reindex the conjugate power basis using index set $[\varphi(p)] \times [m']$, as $\tau(p_{(j_0,j_1)}) = \zeta_p^{-j_0} \cdot \zeta_m^{-j_1}$, and reindex $\vec{d}$ similarly. Equation (6.2) may then be rewritten equivalently as

$$d_{(j_0,j_1)} = t^{-1} \cdot \left(\zeta_p^{j_0} + \zeta_p^{j_0+1} + \cdots + \zeta_p^{p-2}\right) \cdot \zeta_m^{j_1} = \tfrac{1}{m}\left(\zeta_p^{j_0} - \zeta_p^{p-1}\right) \cdot \zeta_m^{j_1}, \tag{6.3}$$

where recall from Definition 2.17 that $t^{-1} = (1-\zeta_p)/m$. To verify the above equation, observe that the product of the right-hand expression with $\tau(p_{(j_0',j_1')})$ for any $(j_0', j_1') \in [\varphi(p)] \times [m']$ is

$$\tfrac{1}{m}\left(\zeta_p^{j_0-j_0'} - \zeta_p^{p-1-j_0'}\right) \cdot \zeta_m^{j_1-j_1'}.$$

By Lemma 2.15, the trace of this is 0 if $j_1 \neq j_1'$ (because $j_1 - j_1' \neq 0 \bmod m'$); otherwise it is 0 if $j_0 \neq j_0'$ (because both $j_0 - j_0', p-1-j_0' \neq 0 \bmod p$); otherwise, it is 1, as desired. □

Observe that multiplication by L_p can be done in $O(\varphi(p))$ scalar operations via partial sums, and similarly for L_p^{-1} via successive differences. Therefore, multiplication by $L_m = (L_p \otimes I_{[m']})$ or L_m^{-1} can be done in a linear number of scalar operations. Finally, for arbitrary m having prime-power factorization $m = \prod_\ell m_\ell$, by the definitions of $\vec{p}$, $\vec{d}$, and t as tensor products and the mixed-product property, we also have

$$\vec{d}^T = t^{-1}\vec{p}^T \cdot L_m, \quad \text{where} \quad L_m = \bigotimes\nolimits_\ell L_{m_\ell}. \tag{6.4}$$

By the discussion at the end of Section 2.1, we can therefore multiply by L_m or L_m^{-1} in $O(nd)$ scalar operations, where d is the number of distinct prime divisors of m and $n = \varphi(m)$.

6.2 Decoding $R^\vee$ and Its Powers

Recall from Section 2.4.1 the "round-off" decoding procedure, which uses short linearly independent vectors in a dual lattice $\Lambda^\vee$ to recover a sufficiently short $\mathbf{x}$, given $\mathbf{x} \bmod \Lambda$. To decode $K/R^\vee$, we apply the procedure using the decoding basis $\vec{d}$ of $R^\vee$, whose dual basis in $(R^\vee)^\vee = R$ is the conjugate powerful basis $\tau(\vec{p})$. By Claim 2.10, the distance (or subgaussian parameter) that the procedure successfully decodes from depends inversely on the maximum length of the dual elements, and by Claim 4.2, every p_j in the powerful basis has $\|\tau(p_j)\|_2 = \sqrt{n}$. From this we get corresponding bounds on the decoding operation, as summarized below in Lemmas 6.5 and 6.6. We remark that the decoding basis is an optimal choice here: by Lemma 2.14, every nonzero element of R has length at least $\sqrt{n}$, hence no shorter set of dual elements exists.

In some applications (e.g., homomorphic encryption), we need to solve the more general problem of decoding $K/\mathcal{I}$, where $\mathcal{I} = (R^\vee)^k = \langle t^{-k} \rangle$ for some (usually small) $k \geq 1$. The naïve way to do this would be to apply the round-off procedure with the $\mathbb{Z}$-basis $t^{1-k}\vec{d}$ of $\mathcal{I}$. This, however, turns out to be highly suboptimal for many values of m, because the elements of the dual basis $t^{k-1}\tau(\vec{p})$ might be much longer than the shortest nonzero elements of $\mathcal{I}^\vee = \langle t^{k-1} \rangle$.[12]

[12] This can be seen already when $k = 2$ and m is a moderately large prime: using the equality $t = m/g$ and noticing that some of the embeddings of $g = 1 - \zeta_m$ are very close to zero, we see that the length of t is a rather large $\Omega(m^2)$.

Instead, in the round-off algorithm we use the *scaled decoding basis* $\hat{m}^{1-k}\vec{d}$, which generates the superideal $\mathcal{J} = \hat{m}^{1-k}R^\vee = t^{-k}g^{1-k} \supseteq \mathcal{I}$, and whose dual elements are $\hat{m}^{k-1}\tau(\vec{p}) \subset \mathcal{I}^\vee$. (Recall from Definition 2.17 that $\hat{m} = t \cdot g$ for some $g \in R$, where $\hat{m} = m/2$ if m is even and $\hat{m} = m$ otherwise.) The lengths of the dual elements are therefore $\hat{m}^{k-1}\sqrt{n}$, from which one gets the bounds summarized in Lemma 6.6 below.

We point out that the use of $\hat{m}^{1-k}\vec{d}$ for decoding $K/\mathcal{I}$ is either optimal or nearly so. Indeed, by Lemma 2.14 and Equation (2.10), the minimum distance of $\mathcal{I}^\vee = (R^\vee)^{1-k}$ is at least $\sqrt{n} \cdot \mathrm{N}(R^\vee)^{(1-k)/n} = \sqrt{n} \cdot \Delta_K^{(k-1)/n}$, so by Equation (2.8), the dual elements $\hat{m}^{k-1}\tau(\vec{p}) \subset \mathcal{I}^\vee$ are nearly as short as possible:

$$\frac{\|\hat{m}^{k-1}\tau(\vec{p})\|_2}{\lambda_1(\mathcal{I}^\vee)} = \frac{\hat{m}^{k-1}\sqrt{n}}{\lambda_1(\mathcal{I}^\vee)} \leq \Big(\prod_{\text{odd prime } p|m} p^{1/(p-1)} \Big)^{k-1},$$

which for almost all choices of m and small k is quite small. (For example, the term inside the parentheses is only ≈ 6.73 when taking all odd primes up to 17, which corresponds to $m \geq 255{,}255$.) Moreover, the above lower bound on $\lambda_1(\mathcal{I}^\vee)$ may not be tight; we suspect that in most cases of interest the minimum distance of $\mathcal{I}^\vee$ is exactly $\hat{m}^{k-1}\sqrt{n}$, which would imply that the scaled decoding basis is optimal.

We summarize the above discussion in the following definition and lemmas. As it will be more convenient for applications, we consider a "scaled up and discretized" version of the decoding procedure, where we decode from $\mathcal{I}_q$ to $\mathcal{I}$ for some $q \geq 1$. So the unknown short element is guaranteed to be in $\mathcal{I}$ but is given modulo $q\mathcal{I}$, and the output is also expected to be in $\mathcal{I}$. The only difference this makes (apart from the scaling by q) is that for $k \geq 2$, since the scaled decoding basis $\hat{m}^{1-k}\vec{d}$ may generate a strict superideal $\mathcal{J} \supset \mathcal{I}$, the round-off procedure might output an element that is not in $\mathcal{I}$. In such a case we just consider the output to be undefined. Lemmas 6.5 and 6.6 show that as long as the unknown element in $\mathcal{I}$ is short enough (or has a small enough subgaussian parameter), the decoding procedure correctly outputs it.

Definition 6.4 (Decoding $\mathcal{I}_q$ to $\mathcal{I}$). *Let $\mathcal{I} = (R^\vee)^k$ for some $k \geq 1$, and define the decoding function $[\![\cdot]\!] : \mathcal{I}_q \to \mathcal{I}$ as follows. For input $\bar{a} \in \mathcal{I}_q$, write $\bar{a} = \langle \hat{m}^{1-k}\vec{d}, \bar{\mathbf{a}} \rangle \bmod q\mathcal{J}$ for some vector $\bar{\mathbf{a}}$ over $\mathbb{Z}_q$, where $\mathcal{J} = \hat{m}^{1-k}R^\vee \supseteq \mathcal{I}$. Define $[\![\bar{a}]\!] := \langle \hat{m}^{1-k}\vec{d}, [\![\bar{\mathbf{a}}]\!] \rangle$ if this value is in $\mathcal{I}$, otherwise $[\![\bar{a}]\!]$ is undefined.*
(Recall that $[\![\bar{\mathbf{a}}]\!]$ is a vector over $\mathbb{Z}$, as defined in the beginning of Section 2.)

Lemma 6.5. *Let $\mathcal{I} = (R^\vee)^k$ for some $k \geq 1$, let $a \in \mathcal{I}$ and write $a = \langle \hat{m}^{1-k}\vec{d}, \mathbf{a} \rangle$ for some integral coefficient vector $\mathbf{a}$, and let $q \geq 1$ be an integer. If every coefficient $a_j \in [-q/2, q/2)$, then $[\![a \bmod q\mathcal{I}]\!] = a$. In particular, if every a_j is δ-subgaussian with parameter s, then $[\![a \bmod q\mathcal{I}]\!] = a$ except with probability at most $2n\exp(\delta - \pi q^2/(2s)^2)$.*

Proof. The first part is by Claim 2.10. The second part is by the tail bound on subgaussian random variables (Equation (2.2)), and the union bound. □

Lemma 6.6. *Let $\mathcal{I} = (R^\vee)^k$ for some $k \geq 1$, and let $a \in \mathcal{I}$.*

- *Writing $a = \langle \hat{m}^{1-k}\vec{d}, \mathbf{a} \rangle$ for some integral vector $\mathbf{a}$, we have that every $|a_j| \leq \hat{m}^{k-1}\sqrt{n} \cdot \|a\|_2$.*
- *If a is δ-subgaussian with parameter s, and $b \in (R^\vee)^\ell$ for some $\ell \geq 0$ is arbitrary, then writing $a \cdot b = \langle \hat{m}^{1-k-\ell}\vec{d}, \mathbf{c} \rangle$ for some integral vector $\mathbf{c}$, we have that every c_j is δ-subgaussian with parameter $\hat{m}^{k+\ell-1}\|b\|_2 \cdot s$.*

We remark that the second item above gives a bound that is a $\sqrt{n}$ factor tighter than what we would obtain by treating $a \cdot b$ as δ-subgaussian with parameter $s\|b\|_2$. The tighter bound results from using the particular properties of the powerful basis, namely, that all its elements have ℓ_∞ norm 1.

Proof. The dual elements of $\hat{m}^{1-k}\vec{d}$ are $\hat{m}^{k-1}\tau(\vec{p})$, which all have ℓ_2 norm $\hat{m}^{k-1}\sqrt{n}$. The first item then follows by the Cauchy-Schwarz inequality.

For the second item, notice that the coefficient c_j of $a \cdot b$ in the scaled decoding basis $\hat{m}^{1-k-\ell}\vec{d}$ is

$$c_j = \mathrm{Tr}(\hat{m}^{k+\ell-1}\tau(p_j) \cdot ab) = \hat{m}^{k+\ell-1}\,\mathrm{Tr}((\tau(p_j) \cdot b) \cdot a),$$

which by definition and by Claim 4.2 is δ-subgaussian with parameter

$$\hat{m}^{k+\ell-1}\|\tau(p_j) \cdot b\|_2 \cdot s \le \hat{m}^{k+\ell-1}\|\tau(p_j)\|_\infty \cdot \|b\|_2 \cdot s = \hat{m}^{k+\ell-1}\|b\|_2 \cdot s. \qquad \square$$

6.2.1 Implementation Notes

We conclude this subsection by outlining an efficient implementation of the decoding operation from Definition 6.4. As usual, we wish to use only (nearly) linear time operations, and avoid high-precision quantities. Recall that our goal is to recover an unknown element $a \in \mathcal{I}$ given $\bar{a} = a \bmod q\mathcal{I}$, where $\mathcal{I} = (R^\vee)^k$ for some $k \ge 1$. We assume that the input $\bar{a} \in \mathcal{I}_q$ is given in the form of a coefficient vector $\bar{\mathbf{a}}$ over $\mathbb{Z}_q$ satisfying $\bar{a} = \langle t^{1-k}\vec{b}, \bar{\mathbf{a}}\rangle \bmod q\mathcal{I}$, where $\vec{b}$ is some $\mathbb{Z}_q$-basis of $R_q^\vee$. The output will be given as a coefficient vector $\mathbf{a}$ over $\mathbb{Z}$ with respect to the decoding basis $t^{1-k}\vec{d}$ of $\mathcal{I}$.

The case $k = 1$ can be implemented straightforwardly. Suppose the basis $\vec{b}$ used to specify the input $\bar{a} \in R_q^\vee$ is the decoding basis, i.e., $\bar{a} = \langle \vec{d}, \bar{\mathbf{a}}\rangle \bmod qR^\vee$. We then simply output the integer coefficient vector $\mathbf{a} = [\![\bar{\mathbf{a}}]\!]$ also relative to the decoding basis, i.e., $a = \langle \vec{d}, \mathbf{a}\rangle \in R^\vee$. The number of operations is clearly linear. If the input is represented in a different basis $\vec{b}$, we first convert to the decoding basis, which is very efficient for all bases we consider.

The case $k > 1$ is more interesting, and consists of three efficient steps:

1. compute the representation of $\bar{a}' = \bar{a} \bmod q\mathcal{J}$ in the $\mathbb{Z}_q$-basis $\hat{m}^{1-k}\vec{b}$ of $\mathcal{J}_q$ (where recall that $\mathcal{J} = \hat{m}^{1-k}R^\vee \supseteq \mathcal{I}$);
2. decode it as in the case $k = 1$ to an element $a' \in \mathcal{J}$ (which will equal a if decoding was successful);
3. compute the representation of a' in the $\mathbb{Z}$-basis $t^{1-k}\vec{d}$ of $\mathcal{I}$.

We next explain each of the three steps in detail.

The first step, it turns out, is equivalent to multiplication by $g^{k-1} \in R$, where recall from Definition 2.17 that $\hat{m} = g \cdot t$. Indeed, by factoring out g^{k-1} from the modulus and both sides of the equality, we have

$$g^{k-1} \cdot \bar{a} = \langle t^{1-k}\vec{b}, \bar{\mathbf{a}}\rangle \bmod q\mathcal{I} \quad \iff \quad \bar{a} = \langle \hat{m}^{1-k}\vec{b}, \bar{\mathbf{a}}\rangle \bmod q\mathcal{J},$$

i.e., the desired coefficients of $\bar{a} \bmod q\mathcal{J}$ in basis $\hat{m}^{1-k}\vec{b}$ are exactly those of $g^{k-1}\bar{a}$ in basis $t^{1-k}\vec{b}$. Typically the input basis $\vec{b}$ at this stage would be the CRT basis, and for efficiency one could precompute the CRT coefficients of g^{k-1}, making this step linear time. In addition, multiplication by g in the powerful and decoding bases is also (nearly) linear time, as described below.

The second step is essentially identical to the case $k = 1$. Take the output $\bar{a}'$ of the first step, convert it (if needed) to a representation in the scaled decoding basis $\hat{m}^{1-k}\vec{d}$, so that $\bar{a}' = \langle \hat{m}^{1-k}\vec{d}, \bar{\mathbf{a}}'\rangle$ for some $\bar{\mathbf{a}}'$ over $\mathbb{Z}_q$, and then output the coefficient vector $[\![\bar{\mathbf{a}}']\!]$ over $\mathbb{Z}$, which represents the element $a' = \langle \hat{m}^{1-k}\vec{d}, [\![\bar{\mathbf{a}}']\!]\rangle \in \mathcal{J}$. The element a' is exactly the output of the decoding procedure as in Definition 6.4, except that it might not be in $\mathcal{I}$ (in which case decoding failed).

Finally, in the third step, we convert the representation of a' in the $\mathbb{Z}$-basis $\hat{m}^{1-k}\vec{d}$ of $\mathcal{J}$ to a representation in a $\mathbb{Z}$-basis of $\mathcal{I}$, namely $t^{1-k}\vec{d}$. This conversion might be impossible if $a' \notin \mathcal{I}$, which indicates decoding failure. Assuming $a' \in \mathcal{I}$, it is immediate to see that this conversion is equivalent to division by g^{k-1}:

$$g^{1-k} \cdot a' = \langle \hat{m}^{1-k}\vec{d}, \mathbf{a} \rangle \in \mathcal{J} \quad \iff \quad a' = \langle t^{1-k}\vec{d}, \mathbf{a} \rangle \in \mathcal{I},$$

i.e., the desired coefficients of a' in the $\mathbb{Z}$-basis $t^{1-k}\vec{d}$ of $\mathcal{I}$ are exactly those of $g^{1-k} \cdot a'$ in basis $\hat{m}^{1-k}\vec{d}$.

Division by g^{k-1} can be performed somewhat efficiently using the CRT transform over $\mathbb{C}$, but this requires $\Omega(n \log n)$ time and high-precision operations (since in contrast with the first step, here we are working with $\mathbb{Z}$-bases, and not modulo q). A better way follows from noticing that multiplication and division by g have nice forms in the decoding basis, i.e., $g \cdot \vec{d}^T = \vec{d}^T \cdot A$ for some integral matrix A that is efficient to multiply and divide by. By the tensorial decompositions of $\vec{d}$ and of g, it suffices to consider the case where m is a power of a prime p. Using Equation (6.3) and letting $m' = m/p$, one can verify that multiplication and division by $g = 1 - \zeta_p$ in the decoding basis are given, respectively, by the $[n]$-by-$[n]$ matrices

$$A = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & & \ddots & \\ & & & -1 & 1 \end{pmatrix} \otimes I_{[m']}, \quad A^{-1} = \frac{1}{p} \begin{pmatrix} 1 & 2-p & 3-p & \cdots & -1 \\ 1 & 2 & 3-p & \cdots & -1 \\ 1 & 2 & 3 & \cdots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & 3 & \cdots & p-1 \end{pmatrix} \otimes I_{[m']}.$$

It is easy to see that left-multiplication by A can be performed in time linear in n. Moreover, multiplication by A^{-1} can also be done in linear time, because every row differs from each of its adjacent rows in just one entry. Note that to avoid rational arithmetic, one would actually multiply by the integer matrix pA^{-T} and then evenly divide the result by p. If the latter step is not possible, that indicates decoding failure.

Lastly, we also note that multiplication by g in the powerful basis is given by JA^TJ, where $J = J_{[n]}$ is the $[n]$-by-$[n]$ reversal matrix, obtained by reversing the columns of the identity matrix $I_{[n]}$ (so $J = J^{-1}$ and $J_{[n]} = J_{[\varphi(p)]} \otimes J_{[m']}$). Therefore, in the powerful basis we can also multiply and divide by g in linear time per prime-power divisor of m.

6.3 Sampling Gaussians in the Decoding Basis

We now describe how to efficiently sample continuous Gaussians over $K_{\mathbb{R}}$, as represented in the decoding basis. In order to obtain the real coefficient vector $\mathbf{a}$ of some Gaussian-distributed $a \in K_{\mathbb{R}}$, by Equation (6.1) it suffices to sample $\sigma(a)$ from the continuous Gaussian distribution over H and then left-multiply by CRT^*_m. The latter step is best done using the sparse decomposition given in Section 3. Recalling the definition of H and its unitary basis matrix $B = \frac{1}{\sqrt{2}} \begin{pmatrix} I & \sqrt{-1}J \\ J & -\sqrt{-1}I \end{pmatrix} \in \mathbb{C}^{\mathbb{Z}^*_m \times [\varphi(m)]}$ from Section 2.2, we see that sampling $\sigma(a)$ amounts to sampling n independent real Gaussians used as coefficients for the columns of B, or equivalently, sampling the first $n/2$ complex coordinates as independent complex Gaussians, and completing the remaining $n/2$ coordinates using the conjugate symmetry of H.

While the above is already quite efficient, here we show that a significantly faster algorithm exists when $\mathrm{rad}(m) \ll m$. The basic idea is to notice that multiplication by the matrix CRT^*_m, with its decomposition as in Equation (3.2), starts with multiplication by two scaled unitary matrices: a (typically high-dimensional) DFT tensored with identity, and a twiddle matrix. Since spherical Gaussians are invariant under unitary transformations, we can effectively skip these two multiplications, and we only need to multiply by the (often much lower-dimensional) CRT^*_p matrices for those primes p dividing m. Details follow.

Using Equation (3.2), for any m_ℓ that is a power of a prime p_ℓ, letting $m'_\ell = m_\ell/p_\ell$ we have

$$\mathrm{CRT}^*_{m_\ell} = (\mathrm{CRT}^*_{p_\ell} \otimes I_{[m'_\ell]}) \cdot \sqrt{m'_\ell} \cdot Q_\ell$$

for some unitary Q_ℓ, because the twiddle matrix $\hat{T}_{m_\ell}$ and scaled Fourier matrix $\mathrm{DFT}_{m'_\ell}/\sqrt{m'_\ell}$ are both unitary. Therefore, by the mixed-product property we have

$$\mathrm{CRT}^*_m = \bigotimes\nolimits_\ell \mathrm{CRT}^*_{m_\ell} = \bigotimes\nolimits_\ell (\mathrm{CRT}^*_{p_\ell} \otimes I_{[m'_\ell]}) \cdot \sqrt{m/\operatorname{rad}(m)} \bigotimes\nolimits_\ell Q_\ell.$$

Since $Q = \bigotimes_\ell Q_\ell$ is unitary, it sends a spherical Gaussian distribution over $H \subset \mathbb{C}^{\mathbb{Z}^*_m}$ to a spherical Gaussian distribution (of the same parameter) over the subspace $H' = QH \subset \mathbb{C}^{\mathbb{Z}^*_m}$.[13] Therefore, to sample a continuous Gaussian of parameter s in the decoding basis, it suffices to generate a Gaussian of parameter $s\sqrt{m/\operatorname{rad}(m)}$ over H' and then left-multiply the result by

$$C^* := \bigotimes\nolimits_\ell (\mathrm{CRT}^*_{p_\ell} \otimes I_{[m'_\ell]}) = \mathrm{CRT}^*_{\mathrm{rad}(m)} \otimes I_{[m/\operatorname{rad}(m)]}.$$

The latter requires $n/\varphi(p_\ell)$ parallel applications of $\mathrm{CRT}^*_{p_\ell}$, in sequence for each ℓ, which can be done in a total of $O(n \log(\mathrm{rad}(m)))$ scalar operations.

It remains to explain how to sample a spherical Gaussian from H'. For this it suffices to give a unitary basis matrix B' of H', which allows us to generate a Gaussian over H' as $B'\mathbf{c}$, where $\mathbf{c}$ is real Gaussian. Now, observe that the subspace H' is

$$H' = \{\mathbf{x} \in \mathbb{C}^{\mathbb{Z}^*_m} : C^*\mathbf{x} \in \mathbb{R}^{[\varphi(m)]}\},$$

because H' is a real vector space of dimension n, and $C^*H' = \mathbb{R}^{[\varphi(m)]}$. So it suffices to give a unitary matrix B' such that C^*B' is real. By the mixed-product property, such a matrix is

$$B' = \bigotimes\nolimits_\ell (B'_{p_\ell} \otimes I_{[m'_\ell]}),$$

where $B'_{p_\ell} = \frac{1}{\sqrt{2}} \begin{pmatrix} I & \sqrt{-1}J \\ J & -\sqrt{-1}I \end{pmatrix}$ for $p_\ell > 2$, and is the scalar identity for $p_\ell = 2$. Clearly, multiplication by B' is a simple linear-time operation in the dimension.

Finally, we remark that because the final vector of decoding basis coefficients is $C^*B'\mathbf{c}$ for a real Gaussian $\mathbf{c}$, it is possible to generate these coefficients using just real arithmetic as $D\mathbf{c}$, where $D = \bigotimes_\ell (D_{p_\ell} \otimes I_{[m'_\ell]})$ and $D_{p_\ell} = \mathrm{CRT}^*_{p_\ell} \cdot B'_{p_\ell}$ is a real $\varphi(p_\ell)$-by-$\varphi(p_\ell)$ matrix.

7 Regularity

In this section we prove a certain "regularity lemma" that is useful in cryptographic applications, such as when adapting the "primal" [Reg05] and "dual" [GPV08] LWE-based cryptosystems, and the identity-based versions of the latter scheme, to ring-LWE. (See Section 8.1 for such an adaptation of the dual cryptosystem.) Independently, a closely related statement, specialized to power-of-2 cyclotomics, was recently shown in [SS11] with a different style of proof.

The theorem says the following. Assume we are working with the mth cyclotomic of degree $n = \varphi(m)$, and let $q \geq 1$ be a prime integer. Let $a_1, \ldots, a_{\ell-1}$ be chosen uniformly and independently from R_q. Then,

[13] Here and in what follows, we identify the index set $\mathbb{Z}^*_m$ with the set $\prod_\ell (\mathbb{Z}^*_{p_\ell} \times [m'_\ell])$ as in the decomposition of CRT_m, and similarly identify $[\varphi(m)]$ with $\prod_\ell [\varphi(m_\ell)]$.

with high probability over the choice of the a_i, the distribution of $b_0 + \sum_{i=1}^{\ell-1} b_i a_i$ is within statistical distance $2^{-\Omega(n)}$ of uniform, where the b_i are chosen from a discrete Gaussian distribution on R of width essentially $n \cdot q^{1/\ell}$ (in the canonical embedding). Equivalently, the lemma says that if a_0 is any fixed invertible element of R_q and $a_1, \ldots, a_{\ell-1}$ are uniformly and independently chosen from R_q, then $\sum_{i=0}^{\ell-1} b_i a_i$ is within $2^{-\Omega(n)}$ of uniform, where the b_i are chosen as before. The equivalence follows by simply dividing by a_0. (The lemma we prove is actually more general, and applies to the joint distribution of $k \geq 1$ sums as above; see Theorem 7.4 and Corollary 7.5 for the exact statement.)

This regularity statement is already interesting and non-trivial when ℓ is as small as 2, and is close to being tight: for instance, when m is a power of 2, a width of at least $\sqrt{n}q^{1/\ell}$ is required just for entropy reasons. To see this, recall that R is a rotation of $\sqrt{n}\mathbb{Z}^n$, so roughly speaking, a discrete Gaussian of width t covers $(t/\sqrt{n})^n$ points.

One might wonder about the significance of the b_0 term, and why we do not analyze the regularity of $\sum_{i=0}^{\ell-1} b_i a_i$ when all the a_i are chosen uniformly from R_q. In fact, a regularity lemma for exactly such sums was shown by Micciancio [Mic02]. (His work is specialized to the ring $R = \mathbb{Z}[x]/\langle x^n - 1\rangle$, but can be extended to other rings, as observed in [SSTX09].) Unfortunately, such sums have a much worse regularity property, and in particular require super-constant ℓ to get negligible distance to uniformity. To see why this is the case, assume that q is a prime satisfying $q = 1 \bmod m$, so that $\langle q \rangle$ splits completely into n ideals of norm q each. Letting $\mathfrak{q}$ denote one of these prime factors, notice that with probability $q^{-\ell}$, all the a_i are in $\mathfrak{q}$. In this case, $\sum_{i=1}^{m} b_i a_i$ is in $\mathfrak{q}$ with certainty, and its distribution is therefore very far from uniform. By adding the b_0 term we avoid this "common divisor" problem and get much better regularity, providing exponentially small distance to uniformity already for ℓ as small as 2. It is also worth mentioning that including the b_0 term (or equivalently, requiring a_0 to be uniform) corresponds to the "normal form" of ring-LWE and ring-SIS.

We start with a technical claim on the Gaussian weight on a lattice.

Claim 7.1. *For any n-dimensional lattice Λ and $\varepsilon, r > 0$,*

$$\rho_{1/r}(\Lambda) \leq \max\left(1, \left(\frac{\eta_\varepsilon(\Lambda^\vee)}{r}\right)^n\right)(1+\varepsilon).$$

Proof. For $r \geq \eta_\varepsilon(\Lambda^\vee)$, the claim follows from Definition 2.5. For $r < \eta = \eta_\varepsilon(\Lambda^\vee)$, it follows from the Poisson summation formula (see [MR04, Lemma 2.8]) that

$$\rho_{1/r}(\Lambda) = (\det \Lambda)^{-1} \cdot r^{-n} \cdot \rho_r(\Lambda^\vee) < (\det \Lambda)^{-1} \cdot r^{-n} \cdot \rho_\eta(\Lambda^\vee) = (\eta/r)^n \cdot \rho_{1/\eta}(\Lambda),$$

and the claim follows from the previous case. □

Using Lemma 2.6 and Lemma 2.14 we have

$$\eta_{2^{-2n}}(\mathcal{I}^\vee) \leq \sqrt{n}/\lambda_1(\mathcal{I}) \leq (\mathrm{N}(\mathcal{I}))^{-1/n},$$

which implies the following corollary.

Corollary 7.2. *For any ideal $\mathcal{I}$ and $r > 0$,*

$$\rho_{1/r}(\mathcal{I}) \leq \max\left(1, \mathrm{N}(\mathcal{I})^{-1} r^{-n}\right)(1 + 2^{-2n}).$$

We will also need the following algebraic claim.

Claim 7.3. *In the mth cyclotomic number field of degree n, for any $q, k \geq 1$,*

$$\sum_{\mathcal{J} | \langle q \rangle} \mathrm{N}(\mathcal{J})^k \leq \exp(c) q^{kn} \leq q^{kn+2},$$

where c is the number of distinct prime integer divisors of q.

Proof. The second inequality is clear. For the first inequality, it suffices to consider the case of a prime power $q = p^e$. Indeed, if q_1 and q_2 are coprime then

$$\sum_{\mathcal{J} | \langle q_1 q_2 \rangle} \mathrm{N}(\mathcal{J})^k = \Big(\sum_{\mathcal{J} | \langle q_1 \rangle} \mathrm{N}(\mathcal{J})^k \Big) \Big(\sum_{\mathcal{J} | \langle q_2 \rangle} \mathrm{N}(\mathcal{J})^k \Big).$$

Next, recall from Section 2.5.5 that for any integer prime p, the ideal $\langle p \rangle$ factors as $\mathfrak{p}_1^h \cdots \mathfrak{p}_g^h$ where $h = \varphi(p^d)$, $d \geq 0$ is the largest integer such that p^d divides m, each $\mathfrak{p}_i$ is of norm p^f where $f \geq 1$ is the multiplicative order of p modulo m/p^d, and $g = n/(hf)$. Therefore, $\langle q \rangle = \mathfrak{p}_1^{eh} \cdots \mathfrak{p}_g^{eh}$, and

$$\begin{aligned}
\sum_{\mathcal{J} | \langle q \rangle} \mathrm{N}(\mathcal{J})^k &= \prod_{i=1}^{g} (1 + \mathrm{N}(\mathfrak{p}_i)^k + \cdots + \mathrm{N}(\mathfrak{p}_i)^{ehk}) \\
&= \Big(1 + p^{fk} + \cdots + p^{ehfk}\Big)^g \\
&\leq p^{ehfkg} (1 - p^{-fk})^{-g} \\
&\leq q^{nk} \exp(g \cdot p^{-fk}).
\end{aligned}$$

Next, observe that p^f is greater than m/p^d (since it is greater than 1 and equals 1 modulo m/p^d) and that $g \leq n/\varphi(p^d) = \varphi(m/p^d)$, hence

$$g \cdot p^{-fk} \leq g \cdot p^{-f} \leq 1,$$

which completes the proof. □

The following is the regularity theorem. Here, for a matrix $A \in R_q^{[k] \times [\ell]}$ we define

$$\Lambda^\perp(A) = \{ \vec{z} \in R^{[\ell]} : A\vec{z} = 0 \bmod qR \},$$

which we identify with a lattice in H^ℓ. Its dual lattice (which is again a lattice in H^ℓ) is denoted by $\Lambda^\perp(A)^\vee$.

Theorem 7.4. *Let R be the ring of integers in the mth cyclotomic number field K of degree n, and $q \geq 2$ an integer. For positive integers $k \leq \ell \leq \mathrm{poly}(n)$, let $A = [I_{[k]} \mid \bar{A}] \in (R_q)^{[k] \times [\ell]}$, where $I_{[k]} \in (R_q)^{[k] \times [k]}$ is the identity matrix and $\bar{A} \in (R_q)^{[k] \times [\ell - k]}$ is uniformly random. Then for all $r > 2n$,*

$$\mathbb{E}_{\bar{A}} \Big[\rho_{1/r}\big(\Lambda^\perp(A)^\vee\big) \Big] \leq 1 + 2(r/n)^{-n\ell} q^{kn+2} + 2^{-\Omega(n)}.$$

In particular, if $r > 2n \cdot q^{k/\ell + 2/(n\ell)}$ then $\mathbb{E}_{\bar{A}}[\rho_{1/r}(\Lambda^\perp(A)^\vee)] \leq 1 + 2^{-\Omega(n)}$, and so by Markov's inequality, $\eta_{2^{-\Omega(n)}}(\Lambda^\perp(A)) \leq r$ except with probability at most $2^{-\Omega(n)}$.

Using Lemma 2.7, and the fact that A contains an identity submatrix $I_{[k]}$ and so the columns of A generate all of $R_q^{[k]}$, we obtain the following corollary, which is often more useful in applications.

Corollary 7.5. *Let R, n, q, k, and ℓ be as in Theorem 7.4. Assume that $A = [I_{[k]} \mid \bar{A}] \in (R_q)^{[k]\times[\ell]}$ is chosen as in Theorem 7.4. Then, with probability $1 - 2^{-\Omega(n)}$ over the choice of $\bar{A}$, the distribution of $A\vec{x} \in R_q^{[k]}$ where each coordinate of $\vec{x} \in R_q^{[\ell]}$ is chosen from a discrete Gaussian distribution of parameter $r > 2n \cdot q^{k/\ell+2/(n\ell)}$ over R, satisfies that the probability of each of the q^{nk} possible outcomes is in the interval $(1 \pm 2^{-\Omega(n)})q^{-nk}$ (and in particular is within statistical distance $2^{-\Omega(n)}$ of the uniform distribution over $R_q^{[k]}$).*

Proof of Theorem 7.4. Observe that for any $A \in (R_q)^{[k]\times[\ell]}$, the dual lattice of $\Lambda^\perp(A)$ is

$$\Lambda^\perp(A)^\vee = (R^\vee)^{[\ell]} + \left\{ \tfrac{1}{q} A^T \vec{s} : \vec{s} \in (R_q^\vee)^{[k]} \right\}.$$

We therefore have

$$\begin{aligned}
\mathbb{E}_{\bar{A}}\Big[\rho_{1/r}\big(\Lambda^\perp(A)^\vee\big)\Big] &= \sum_{\vec{s}\in(R_q^\vee)^{[k]}} \mathbb{E}_{\bar{A}}\Big[\rho_{1/r}\Big((R^\vee)^{[\ell]} + \tfrac{1}{q}A^T\vec{s}\Big)\Big] \\
&= \sum_{\vec{s}\in(R_q^\vee)^{[k]}} \rho_{1/r}\Big((R^\vee)^{[k]} + \tfrac{1}{q}\vec{s}\Big) \cdot \mathbb{E}_{\vec{a}}\Big[\rho_{1/r}\Big(R^\vee + \tfrac{1}{q}\langle\vec{a},\vec{s}\rangle\Big)\Big]^{\ell-k}, \qquad (7.1)
\end{aligned}$$

where $\vec{a}$ is chosen uniformly from $R_q^{[k]}$. For any $\vec{s} = (s_1, \ldots, s_k) \in (R_q^\vee)^{[k]}$, define the ideal $\mathcal{I}_{\vec{s}} = s_1R + \cdots + s_kR + qR^\vee \subseteq R^\vee$; this is the "greatest common divisor" ideal of all the s_i and $qR^\vee$. Note that $\langle\vec{a},\vec{s}\rangle$ is uniformly random over $\mathcal{I}_{\vec{s}}/qR^\vee$, and so the expectation above is

$$\big|\mathcal{I}_{\vec{s}}/qR^\vee\big|^{-1} \cdot \rho_{1/r}(\tfrac{1}{q}\mathcal{I}_{\vec{s}}).$$

Therefore, if we let T denote the set of all ideals $\mathcal{J}$ satisfying $qR^\vee \subseteq \mathcal{J} \subseteq R^\vee$, we can write (7.1) as

$$\begin{aligned}
&\sum_{\mathcal{J}\in T} |\mathcal{J}/qR^\vee|^{-(\ell-k)} \cdot \rho_{1/r}(\tfrac{1}{q}\mathcal{J})^{\ell-k} \sum_{\vec{s}\text{ s.t. }\mathcal{I}_{\vec{s}}=\mathcal{J}} \rho_{1/r}\Big((R^\vee)^{[k]} + \tfrac{1}{q}\vec{s}\Big) \\
&\quad\le \rho_{1/r}(R^\vee)^\ell + \sum_{\mathcal{J}\in T\setminus\{qR^\vee\}} |\mathcal{J}/qR^\vee|^{-(\ell-k)} \cdot \rho_{1/r}(\tfrac{1}{q}\mathcal{J})^{\ell-k} \cdot \Big(\rho_{1/r}(\tfrac{1}{q}\mathcal{J})^k - 1\Big) \\
&\quad\le \rho_{1/r}(R^\vee)^\ell + \sum_{\mathcal{J}\in T\setminus\{qR^\vee\}} |\mathcal{J}/qR^\vee|^{-(\ell-k)} \cdot \Big(\rho_{1/r}(\tfrac{1}{q}\mathcal{J})^\ell - 1\Big) \\
&\quad= 1 + \sum_{\mathcal{J}\in T} |\mathcal{J}/qR^\vee|^{-(\ell-k)} \cdot \Big(\rho_{1/r}(\tfrac{1}{q}\mathcal{J})^\ell - 1\Big), \qquad (7.2)
\end{aligned}$$

where in the first inequality we used the fact that for every $\mathcal{J} \in T \setminus \{qR^\vee\}$, the sets $(R^\vee)^{[k]} + \frac{1}{q}\vec{s}$ for all $\vec{s}$ satisfying $\mathcal{I}_{\vec{s}} = \mathcal{J}$ are disjoint, and their union is contained in $(\frac{1}{q}\mathcal{J})^{[k]} \setminus \{\mathbf{0}\}$. Next, using Corollary 7.2, we see that

$$\begin{aligned}
\rho_{1/r}(\tfrac{1}{q}\mathcal{J})^\ell &\le \max\big(1, (|\mathcal{J}/qR^\vee| \cdot \Delta_K r^{-n})^\ell\big)(1 + 2^{-2n})^\ell \\
&\le 1 + \ell 2^{1-2n} + 2(|\mathcal{J}/qR^\vee| \cdot \Delta_K r^{-n})^\ell.
\end{aligned}$$

This, together with Claim 7.3 and (2.8), allows us to bound (7.2) by

$$\begin{aligned}
&1 + 2^{-\Omega(n)} + 2\Delta_K^\ell r^{-n\ell} \sum_{\mathcal{J}\in T} |\mathcal{J}/qR^\vee|^k \\
&\quad\le 1 + 2^{-\Omega(n)} + 2(r/n)^{-n\ell} q^{kn+2},
\end{aligned}$$

and the theorem follows. □

8 Cryptosystems

Here we give three example applications of our toolkit, which all work in arbitrary cyclotomic rings:

- In Section 8.1, we give a simple adaptation of the "dual" LWE-based public-key cryptosystem of [GPV08], which uses our regularity lemma of Section 7, and which can serve as a foundation for (hierarchical) identity-based encryption;
- In Section 8.2, we give a public-key cryptosystem with more compact public keys and ciphertexts (of only two ring elements each), analogous to the ones of [LPS10, LP11];
- In Section 8.3, we describe a symmetric-key "somewhat homomorphic" cryptosystem and associated "modulus reduction" and "key switching" algorithms.

We emphasize that throughout this section, the cryptosystems and associated operations are defined almost entirely in an implementation- and basis-independent manner, using just abstract mathematical objects and operations (e.g., ring addition and multiplication, cosets of ideals and probability distributions over them, etc.). All of the operations can be performed very efficiently using the algorithms described earlier in the paper.

In particular, our cryptosystems need to sample from subgaussian distributions over cosets of $R^\vee$ (or a scaling of it). For this purpose we can use any valid discretization $\lfloor \cdot \rceil$ as described in Section 2.4.2, applied to any continuous error distribution ψ over $K_\mathbb{R}$. The choice of discretization affects only the resulting subgaussian parameter of the sample. For example, we can use the "coordinate-wise randomized rounding" method with the decoding basis $\vec{d}$ of $R^\vee$, which gives good subgaussian bounds (see Lemma 6.2).

8.1 Dual-Style Cryptosystem

In this section we present the ring-based variant of what is commonly called "dual" LWE encryption, first introduced in [GPV08] for the purposes of constructing identity-based encryption schemes. (The name "dual" refers to the fact that the system has dual properties to Regev's first LWE-based cryptosystem [Reg05], namely, the public key is statistically close to uniform, whereas ciphertexts are only pseudorandom and have unique encryption randomness.)

Let R denote the mth cyclotomic ring (of degree $n = \varphi(m)$) and let p and q be coprime integers, where p defines the message space R_p and q is the ring-LWE modulus. Let ψ be a continuous LWE error distribution over $K_\mathbb{R}$, and let $\lfloor \cdot \rceil$ denote a valid discretization to (cosets of) $R^\vee$ or $pR^\vee$. In the key-generation algorithm we need to sample from the discrete Gaussian distribution $D_{R,r}$ for some $r \geq \sqrt{n} \cdot \omega(\sqrt{\log n})$; we can do so using the algorithm from Lemma 2.9 with the powerful basis $\vec{p}$ of R, since by Claim 4.2 its (Gram-Schmidt orthogonalized) elements have maximum length $\sqrt{n}$. We also let $\ell \geq 2$ be a parameter.

The cryptosystem is defined as follows.

- Gen: choose $a_0 = -1 \in R_q$ and uniformly random and independent $a_1, \ldots, a_{\ell-1} \in R_q$, and independent $x_0, \ldots, x_{\ell-1} \leftarrow D_{R,r}$. Output $\vec{a} = (a_1, \ldots, a_{\ell-1}, a_\ell = -\sum_{i \in [\ell]} a_i x_i) \in R_q^{\{1,\ldots,\ell\}}$ as the public key, and $\vec{x} = (x_1, \ldots, x_{\ell-1}, x_\ell = 1) \in R^{\{1,\ldots,\ell\}}$ as the secret key. Note that $\langle \vec{a}, \vec{x} \rangle = x_0 \in R_q$, by construction.
- $\mathsf{Enc}_{\vec{a}}(\mu \in R_p)$: choose independent $e_0, e_1, \ldots, e_{\ell-1} \leftarrow \lfloor p \cdot \psi \rceil_{pR^\vee}$, and $e_\ell \leftarrow \lfloor p \cdot \psi \rceil_{t^{-1}\mu + pR^\vee}$. Let $\vec{e} = (e_1, \ldots, e_\ell) \in (R^\vee)^{\{1,\ldots,\ell\}}$. Output ciphertext $\vec{c} = e_0 \cdot \vec{a} + \vec{e} \in (R_q^\vee)^{\{1,\ldots,\ell\}}$.

- $\mathsf{Dec}_{\vec{x}}(\vec{c})$: compute $d = [\![\langle \vec{c}, \vec{x} \rangle]\!] \in R^\vee$ (see Definition 6.4), and output $\mu = t \cdot d \bmod pR$.

Lemma 8.1. *If $r > 2n \cdot q^{1/\ell + 2/(n\ell)}$, then the above cryptosystem is IND-CPA secure assuming the hardness of R-$\mathsf{DLWE}_{q,\psi}$ given $\ell + 1$ samples.*

Proof. By Corollary 7.5 (with $k = 1$), the public key $\vec{a}$ is within statistical distance $2^{-\Omega(n)}$ of the uniform distribution over $R_q^{\{1,\ldots,\ell\}}$. By Lemma 2.23 and Lemma 2.24, it follows that for any message μ (chosen adversarially given $\vec{a}$), the ciphertext $\vec{c} = e_0 \cdot \vec{a} + \vec{e}$ is computationally indistinguishable from uniform and independent of the public key, under the hardness assumption. □

Lemma 8.2. *Suppose that for any $c \in R_p^\vee$, $\lfloor p \cdot \psi \rceil_{c+pR^\vee}$ is δ-subgaussian with parameter s for some $\delta = O(1/\ell)$, and $q \geq s\sqrt{(r^2\ell + 1)n} \cdot \omega(\sqrt{\log n})$. Then decryption is correct with probability $1 - \mathrm{negl}(n)$ over all the randomness of key generation and encryption.*

In particular, if ψ is a continuous Gaussian with parameter $s' \geq 1$, and we use coordinate-wise randomized rounding in the decoding basis for discretization, then by the discussion in Section 2.4.2 and the equality $s_1(\vec{d}) = \sqrt{\mathrm{rad}(m)/m}$ from Lemma 6.2, we have that $\lfloor p \cdot \psi \rceil_{c+pR^\vee}$ is 0-subgaussian with parameter $s = p\sqrt{s'^2 + 2\pi\,\mathrm{rad}(m)/m} = O(ps')$.

Proof. By construction, $\langle \vec{c}, \vec{x} \rangle = e_0 x_0 + \langle \vec{e}, \vec{x} \rangle = \langle \vec{e}', \vec{x}' \rangle \bmod qR^\vee$, where $\vec{e}' = (e_0, e_1, \ldots, e_\ell) \in (R^\vee)^{[\ell+1]}$ and $\vec{x}' = (x_0, x_1, \ldots, x_\ell = 1) \in R^{[\ell+1]}$. Furthermore, $\langle \vec{e}', \vec{x}' \rangle = t^{-1}\mu \bmod pR^\vee$, so decryption is correct as long as $[\![\langle \vec{e}', \vec{x}' \rangle \bmod qR^\vee]\!] = \langle \vec{e}', \vec{x}' \rangle \in R^\vee$. We next show that this holds with probability $1 - \mathrm{negl}(n)$ over the choice of $\vec{e}', \vec{x}'$.

By Lemma 2.8, for each $i \in [\ell]$ we have $\|x_i\|_2 \leq r\sqrt{n}$ except with probability at most $2^{-n} = \mathrm{negl}(n)$, and $\|x_\ell\|_2 = \|1\|_2 = \sqrt{n}$. Then by Item 6.6 of Lemma 6.6 (with $k = 1$, $\ell = 0$), for every $i \in [\ell]$ each coefficient of $e_i x_i$ when represented in the decoding basis is δ-subgaussian with parameter $sr\sqrt{n}$, and each one of $e_\ell x_\ell$ is δ-subgaussian with parameter $s\sqrt{n}$. Since the e_i are mutually independent, each decoding-basis coefficient of $\langle \vec{e}', \vec{x}' \rangle$ is $\delta(\ell + 1)$-subgaussian with parameter $s\sqrt{(r^2\ell + 1)n}$. Since $\delta(\ell + 1) = O(1)$, the claim follows by Lemma 6.5. □

8.2 Compact Public-Key Cryptosystem

As in the previous subsection, let R denote the mth cyclotomic ring and let p, q be coprime integers, where the message space is R_p. We also require q to be coprime with every odd prime dividing m. Also let ψ be a continuous LWE error distribution over $K_\mathbb{R}$, and let $\lfloor \cdot \rceil$ denote a valid discretization to (cosets of) $R^\vee$ or $pR^\vee$. The cryptosystem is defined as follows.

- Gen: choose a uniformly random $a \leftarrow R_q$. Choose $x \leftarrow \lfloor \psi \rceil_{R^\vee}$ and $e \leftarrow \lfloor p \cdot \psi \rceil_{pR^\vee}$.

  Output $(a, b = \hat{m}(a \cdot x + e) \bmod qR) \in R_q \times R_q$ as the public key, and x as the secret key.

  (Note that because $\hat{m} = t \cdot g$, $R^\vee = \langle t^{-1} \rangle$, and $a \cdot x + e \in R^\vee / qR^\vee$, we have $\hat{m}(a \cdot x + e) \in gR/gqR$, which is then reduced mod qR to obtain $b \in R_q$.)

- $\mathsf{Enc}_{(a,b)}(\mu \in R_p)$: choose $z \leftarrow \lfloor \psi \rceil_{R^\vee}$, $e' \leftarrow \lfloor p \cdot \psi \rceil_{pR^\vee}$, and $e'' \leftarrow \lfloor p \cdot \psi \rceil_{t^{-1}\mu + pR^\vee}$.

  Let $u = \hat{m}(z \cdot a + e') \bmod qR$ and $v = z \cdot b + e'' \in R_q^\vee$. Output $(u, v) \in R_q \times R_q^\vee$.

- $\mathsf{Dec}_x(u, v)$: compute $v - u \cdot x = \hat{m}(e \cdot z - e' \cdot x) + e'' \bmod qR^\vee$, and decode it to $d = [\![v - u \cdot x]\!] \in R^\vee$ (see Definition 6.4). Output $\mu = t \cdot d \bmod pR$.

Lemma 8.3. *The above cryptosystem is IND-CPA secure assuming the hardness of* R-$\mathsf{DLWE}_{q,\psi}$.

Proof. The security proof follows from two applications of the ring-LWE assumption in its normal form (see Lemma 2.24), with secret drawn from $\lfloor \psi \rceil_{R^\vee}$. First, we claim that the public key is indistinguishable from uniform. Using the transformation from Lemma 2.23 with $w = 0$, we see that the pair $(a, a \cdot x + e) \in R_q \times R_q^\vee$, where a, x, e are sampled as in the Gen procedure, is indistinguishable from uniform. Now consider the transformation that multiplies the second component by $\hat{m}$ and reduces the result modulo qR. This transformation maps pairs $(a, a \cdot x + e)$ distributed as before, to pairs in $R_q \times R_q$ distributed as the output of the Gen procedure. Moreover, since $\langle g \rangle$ and $\langle q \rangle$ are coprime by Corollary 2.18, and recalling that $\hat{m} R^\vee = gR$, we see that this transformation maps the uniform distribution over $R_q \times R_q^\vee$ to the uniform distribution over $R_q \times R_q$. This completes the proof of the first claim.

It remains to show that if the public key (a, b) is uniformly random in $R_q \times R_q$, then for any message $\mu \in R_p$, the joint distribution of the public key together with $\mathsf{Enc}_{(a,b)}(\mu)$ is computationally indistinguishable from uniform. To see this, consider a reduction that is given access to a distribution over $R_q \times K_{\mathbb{R}}/qR^\vee$ which is either $A_{z,\psi}$ (for $z \leftarrow \lfloor \psi \rceil_{R^\vee}$) or uniform. It obtains two samples (a', u'') and (b', v') from the distribution, and applies the transformation from Lemma 2.23 with $w = 0$ to (a', u'') to obtain (a, u'), and with $w = t^{-1}\mu \in R_p^\vee$ to (b', v') to obtain (b, v). The reduction then outputs (a, b) as the public key, and $(u = \hat{m}u' \bmod qR, v) \in R_q \times R_q^\vee$ as the encryption of μ.

If the unknown distribution was uniform, then it follows that (a, b, u, v) is uniform in $R_q^{[3]} \times R_q^\vee$. (Showing that u is uniform in R_q is done as above, in the proof of the first claim.) On the other hand, if the unknown distribution is $A_{z,\psi}$, then (a, b) has uniform distribution, and it can be verified that (u, v) has the same distribution as generated by $\mathsf{Enc}_{(a,b)}(\mu)$. This completes the proof. □

We finally show that under suitable parameters, decryption is correct with overwhelming probability.

Lemma 8.4. *Suppose that* $\lfloor \psi \rceil_{R^\vee}$ *outputs elements having* ℓ_2 *norm bounded by* ℓ *with* $1 - \mathrm{negl}(n)$ *probability, that* $\lfloor p \cdot \psi \rceil_{c+pR^\vee}$ *(for any coset* $c + pR^\vee$*) is* δ*-subgaussian with parameter* s *for some* $\delta = O(1)$*, and that* $q \geq s\sqrt{2(\hat{m}\ell)^2 + n} \cdot \omega(\sqrt{\log n})$*. Then decryption is correct with probability* $1 - \mathrm{negl}(n)$ *over all the randomness of key generation and encryption.*

In particular, and just as in the previous subsection, if ψ is a continuous Gaussian with parameter $s' \geq 1$, and we use coordinate-wise randomized rounding in the decoding basis for discretization, then $\lfloor p \cdot \psi \rceil_{c+pR^\vee}$ is 0-subgaussian with parameter $s = p\sqrt{s'^2 + 2\pi \operatorname{rad}(m)/m} = O(ps')$. Moreover, by the fact that ψ has $1 - 2^{-\Omega(n)}$ of its mass on vectors of length at most $s'\sqrt{n}$, and because discretization increases lengths by at most $s_1(\vec{d})\sqrt{n}$ (by the triangle inequality), we have that $\lfloor \psi \rceil_{R^\vee}$ outputs elements having norm bounded by $\ell := (s' + \sqrt{\operatorname{rad}(m)/m})\sqrt{n} = O(s'\sqrt{n})$, except with $\mathrm{negl}(n)$ probability.

Proof. By construction, $e, e' \in pR^\vee$ and $x, z \in R^\vee$, so $\hat{m}(e \cdot z - e' \cdot x) \in pR^\vee$. Therefore, $E := \hat{m}(e \cdot z - e' \cdot x) + e'' \in R^\vee$ satisfies $E = \mu \bmod pR^\vee$ when e'' is chosen as when encrypting μ, so decryption is correct as long as $[\![E \bmod qR^\vee]\!] = E$. We next show that this holds with probability $1 - \mathrm{negl}(n)$.

By assumption, $\|x\|_2, \|z\|_2 \leq \ell$ with probability $1 - \mathrm{negl}(n)$, and e, e', and e'' are δ-subgaussian with parameter s. Then by Item 2 of Lemma 6.6 (with $k = 1$, $\ell = 0$), each coefficient of $\hat{m} \cdot ez, \hat{m} \cdot e'x \in R^\vee$ when represented in the decoding basis is δ-subgaussian with parameter $s\hat{m}\ell$, and those of e'' are δ-subgaussian with parameter $s\sqrt{n}$. Since e, e', e'' are mutually independent, each decoding-basis coefficient of E is 3δ-subgaussian with parameter $s\sqrt{2(\hat{m}\ell)^2 + n}$. The claim follows by Lemma 6.5. □

8.3 Symmetric-Key Homomorphic Cryptosystem

Here we define a symmetric-key cryptosystem that is "somewhat homomorphic," i.e., it supports limited additive and multiplicative homomorphic operations. It is essentially the Brakerski-Vaikuntanathan system [BV11b] based on ring-LWE, but with improved parameters and generalized to arbitrary cyclotomics, which introduces several technical challenges. We also describe generalized "key switching" (also known as degree reduction) and "modulus reduction" procedures akin to those first described for standard LWE in [BV11a], and for ring-LWE in power-of-2 cyclotomics in [BGV12]. (The techniques developed here can also be adapted to work with the "scale free" perspective adopted in [Bra12].) The scheme can also be made to support unbounded homomorphic operations using Gentry's "bootstrapping" technique [Gen09b, Gen09a], and also can be efficiently adapted to a public-key system using the regularity lemma from Section 7.

Description of the scheme. Let R denote the mth cyclotomic ring (of degree $n = \varphi(m)$) and let p and q be coprime integers, where p defines the message space R_p and q is the ring-LWE modulus. To support "degree reduction" (see Section 8.3.2 below), we also require $\langle p \rangle, \langle g \rangle \subseteq R$ to be coprime ideals, which is the case if and only if p is coprime with all odd primes dividing m (see Corollary 2.18).

The secret key is a ring element $s \in R$ chosen from a certain distribution (specifically, t times the LWE error distribution over $R^\vee$; see below). We say that a ciphertext of *degree* $k \geq 1$ is a polynomial $c = c(S)$ of degree at most (and usually equal to) k in an indeterminate S, having coefficients in $\mathcal{I}_q$ where $\mathcal{I} = (R^\vee)^k$. (Fresh ciphertexts produced by the encryption algorithm will have degree $k = 1$, whereas those produced by the homomorphic operations may have larger degree.) A ciphertext $c(S)$ encrypting a message $\mu \in R_p$ under secret key $s \in R$ satisfies the relation

$$c(s) = e \bmod q\mathcal{I}$$

for some sufficiently "short" $e \in \mathcal{I}$ such that $e = t^{-k} \cdot \mu \bmod p\mathcal{I}$ (where "short" can refer to the ℓ_2 norm, ℓ_∞ norm, or subgaussian parameter as needed). Therefore, given the secret key $s \in R$ one can compute $e = [\![c(s)]\!] \in \mathcal{I}$ and recover the message as $t^k \cdot e \bmod pR$. We refer to e as the "noise" in the ciphertext, and its subgaussian parameter or ℓ_2 norm determines the size of q needed to ensure correct decryption with high probability, and the underlying hardness assumption. For each operation supported by the system, we give (nearly) tight bounds on the growth or shrinkage of the noise's subgaussian parameter and ℓ_2 norm; these bounds can be combined in a modular way to calculate appropriate parameters for a particular application.

Throughout this subsection, let ψ be a continuous LWE error distribution over $K_\mathbb{R}$, and let $\lfloor \cdot \rceil$ denote any valid discretization to cosets of some scaling of $R^\vee$ (e.g., using the decoding basis $\vec{d}$ of $R^\vee$). The cryptosystem is defined formally as follows.

- Gen: choose $s' \leftarrow \lfloor \psi \rceil_{R^\vee}$, and output $s = t \cdot s' \in R$ as the secret key.
- $\mathsf{Enc}_s(\mu \in R_p)$: choose $e \leftarrow \lfloor p \cdot \psi \rceil_{t^{-1}\mu + pR^\vee}$. Let $c_0 = -c_1 \cdot s + e \in R_q^\vee$ for uniformly random $c_1 \leftarrow R_q^\vee$, and output the ciphertext $c(S) = c_0 + c_1 S$. The "noise" in $c(S)$ is defined to be e.
- $\mathsf{Dec}_s(c(S))$ for c of degree k: compute $c(s) \in (R^\vee)_q^k$, and decode it to $e = [\![c(s)]\!] \in (R^\vee)^k$. Output $\mu = t^k \cdot e \bmod pR$.

The homomorphic operations are defined as follows. For ciphertexts c, c' of arbitrary degrees k, k' (respectively), their homomorphic product is the degree-$(k + k')$ ciphertext $c(S) \boxdot c'(S) = c(S) \cdot c'(S)$ (i.e., standard polynomial multiplication). The noise in the result is defined to be the product of the noise terms of c, c'. Similarly, for ciphertexts c, c' of *equal* degree k, their homomorphic sum is defined as the degree-k ciphertext $c(S) \boxplus c'(S) = c(S) + c'(S)$, and the noise in the resulting ciphertext is the sum of those of c, c'.

(Observe that any degree-k ciphertext resulting from these operations has coefficients in $(R^\vee)_q^k$, as required.) To homomorphically add two ciphertexts of different degrees, we must first homomorphically multiply the one having smaller degree by a fixed public encryption of $1 \in R_p$ enough times to match the larger degree.[14]

It is easy to verify that if the noise terms in all the ciphertexts are correctly decoded by the decryption algorithm, then its output is correct:

$$\begin{aligned}\mathsf{Dec}_s(\mathsf{Enc}_s(\mu)) &= \mu,\\ \mathsf{Dec}_s(c \boxplus c') &= \mathsf{Dec}_s(c) + \mathsf{Dec}_s(c') \bmod pR,\\ \mathsf{Dec}_s(c \boxdot c') &= \mathsf{Dec}_s(c) \cdot \mathsf{Dec}_s(c') \bmod pR.\end{aligned}$$

The following lemma gives a sufficient condition for correct decoding to occur, and follows directly from Lemmas 6.5 and 6.6.

Lemma 8.5. *Suppose the noise e in a degree-k ciphertext c is δ-subgaussian with parameter r for some $\delta = O(1)$, and $q \geq r \cdot \hat{m}^{k-1}\sqrt{n} \cdot \omega(\sqrt{\log n})$. Then $\mathsf{Dec}_s(c)$ correctly recovers e with probability $1 - \mathrm{negl}(n)$. Alternatively, if $q > 2\|e\|_2 \hat{m}^{k-1}\sqrt{n}$, then $\mathsf{Dec}_s(c)$ recovers e with certainty.*

The next two lemmas give (nearly) tight bounds on the subgaussian parameter of the noise under the homomorphic operations. They follow directly from the definition of the noise term, the properties of subgaussian random variables (described in Section 2.3), and the triangle inequality.

Lemma 8.6. *If the noise terms in ciphertexts c_i are independent and δ_i-subgaussian with parameters r_i (respectively), then the noise in the ciphertext $\boxplus_i c_i$ is $(\sum_i \delta_i)$-subgaussian with parameter $(\sum_i r_i^2)^{1/2}$. Moreover, it is always the case that the ℓ_2 and ℓ_∞ norms of the noise terms in $\boxplus_i c_i$ are at most the sums of those in the c_i.*

Lemma 8.7. *Let e, e' be the noise terms in ciphertexts c, c', respectively. Then the noise $e \cdot e'$ in the ciphertext $c \boxdot c'$ satisfies $\|e \cdot e'\| \leq \|e\| \cdot \|e'\|_\infty$, where $\|\cdot\|$ denotes either the ℓ_2 or ℓ_∞ norm. Moreover, if e is δ-subgaussian with parameter r, then the noise $e \cdot e'$ is δ-subgaussian with parameter $r \cdot \|e'\|_\infty$. In particular, if e' is δ-subgaussian with parameter r' and is independent of e, then $e \cdot e'$ is within $\mathrm{negl}(n)$ statistical distance of a δ-subgaussian with parameter $r \cdot r' \cdot \omega(\sqrt{\log n})$.*

Proof. The first claim follows directly from Equation (2.5), and the second one by the first part of Claim 2.4. For the last claim, by subgaussianity we have $\|e'\|_\infty \leq r' \cdot \omega(\sqrt{\log n})$, except with $\mathrm{negl}(n)$ probability. □

Lemma 8.8. *The above cryptosystem is IND-CPA secure assuming the hardness of R-$\mathsf{DLWE}_{q,\psi}$.*

Proof. We describe a reduction that is given access to either an LWE distribution $A_{s',\psi}$ or the uniform distribution over $R_q \times K_\mathbb{R}/qR^\vee$. In the former case we can assume that the distribution is in normal form, i.e., the secret $s' \in R^\vee$ is distributed according to $\lfloor \psi \rceil_{R^\vee}$ (see Lemma 2.24). The reduction simulates an encryption oracle that in the former case implements the encryption algorithm Enc_s for secret key $s = t \cdot s' \in R$ (which is distributed according to the output of Gen), and in the latter case simply returns ciphertexts that are uniformly random and independent of the queried messages. This suffices to prove IND-CPA security.

To respond to an encryption query on message $\mu \in R_p$, the reduction draws a sample $(a', b') \in R_q \times K_\mathbb{R}/qR^\vee$ from the unknown distribution. It then applies the transformation from Lemma 2.23 with

[14] In particular, we can just multiply $c(S)$ by (an appropriate power of) $t^{-1} = g/\hat{m} \in R^\vee$. By definition of g, this element has ℓ_∞ norm $\|t^{-1}\|_\infty \leq 2^\ell/\hat{m} \leq 1$, where ℓ is the number of odd primes dividing m, so multiplication by t^{-1} does not increase the ℓ_2 norm or subgaussian parameter of the noise.

$w = t^{-1}\mu \in R_p^\vee$ to obtain $(a, b) \in R_q \times R_q^\vee$. It lets $c_1 = -t^{-1}a \in R_q^\vee$ and $c_0 = b$, and outputs the ciphertext $c(S) = c_0 + c_1 S$.

Suppose that the unknown distribution is the ring-LWE distribution $A_{s',\psi}$ for $s' \in R^\vee$, and let $s = t \cdot s' \in R$. Then by Lemma 2.23, the pair (a, b) is such that a is uniformly random in R_q, and $b = a \cdot s' + e = (t^{-1}a)s + e \bmod qR^\vee$, where $e \leftarrow \lfloor p \cdot \psi \rceil_{t^{-1}\mu + pR^\vee}$. Therefore, $c_1 = -t^{-1}a$ is uniformly random in $R_q^\vee$, and $c_0 = b = -c_1 \cdot s + e$, so $c(S)$ is distributed exactly according to $\mathsf{Enc}_s(\mu)$.

On the other hand, if the unknown distribution is the uniform distribution, then by Lemma 2.23 the pair (a, b) is uniformly random and independent of μ, and therefore so are the coefficients of the ciphertext $c(S)$.□

8.3.1 Modulus Reduction

The modulus reduction procedure changes the ciphertext modulus from q to some $q' < q$ (where q' is coprime with p), and outputs a ciphertext that encrypts (essentially) the same message, and whose noise term shrinks nearly proportionately. The procedure works best and is simplest to describe in the case of degree-1 ciphertexts, which can always be obtained via the key switching procedure described below in Section 8.3.2.

The following operation is central to the modulus reduction procedure. Let $\mathcal{J}$ be an ideal and let q, q', p be integers with both q and q' coprime to p. Let $v \in \mathbb{Z}_p$ be $v = q' \cdot q^{-1} \bmod p$. Define a randomized function $F_\mathcal{J} : \mathcal{J}_q \to K$ in the following way: given $x \in \mathcal{J}_q$ and some good basis of $\mathcal{J}$, sample a short (subgaussian) element from the coset $(v - q'/q) \cdot x + p\mathcal{J}$ using one of the valid methods described in Section 2.4.2, and let $F_\mathcal{J}(x)$ be the result. Note that the coset $(v - q'/q) \cdot x + p\mathcal{J}$ is well defined because $(v - q'/q)(q\mathcal{J}) = (vq - q')\mathcal{J} \subseteq p\mathcal{J}$. Also observe that for all $x \in \mathcal{J}_q$, we have $(q'/q)x + F_\mathcal{J}(x) \in \mathcal{J}_{q'}$ and $qF_\mathcal{J}(x) \in p\mathcal{J}$ with certainty.

We now describe the modulus reduction procedure. Let $c(S) = c_0 + c_1 S$ be an input ciphertext, with $c_0, c_1 \in R_q^\vee$. Let $f_0 \leftarrow F_{R^\vee}(c_0)$ and $f_1 \leftarrow t^{-1} \cdot F_R(t \cdot c_1)$, where we use coordinate-wise randomized rounding with the decoding basis $\vec{d}$ of $R^\vee$ for the former, and with the powerful basis $\vec{p}$ of R for the latter. The output is the ciphertext $c'(S) = c'_0 + c'_1 S$, where

$$c'_0 = \frac{q'}{q} c_0 + f_0 \bmod q'R^\vee, \qquad c'_1 = \frac{q'}{q} c_1 + f_1 \bmod q'R^\vee.$$

Notice that by the first of the above properties, we have $c'_0, c'_1 \in R_{q'}^\vee$ as required. Notice also that if $s = t \cdot s' \in R$ is the secret key and e is the noise in $c(S)$, so that $c_0 + c_1 s = e \bmod qR^\vee$, then

$$c'_0 + c'_1 s = \frac{q'}{q}(c_0 + c_1 s) + (f_0 + f_1 s) = \frac{q'}{q} e + (f_0 + (tf_1) \cdot s') \bmod q'R^\vee. \tag{8.1}$$

Accordingly, we define the noise in the ciphertext $c'(S)$ to be $e' = (q'/q)e + (f_0 + f_1 s)$, which is in $R^\vee$ because $c'_0, c'_1 \in R_{q'}^\vee$.

The following lemma describes the procedure's effect on the noise and plaintext. It says that the error is scaled by a factor of q'/q, plus a modulus-independent amount that depends only on the ℓ_∞ norm of $s' = t^{-1}s \in R^\vee$ (which was chosen from $\lfloor \psi \rceil_{R^\vee}$ and hence is short). It also shows that the procedure implicitly introduces a factor of $v = q' \cdot q^{-1} \in R_p$ into the message, which can be kept track of and removed upon decryption, because q' is coprime with p by assumption. In general, this extra factor seems inherent to modulus reduction, though it can be avoided by always using $q' = q \bmod p$, which always holds in the common case $p = 2$.

Lemma 8.9. *If the noise in the input ciphertext is $e \in R^\vee$, then the noise $e' \in R^\vee$ in the output ciphertext satisfies $e' = q' \cdot q^{-1} \cdot e \bmod pR^\vee$. Moreover, e' equals $(q'/q)e$ plus a random variable f that, for any value of e, is 0-subgaussian with parameter*

$$p\sqrt{2\pi}\Big(\mathrm{rad}(m)/m + \hat{m}\|t^{-1}s\|_\infty^2\Big)^{1/2},$$

and for which $\|f\|_2 \le p\sqrt{n}\Big(\sqrt{\mathrm{rad}(m)/m} + \sqrt{\hat{m}}\|t^{-1}s\|_\infty\Big)$ always.

In particular, if e is δ-subgaussian then by Claim 2.1 so is e', although it may not be independent of e.

Proof. Since both $e, e' \in R^\vee$ and q is coprime with p, showing that $e' = v \cdot e \bmod pR^\vee$ is equivalent to showing that $qe' - q'e \in pR^\vee$. The latter follows immediately from the definition of e' and the fact that $qF_{\mathcal{J}}(x) \in p\mathcal{J}$ always.

The subgaussianity claim on $e' = (q'/q)e + (f_0 + f_1 s)$ follows by the fact that for any values of c_0, c_1, the terms f_0 and tf_1 are 0-subgaussian with respective parameters $p\sqrt{2\pi}s_1(\vec{d})$ and $p\sqrt{2\pi}s_1(\vec{p})$; the bounds on $s_1(\vec{d})$ and $s_1(\vec{p})$ given in Lemmas 6.2 and 4.3 respectively; and Claim 2.1. Similarly, the claim on $\|f\|_2$ follows from the fact that coordinate-wise randomized rounding to a coset of $pR^\vee$ (respectively, pR) using basis $p \cdot \vec{d}$ (resp., $p \cdot \vec{p}$) always yields an element having ℓ_2 norm bounded by $p\sqrt{n}s_1(\vec{d})$ (resp., $p\sqrt{n}s_1(\vec{p})$); by Equation (2.5); and by the triangle inequality. □

8.3.2 Key Switching/Degree Reduction

The key-switching procedure (also known as "degree reduction") converts any degree-k ciphertext $c(S)$ encrypted under a secret key $s \in R$, to a degree-1 ciphertext $c'(S')$ encrypted under a key $s' \in R$ (which may or may not be the same as s). Notice that when decrypting $c(S)$, the evaluation $c(s)$ is simply a linear function in the powers $s^0, s^1, \ldots, s^k \in R$. The main idea behind the key-switching method introduced in [BV11a] is to homomorphically apply this linear function to suitable *encryptions* (under s') of these powers; we refer to these ciphertexts as the key-switching "hint." Implementing this idea requires some care in our setting, however, due to the different powers of $R^\vee$ involved in the operations and their homomorphic counterparts.

Rewriting the decryption relation. Let $\mathcal{I} = (R^\vee)^k$ and $d = k + 1$, let $\vec{s} = (s^0, \ldots, s^k) \in R^{[d]}$, and let $\vec{c} \in \mathcal{I}_q^{[d]}$ be the coefficient vector of a valid degree-k ciphertext $c(S)$. Then for a degree-k ciphertext c, we have the decryption relation

$$\langle \vec{c}, \vec{s} \rangle = e \bmod q\mathcal{I}$$

for some short (subgaussian) $e \in t^{-k}\mu + p\mathcal{I}$. We first put this relation in a more convenient form, viewing the ciphertext in the slightly "denser" quotient $\hat{m}^{1-k}R_q^\vee$ (because $\hat{m}^{1-k}R^\vee \supseteq \mathcal{I}$), and then scaling it up by a $\hat{m}^{k-1}$ factor.[15] We also multiply and divide $\vec{c}$ and $\vec{s}$ (respectively) by t, yielding

$$\langle \underbrace{t \cdot \hat{m}^{k-1}\vec{c}}_{\vec{y} \in R_q^{[d]}}, t^{-1}\vec{s} \rangle = \hat{m}^{k-1}e \bmod qR^\vee.$$

We write the relation in this way so that $t^{-1}\vec{s}$ is over $R^\vee$, which is the appropriate domain for encrypting it in the key-switching hint, and so that $\vec{y}$ is over R_q, which will be needed for decomposing it into short elements of R as part of the key-switching operation.

[15] This is essentially the same idea used in decoding $\mathcal{I}_q$ to $\mathcal{I}$, as described in Section 6.2.

We finally make one more important change to the decryption relation. Let $\ell = \lceil \log_2 q \rceil$ and define

$$\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{[\ell]} \quad \text{and} \quad G = I_{[d]} \otimes \mathbf{g}^T \in \mathbb{Z}_q^{[d] \times [d\ell]}. \tag{8.2}$$

Then for any $\vec{x} \in R^{[d\ell]}$ such that $G\vec{x} = \vec{y} \in R_q^{[d]}$, we have

$$\langle \vec{x}\,,\, t^{-1}G^T\vec{s} \rangle = \langle G\vec{x}\,,\, t^{-1}\vec{s} \rangle = \hat{m}^{k-1} e \bmod qR^\vee. \tag{8.3}$$

The hint will consist essentially of an encryption of $t^{-1}G^T\vec{s}$, and the key-switching operation will homomorphically compute its inner product with a *short* (subgaussian) $\vec{x}$ so as to keep the error in the resulting ciphertext small. The need for a short $\vec{x}$ is why we arranged for $\vec{y}$ to be over R_q, because we always have a good basis for R (namely, the powerful basis) that has nearly optimal spectral norm $s_1(\vec{p}) = \sqrt{\hat{m}}$, whereas we do not always have such a good basis of $\mathcal{I} = (R^\vee)^k$ for $k \geq 1$.

Alternative relations. As an optimization, we can actually omit the constant term 1 from $\vec{s}$. This decreases the dimension d by one, thereby reducing the size of the hint and the amount of extra noise introduced by the key-switching procedure. For ciphertext $c(S) = \sum_{i=0}^k c_i S^i$ we then define $\vec{c} = (c_1, \ldots, c_k)$, so that the main decryption relation becomes $c_0 + \langle \vec{c}, \vec{s} \rangle = e \bmod q\mathcal{I}$. The hint-generation and key-switching procedures then work exactly as described below, with the additional step that we add the constant term $\hat{m}^{k-1}c_0 \bmod qR^\vee$ to the output ciphertext $c'(S')$. This works because the key-switching procedure ensures that $c'(s') \approx \hat{m}^{k-1}\langle \vec{c}, \vec{s} \rangle = \hat{m}^{k-1}(e - c_0) \bmod qR^\vee$.

Similarly, when the original and target secret keys are equal, i.e., $s' = s$, we can omit both 1 and s from $\vec{s}$, define $\vec{c} = (c_2, \ldots, c_k)$, and write the decryption relation as $(c_0 + c_1 s) + \langle \vec{c}, \vec{s} \rangle = e \bmod q\mathcal{I}$. We can then apply the procedures below, adding the linear polynomial $\hat{m}^{k-1}(c_0 + c_1 S) \bmod qR^\vee$ to the output ciphertext $c'(S)$ of the key-switching procedure.

Finally, the vector $\mathbf{g}$ need not contain only powers of 2, but may be defined with respect to a larger integer base (thereby decreasing the dimension ℓ), or may even consist of other exponentially increasing sequences. The particular choice of $\mathbf{g}$ mainly affects the length (or subgaussian parameter) of the decomposition $\vec{x} \in R^{[d\ell]}$. See [MP12] for further discussion.

Constructing the hint. The hint is a collection of independent degree-1 ciphertexts $h_i(S')$ for each $i \in [d\ell]$, prepared as

$$h_i(S') \leftarrow \mathsf{Enc}_{s'}(0) + t^{-1}(G^T\vec{s})_i \bmod qR^\vee,$$

i.e., we generate degree-1 encryptions of 0 and simply add entries of $t^{-1}G^T\vec{s}$ to their constant terms. Notice that by construction,

$$h_i(s') = f_i + t^{-1}(G^T\vec{s})_i \bmod qR^\vee$$

for some short (subgaussian) $f_i \in pR^\vee$ having distribution $\lfloor p \cdot \psi \rceil_{pR^\vee}$. Note also that $h_i(S')$ may not actually be a well-formed encryption of any particular message, because $h_i(s')$ may not be congruent modulo $qR^\vee$ to any short enough element of $R^\vee$; however, this does not matter for the key-switching application.

To the vector $\vec{f} = (f_i)_{i \in [d\ell]}$ of noise terms in the hint we associate a measure of quality F, defined as

$$F := \max_{i \in \mathbb{Z}_m^*} \Big(\sum_{j=1}^{d\ell} |\sigma_i(f_j)|^2 \Big)^{1/2}, \tag{8.4}$$

and bound it as follows.

Claim 8.10. *If the entries $f_j \in R^\vee$ of $\vec{f}$ are all δ-subgaussian with parameter s for some $\delta = O(1)$, then*

$$F \le Cs \cdot \max(\sqrt{d\ell}, \omega(\sqrt{\log n}))$$

except with $\mathrm{negl}(n)$ *probability, for some universal constant $C > 0$.*

Proof. Write

$$\begin{aligned}\max_{i \in \mathbb{Z}_m^*}\Big(\sum_{j=1}^{d\ell} |\sigma_i(f_j)|^2\Big) &= \max_{i \in \mathbb{Z}_m^*}\Big(\sum_{j=1}^{d\ell} \Re(\sigma_i(f_j))^2 + \sum_{j=1}^{d\ell} \Im(\sigma_i(f_j))^2\Big) \\ &\le 2 \max_{i \in \mathbb{Z}_m^*} \max\Big\{\sum_{j=1}^{d\ell} \Re(\sigma_i(f_j))^2, \sum_{j=1}^{d\ell} \Im(\sigma_i(f_j))^2\Big\}.\end{aligned}$$

Each of the $2n$ sums is a sum of squares of $d\ell$ independent δ-subgaussian variables with parameter $s/\sqrt{2}$. The claim now follows by applying Lemma 2.2 to each of the sums and applying the union bound. □

The key-switching procedure. The procedure takes as input $\vec{c} \in \mathcal{I}_q^{[d]}$, computes $\vec{y} = t \cdot \hat{m}^{k-1}\vec{c} \in R_q^{[d]}$, and generates, as described below, a short (subgaussian) $\vec{x} \in R^{[d\ell]}$ such that $G\vec{x} = \vec{y}$. It then outputs the degree-1 ciphertext

$$c'(S') = \sum_{i \in [d\ell]} x_i \cdot h_i(S').$$

Notice that by (8.3), evaluating $c'(S')$ at $S' = s'$ gives

$$c'(s') = \sum_{i \in [d\ell]} x_i(f_i + t^{-1}(G^T\vec{s})_i) = \langle \vec{x}, \vec{f}\rangle + \langle \vec{x}, t^{-1}G^T\vec{s}\rangle = \langle \vec{x}, \vec{f}\rangle + \hat{m}^{k-1}e \bmod qR^\vee.$$

Accordingly, we define the noise term in c' to be $e' = \langle \vec{x}, \vec{f}\rangle + \hat{m}^{k-1}e \in R^\vee$. Notice that the noise is congruent to $\hat{m}^{k-1}e$ modulo $pR^\vee$, because each $f_i \in pR^\vee$ by construction of the hint. The noise is also relatively short: the $\hat{m}^{k-1}$ factor of e is exactly offset by switching from modulus $q\mathcal{I} = q(R^\vee)^k$ to $qR^\vee$, and $\langle \vec{x}, \vec{f}\rangle$ is short because both $\vec{x}$ and $\vec{f}$ are. (See Lemma 8.11 for a precise analysis.)

Also note that while decrypting the original ciphertext $c(S)$ would yield the message $t^k e = \mu \bmod pR$, the resulting degree-1 ciphertext $c'(S')$ decrypts to the message $t \cdot \hat{m}^{k-1}e = g^{k-1}\mu \bmod pR$. This means that an implementation must keep track of the "true" underlying degree of each ciphertext (and limit homomorphic additions to ciphertexts of equal "true" degree), even if its degree as a polynomial has been reduced via key switching. Upon final decryption, the extra g^{k-1} factor in the message can be removed as long as g is invertible modulo p, which by Corollary 2.18 is the case because we have assumed that p is coprime with every odd prime dividing m.

The next lemma says that the key-switching procedure introduces into the ciphertext some subgaussian error, proportional to the quality F of the noise vector $\vec{f}$ in the hint.

Lemma 8.11. *Fix an arbitrary vector $\vec{f}$ and let F be as defined in Equation* (8.4)*. Assume that for some $\delta = O(1)$, every entry $x_j \in R$ of $\vec{x}$ is δ-subgaussian with parameter s', conditioned on any values of the ciphertext c and $x_1, \ldots, x_{j-1}$. Then for any value of the original noise term e, the additional noise term $\langle \vec{x}, \vec{f}\rangle$ is $(d\ell)\delta$-subgaussian with parameter Fs'. In particular, if e is δ-subgaussian with parameter s'' then the new noise term $e' = \langle \vec{x}, \vec{f}\rangle + \hat{m}^{k-1}e$ is $(d\ell + 1)\delta$-subgaussian with parameter $\sqrt{(Fs')^2 + (\hat{m}^{k-1}s'')^2}$.*

Proof. The subgaussianity claim on $\langle \vec{x}, \vec{f}\rangle$ follows directly from Claim 2.4. The claim on e' is immediate by Claim 2.1. □

Choosing $\vec{x}$. In the key-switching procedure we need to sample a subgaussian $\vec{x} \in R^{[d\ell]}$ such that $G\vec{x} = \vec{y}$ for a given $\vec{y} \in R_q^{[d]}$, where $G \in \mathbb{Z}_q^{[d]\times[d\ell]}$ is as defined in Equation (8.2). To start, define the R-module

$$\Lambda^\perp(G) = \{\vec{z} \in R^{[d\ell]} : G\vec{z} = \vec{0} \in R_q^{[d]}\},$$

which may also be seen as a lattice in $H^{[d\ell]}$. The set of all solutions $\vec{x}$ to $G\vec{x} = \vec{y}$ is then a coset of this module. (A solution always exists, because G contains the identity $I_{[d]}$ as a submatrix.) Given a high-quality $\mathbb{Z}$-basis of $\Lambda^\perp(G)$, we can use any of the methods described in Section 2.4.2 for subgaussian sampling over the desired lattice coset, e.g., coordinate-wise randomized rounding. As usual, the relevant measures of basis quality are the largest singular value and the maximum length of the Gram-Schmidt orthogonalized vectors.

Lemma 8.12. *There is an efficiently computable $\mathbb{Z}$-basis $Z \in R^{[d\ell]\times[d\ell n]}$ of $\Lambda^\perp(G)$ satisfying the following bounds, where $\|\widetilde{Z}\|_2$ denotes the largest ℓ_2 norm of the Gram-Schmidt orthogonalized vectors $\widetilde{Z}$. If q is a power of 2, then $s_1(Z) \le 3\sqrt{\hat{m}}$ and $\|\widetilde{Z}\|_2 = 2\sqrt{n}$; otherwise, $s_1(Z) \le \sqrt{(9 + \mathrm{wt}_2(q))\hat{m}}$ and $\|\widetilde{Z}\|_2 = \sqrt{5n}$, where $\mathrm{wt}_2(q)$ denotes the number of 1s in the binary expansion of q.*

The remainder of this section is dedicated to a proof of the lemma. We first recall that Micciancio and Peikert [MP12, Section 4] constructed good bases for the *integer* lattices

$$\mathcal{L}^\perp(G) = \{\mathbf{z} \in \mathbb{Z}^{[d\ell]} : G\mathbf{z} = \mathbf{0} \in \mathbb{Z}_q^{[d]}\},$$

which we now briefly summarize (see that work for further details and full proofs). Recalling that $G = I_{[d]} \otimes \mathbf{g}^T \in \mathbb{Z}_q^{[d]\times[d\ell]}$ where $\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{[\ell]}$ and $\ell = \lceil \log_2 q \rceil$, define $S_g \in \mathbb{Z}^{[\ell]\times[\ell]}$ as

$$S_g = \begin{pmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{pmatrix} \text{ if } q = 2^\ell, \text{ otherwise } S_g = \begin{pmatrix} 2 & & & & q_0 \\ -1 & 2 & & & q_1 \\ & -1 & & & q_2 \\ & & \ddots & & \vdots \\ & & & 2 & q_{\ell-2} \\ & & & -1 & q_{\ell-1} \end{pmatrix},$$

where $q = (q_{\ell-1} \cdots q_1 q_0)_2 = \sum_{i\in[\ell]} q_i 2^i$ for $q_i \in \{0, 1\}$ is the binary representation of q. It is clear by inspection that the columns of S_g are all in the lattice $\mathcal{L}^\perp(\mathbf{g}^T)$; moreover, as shown in [MP12], they are indeed a basis of the lattice. (This can be seen by verifying that the determinants of S_g and $\mathcal{L}^\perp(\mathbf{g}^T)$ are equal.) It immediately follows that $S = I_{[d]} \otimes S_g \in \mathbb{Z}^{[d\ell]\times[d\ell]}$ is a basis for the lattice $\mathcal{L}^\perp(G)$.

In [MP12] it is shown that $\|\widetilde{S}\|_2 = \|\widetilde{S_g}\|_2 = 2$ if $q = 2^\ell$ (where we orthogonalize from right to left), and is $\sqrt{5}$ otherwise (where we orthogonalize from left to right). It also directly follows from the triangle inequality and Pythagorean theorem that $s_1(S) = s_1(S_g) \le 3$ if $q = 2^\ell$, and is at most $\sqrt{9 + \mathrm{wt}_2(q)}$ otherwise.

We now claim that

$$Z = S \otimes \vec{p}^T = I_{[d]} \otimes S_g \otimes \vec{p}^T \in R^{[d\ell]\times[d\ell n]}$$

is a $\mathbb{Z}$-basis of $\Lambda^\perp(G)$ satisfying the bounds in Lemma 8.12, where $\vec{p}$ is the powerful basis of R. For the bounds, observe that by Lemma 4.3, the fact that the longest Gram-Schmidt orthogonalized vector of $\sigma(\vec{p}^T) = \mathrm{CRT}_m$ has length $\sqrt{n}$, and the properties of singular values and orthogonalization under the tensor product, we have

$$s_1(Z) = s_1(S) \cdot s_1(\vec{p}) = s_1(S) \cdot \sqrt{\hat{m}} \quad \text{and} \quad \|\widetilde{Z}\|_2 = \|\widetilde{S}\|_2 \cdot \|\widetilde{\mathrm{CRT}_m}\|_2 = \|\widetilde{S}\|_2 \cdot \sqrt{n},$$

which when combined with the above bounds from [MP12] yields the claim. It only remains to show that Z is a $\mathbb{Z}$-basis of $\Lambda^\perp(G)$, which is a consequence of the following simple lemma.

Lemma 8.13. *Let $A \in \mathbb{Z}_q^{[h]\times[k]}$ for some $h, k \geq 1$ be arbitrary. If B is any $\mathbb{Z}$-basis of $\mathcal{L}^\perp(A) \subseteq \mathbb{Z}^{[k]}$ and $\vec{b}$ is any $\mathbb{Z}$-basis of R, then $B \otimes \vec{b}^T$ is a $\mathbb{Z}$-basis of $\Lambda^\perp(A) \subseteq R^{[k]}$.*

Proof. Clearly, every element of $B \otimes \vec{b}^T$ is in $\Lambda^\perp(A)$. To show that it is a basis, let $\vec{z} \in \Lambda^\perp(A)$ be arbitrary, so $A\vec{z} = \vec{0} \in R_q^{[h]}$. Then we can uniquely write $\vec{z} = \sum_j b_j \cdot \mathbf{z}_j$ for some vectors $\mathbf{z}_j \in \mathbb{Z}^{[k]}$. By linearity and uniqueness with respect to $\vec{b}$, this implies that $A\mathbf{z}_j = \mathbf{0} \in \mathbb{Z}_q^{[h]}$ for every j, so each $\mathbf{z}_j \in \mathcal{L}^\perp(A)$ can be written uniquely as a $\mathbb{Z}$-linear combination of elements in B. It follows that $\vec{z}$ can be expressed uniquely as a $\mathbb{Z}$-linear combination of elements in $B \otimes \vec{b}^T$. □

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, April 2011. Preliminary version in STACS 2009.

[AP13] J. Alperin-Sheriff and C. Peikert. Practical bootstrapping with polylogarithmic overhead. In *CRYPTO*. 2013. To appear.

[Bab85] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS 1985.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ICTS*, pages 309–325. 2012.

[Bos90] W. Bosma. Canonical bases for cyclotomic fields. *Appl. Algebra Eng. Commun. Comput.*, 1:125–134, 1990.

[Boy10] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517. 2010.

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. 2012.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. 2011.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012. Preliminary version in Eurocrypt 2010.

[Con09] K. Conrad. The different ideal, 2009. Available at `http://www.math.uconn.edu/~kconrad/blurbs/`, last accessed 12 Oct 2009.

[DD12] L. Ducas and A. Durmus. Ring-LWE in polynomial rings. In *Public Key Cryptography*, pages 34–51. 2012.

[Erd46] P. Erdős. On the coefficients of the cyclotomic polynomial. *Bulletin of the American Mathematical Society*, 52(2):179–184, 1946.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[Gen10] C. Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, pages 116–137. 2010.

[GHPS12] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN*, pages 19–37. 2012. Full version at `http://eprint.iacr.org/2012/240`.

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482. 2012.

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, pages 850–867. 2012.

[GLP12] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547. 2012.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[Kle00] P. N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. 2000.

[Lan94] S. Lang. *Algebraic number theory*, volume 110 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1994.

[LM06] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155. 2006.

[LM08] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54. 2008.

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72. 2008.

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339. 2011.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 2013. To appear. Preliminary version in Eurocrypt 2010.

[LPS10] V. Lyubashevsky, A. Palacio, and G. Segev. Public-key cryptographic primitives provably as secure as subset sum. In *TCC*, pages 382–400. 2010.

[Lyu09] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616. 2009.

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. 2012.

[Mic02] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[PM08] M. Püschel and J. M. F. Moura. Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs. *IEEE Transactions on Signal Processing*, 56(4):1502–1521, 2008.

[PR07] C. Peikert and A. Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, pages 478–487. 2007.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[SS11] D. Stehlé and R. Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47. 2011.

[SSTX09] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635. 2009.

[Ste04] W. Stein. A brief introduction to classical and adelic algebraic number theory, 2004. Available at `http://modular.math.washington.edu/papers/ant/`, last accessed 12 Oct 2009.

[SV11] N. Smart and F. Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011. http://eprint.iacr.org/.

[Ver11] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, January 2011. Available at http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf, last accessed 4 Feb 2011.

Classical Hardness of Learning with Errors

Zvika Brakerski* Adeline Langlois † Chris Peikert‡ Oded Regev§ Damien Stehlé ¶

Abstract

We show that the Learning with Errors (LWE) problem is *classically* at least as hard as standard worst-case lattice problems, even with polynomial modulus. Previously this was only known under *quantum* reductions.

Our techniques capture the tradeoff between the dimension and the modulus of LWE instances, leading to a much better understanding of the landscape of the problem. The proof is inspired by techniques from several recent cryptographic constructions, most notably fully homomorphic encryption schemes.

1 Introduction

Over the last decade, lattices have emerged as a very attractive foundation for cryptography. The appeal of lattice-based primitives stems from the fact that their security can be based on *worst-case* hardness assumptions, that they appear to remain secure even against *quantum* computers, that they can be quite efficient, and that, somewhat surprisingly, for certain advanced tasks such as fully homomorphic encryption no other cryptographic assumption is known to suffice.

Virtually all recent lattice-based cryptographic schemes are based directly upon one of two natural average-case problems that have been shown to enjoy worst-case hardness guarantees: the *short integer solution* (SIS) problem and the *learning with errors* (LWE) problem. The former dates back to Ajtai's groundbreaking work [Ajt96], who showed that it is at least as hard as approximating several worst-case lattice problems, such as the (decision version of the) shortest vector problem, known as GapSVP, to within a polynomial factor in the lattice dimension. This hardness result was tightened in followup work (e.g., [MR04]), leading to a somewhat satisfactory understanding of the hardness of the SIS problem. The SIS problem has been the foundation for one-way [Ajt96] and collision-resistant hash functions [GGH96], identification schemes [MV03, Lyu08, KTX08], and digital signatures [GPV08, CHKP10, Boy10, MP12, Lyu12].

*Stanford University, zvika@stanford.edu. Supported by a Simons Postdoctoral Fellowship and DARPA.

†ENS de Lyon and Laboratoire LIP (U. Lyon, CNRS, ENS Lyon, INRIA, UCBL), 46 Allée d'Italie, 69364 Lyon Cedex 07, France. adeline.langlois@ens-lyon.fr.

‡School of Computer Science, College of Computing, Georgia Institute of Technology. This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by DARPA under agreement number FA8750-11-C-0096, and by the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA or the U.S. Government, or the Sloan Foundation. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

§Courant Institute, New York University. Supported by a European Research Council (ERC) Starting Grant. Part of the work done while the author was with the CNRS, DI, ENS, Paris.

¶ENS de Lyon and Laboratoire LIP (U. Lyon, CNRS, ENS Lyon, INRIA, UCBL), 46 Allée d'Italie, 69364 Lyon Cedex 07, France. damien.stehle@ens-lyon.fr. The author was partly supported by the Australian Research Council Discovery Grant DP110100628.

Our focus in this paper is on the latter problem, learning with errors. In this problem our goal is to distinguish with some non-negligible advantage between the following two distributions:

$$((\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q))_i \quad \text{and} \quad ((\mathbf{a}_i, u_i))_i \ ,$$

where $\mathbf{s}$ is chosen uniformly from $\mathbb{Z}_q^n$ and so are the $\mathbf{a}_i \in \mathbb{Z}_q^n$, u_i are chosen uniformly from $\mathbb{Z}_q$, and the "noise" $e_i \in \mathbb{Z}$ is sampled from some distribution supported on small numbers, typically a (discrete) Gaussian distribution with standard deviation αq for $\alpha = o(1)$.

The LWE problem has proved to be amazingly versatile, serving as the basis for a multitude of cryptographic constructions: secure public-key encryption under both chosen-plaintext [Reg05, PVW08, LP11] and chosen-ciphertext [PW08, Pei09, MP12] attacks, oblivious transfer [PVW08], identity-based encryption [GPV08, CHKP10, ABB10a, ABB10b], various forms of leakage-resilient cryptography (e.g., [AGV09, ACPS09, GKPV10]), fully homomorphic encryption [BV11, BGV12, Bra12] (following the seminal work of Gentry [Gen09]), and much more. It was also used to show hardness of learning problems [KS06].

Contrary to the SIS problem, however, the hardness of LWE is not sufficiently well understood. The main hardness reduction for LWE [Reg05] is similar to the one for SIS mentioned above, except that it is *quantum*. This means that the existence of an efficient algorithm for LWE, even a classical (i.e., non-quantum) one, only implies the existence of an efficient *quantum* algorithm for lattice problems. This state of affairs is quite unsatisfactory: even though one might conjecture that efficient quantum algorithms for lattice problems do not exist, our understanding of quantum algorithms is still at its infancy. It is therefore highly desirable to come up with a *classical* hardness reduction for LWE.

Progress in this direction was made by [Pei09] (with some simplifications in the followup by Lyubashevsky and Micciancio [LM09]). The main result there is that LWE with *exponential* modulus is as hard as some standard lattice problems using a classical reduction. As that hardness result crucially relies on the exponential modulus, the open question remained as to whether LWE is hard for smaller moduli, in particular polynomial moduli. In addition to being an interesting question in its own right, this question is of special importance since many cryptographic applications, as well as the learning theory result of Klivans and Sherstov [KS06], are instantiated in this setting. Some additional evidence that reducing the modulus is a fundamental question comes from the Learning Parity with Noise (LPN) problem, which can be seen as LWE with modulus 2 (albeit with a different error distribution), and whose hardness is a long-standing open question. We remark that [Pei09] does include a classical hardness of LWE with polynomial modulus, albeit one based on a non-standard lattice problem, whose hardness is arguably as debatable as that of the LWE problem itself.

To summarize, prior to our work, the existence of an efficient algorithm for LWE with polynomial modulus was only known to imply an efficient *quantum* algorithm for lattice problems, or an efficient classical algorithm for a non-standard lattice problem. While both consequences are unlikely, they are arguably not as earth-shattering as an efficient classical algorithm for lattice problems. Hence, some concern about the hardness of LWE persisted, tainting the plethora of cryptographic applications based on it.

Main result. We provide the first classical hardness reduction of LWE with polynomial modulus. Our reduction is the first to show that the existence of an efficient classical algorithm for LWE with any subexponential modulus would indeed have earth-shattering consequences: it would imply an efficient algorithm for worst-case instances of standard lattice problems.

Theorem 1.1 (Informal). *Solving n-dimensional* LWE *with $\mathrm{poly}(n)$ modulus implies an equally efficient solution to a worst-case lattice problem in dimension $\sqrt{n}$.*

As a result, we establish the hardness of all known applications of polynomial-modulus LWE based on classical worst-case lattice problems, previously only known under a quantum assumption.

Techniques. Even though our main theorem has the flavor of a statement in computational complexity, its proof crucially relies on a host of ideas coming from recent progress in cryptography, most notably recent breakthroughs in the construction of fully homomorphic encryption schemes.

At a high level, our main theorem is a "modulus reduction" result: we show a reduction from LWE with large modulus q and dimension n to LWE with (small) modulus $p = \mathrm{poly}(n)$ and dimension $n \log_2 q$. Theorem 1.1 now follows from the main result in [Pei09], which shows that the former problem with $q = 2^n$ is as hard as n-dimensional GapSVP. We note that the increase in dimension from n to $n \log_2 q$ is to be expected, as it essentially preserves the number of possible secrets (and hence the running time of the naive brute-force algorithm).

Very roughly speaking, the main idea in modulus reduction is to map $\mathbb{Z}_q$ into $\mathbb{Z}_p$ through the naive mapping that sends any $a \in \{0, \ldots, q-1\}$ to $\lfloor pa/q \rceil \in \{0, \ldots, p-1\}$. This basic idea is confounded by two issues. The first is that if carried out naively, this transformation introduces rounding artifacts into LWE, ruining the distribution of the output. We resolve this issue by using a more careful Gaussian randomized rounding procedure (Section 3). A second serious issue is that in order for the rounding errors not to be amplified when multiplied by the LWE secret $\mathbf{s}$, it is essential to assume that $\mathbf{s}$ has small coordinates. A major part of our reduction (Section 4) is therefore dedicated to showing a reduction from LWE (in dimension n) with arbitrary secret in $\mathbb{Z}_q^n$ to LWE (in dimension $n \log_2 q$) with a secret chosen uniformly over $\{0, 1\}$. This follows from a careful hybrid argument (Section 4.3) combined with a hardness reduction to the so-called "extended-LWE" problem, which is a variant of LWE in which we have some control over the error vector (Section 4.2).

We stress that even though our proof is inspired by and has analogues in the cryptographic literature, the details of the reductions are very different. In particular, the idea of modulus reduction plays a key role in recent work on fully homomorphic encryption schemes, giving a way to control the noise growth during homomorphic operations [BV11, BGV12, Bra12]. However, since the goal there is merely to preserve the functionality of the scheme, their modulus reduction can be performed in a rather naive way similar to the one outlined above, and so the output of their procedure does not constitute a valid LWE instance. In our reduction we need to perform a much more delicate modulus reduction, which we do using Gaussian randomized rounding, as mentioned above.

The idea of reducing LWE to have a $\{0, 1\}$ secret also exists already in the cryptographic literature: precisely such a reduction was shown by Goldwasser et al. [GKPV10] who were motivated by questions in leakage-resilient cryptography. Their reduction, however, incurred a severe blow-up in the noise rate, making it useless for our purposes. In more detail, not being able to faithfully reproduce the LWE distribution in the output, they resort to hiding the faults in the output distribution under a huge independent fresh noise, in order to make it close to the correct one. The trouble with this "noise flooding" approach is that the amount of noise one has to add depends on the running time of the algorithm solving the target $\{0, 1\}$-LWE problem, which in turn forces the modulus to be equally big. So while in principle we could use the reduction from [GKPV10] (and shorten our proof by about a half), this would lead to a qualitatively much weaker result: the modulus and the approximation ratio for the worst-case lattice problem would both grow with the running time of the $\{0, 1\}$-LWE algorithm. In particular, we would not be able to show that for some fixed polynomial modulus, LWE is a hard problem; instead, in order to capture all polynomial time algorithms, we would have to take a super-polynomial modulus, and rely on the hardness of worst-case lattice problem to within super-polynomial approximation factors. In contrast, with our reduction, the

modulus and the approximation ratio both remain fixed independently of the target $\{0,1\}$-LWE algorithm.

As mentioned above, our alternative to the reduction in [GKPV10] is based on a hybrid argument combined with a new hardness reduction for the "extended LWE" problem, which is a variant of LWE in which in addition to the LWE samples, we also get to see the inner product of the vector of error terms with a vector $\mathbf{z}$ of our choosing. This problem has its origins in the cryptographic literature, namely in the work of O'Neill, Peikert, and Waters [OPW11] on (bi)deniable encryption and the later work of Alperin-Sheriff and Peikert [AP12] on key-dependent message security. The hardness reductions included in those papers are not sufficient for our purposes, as they cannot handle large moduli or error terms, which is crucial in our setting. We therefore provide an alternative reduction which is conceptually much simpler, and essentially subsumes both previous reductions. Our reduction works equally well with exponential moduli and correspondingly long error vectors, a case earlier reductions could not handle.

Broader perspective. As a byproduct of the proof of Theorem 1.1, we obtain several results that shed new light on the hardness of LWE. Most notably, our modulus reduction result in Section 3 is actually far more general, and can be used to show a "modulus expansion/dimension reduction" tradeoff. Namely, it shows a reduction from LWE in dimension n and modulus p to LWE in dimension n/k and modulus p^k (see Corollary 3.4). Combined with our modulus reduction, this has the following interesting consequence: the hardness of n-dimensional LWE with modulus q is a function of the quantity $n \log_2 q$. In other words, varying n and q individually while keeping $n \log_2 q$ fixed essentially preserves the hardness of LWE.

Although we find this statement quite natural (since $n \log_2 q$ represents the number of bits in the secret), it has some surprising consequences. One is that n-dimensional LWE with modulus 2^n is essentially as hard as n^2-dimensional LWE with polynomial modulus. As a result, n-dimensional LWE with modulus 2^n, which was shown in [Pei09] to be as hard as n-dimensional lattice problems using a classical reduction, is actually as hard as n^2-dimensional lattice problems using a quantum reduction. The latter is presumably a much harder problem, requiring $\exp(\widetilde{\Omega}(n^2))$ time to solve. This corollary highlights an inherent quadratic loss in the classical reduction of [Pei09] (and as a result also our Theorem 1.1) compared to the quantum one in [Reg05].

A second interesting consequence is that 1-dimensional LWE with modulus 2^n is essentially as hard as n-dimensional LWE with polynomial modulus. The 1-dimensional version of LWE is closely related to the Hidden Number Problem of Boneh and Venkatesan [BV96]. It is also essentially equivalent to the Ajtai-Dwork-type [AD97] cryptosystem in [Reg03], as follows from simple reductions similar to the one in the appendix of [Reg10a]. Moreover, the 1-dimensional version can be seen as a special case of the Ring-LWE problem introduced in [LPR10] (for ring dimension 1, i.e., ring equal to $\mathbb{Z}$). This allows us, via the ring switching technique from [GHPS12], to obtain the first hardness proof of Ring-LWE, with arbitrary ring dimension and exponential modulus, under the hardness of problems on general lattices (as opposed to just ideal lattice problems). In addition, this leads to the first hardness proof for the Ring-SIS problem [LM06, PR06] with exponential modulus under the hardness of general lattice problems, via the standard LWE-to-SIS reduction. (We note that since both results are obtained by scaling up from a ring of dimension 1, the hardness does not improve as the ring dimension increases.)

A final interesting consequence of our reductions is that (the decision form of) LWE is hard with an arbitrary huge modulus, e.g., a prime; see Corollary 3.3. Previous results (e.g., [Reg05, Pei09, MM11, MP12]) required the modulus to be *smooth*, i.e., all its prime divisors had to be polynomially bounded.

Open questions. As mentioned above, our Theorem 1.1 inherits from [Pei09] a quadratic loss in the dimension, which does not exist in the quantum reduction [Reg05] nor in the known hardness reductions

for SIS. At a technical level, this quadratic loss stems from the fact that the reduction in [Pei09] is not iterative. In contrast, the quantum reduction in [Reg05] as well as the reductions for SIS are iterative, and as a result do not incur the quadratic loss. We note that an additional side effect of the non-iterative reduction is that the hardness in Theorem 1.1 and [Pei09] is based only on the worst-case lattice problem GapSVP (and the essentially equivalent BDD and uSVP [LM09]), and not on problems like SIVP, which the quantum reduction of [Reg05] and the hardness reductions for SIS can handle. One case where this is very significant is when dealing with ideal lattices, as in the hardness reduction for Ring-LWE, since GapSVP turns out to be an easy problem there.

We therefore believe that it is important to understand whether there exists a classical reduction that does not incur the quadratic loss inherent in [Pei09] and in Theorem 1.1. In other words, is n-dimensional LWE with polynomial modulus classically as hard as n-dimensional lattice problems (as opposed to $\sqrt{n}$-dimensional)? This would constitute the first full dequantization of the quantum reduction in [Reg05].

While it is natural to conjecture that the answer to this question is positive, a negative answer would be quite tantalizing. In particular, it is conceivable that there exists a (classical) algorithm for LWE with polynomial modulus running in time $2^{O(\sqrt{n})}$. Due to the quadratic expansion in Theorem 1.1, this would not lead to a faster classical algorithm for lattice problems; it would, however, lead to a $2^{O(\sqrt{n})}$-time *quantum* algorithm for lattice problems using the reduction in [Reg05]. The latter would be a major progress in quantum algorithms, yet is not entirely unreasonable; in fact, a $2^{O(\sqrt{n})}$-time quantum algorithm for a somewhat related quantum task was discovered by Kuperberg [Kup05] (see also [Reg02]).

2 Preliminaries

Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ denote the cycle, i.e., the additive group of reals modulo 1. We also denote by $\mathbb{T}_q$ its cyclic subgroup of order q, i.e., the subgroup given by $\{0, 1/q, \ldots, (q-1)/q\}$.

For two probability distributions P, Q over some discrete domain, we define their statistical distance as $\sum |P(i) - Q(i)|/2$ where i ranges over the distribution domain, and extend this to continuous distributions in the obvious way. We recall the following easy fact (see, e.g., [AD87, Eq. (2.3)] for a proof).

Claim 2.1. *If P and Q are two probability distributions such that $P(i) \geq (1-\varepsilon)Q(i)$ holds for all i, then the statistical distance between P and Q is at most ε.*

We will use the following immediate corollary of the leftover hash lemma [HILL99].

Lemma 2.2. *Let $k, n, q \geq 1$ be integers, and $\varepsilon > 0$ be such that $n \geq k \log_2 q + 2\log_2(1/\varepsilon)$. For $\mathbf{H} \leftarrow \mathbb{T}_q^{k \times n}$, $\mathbf{z} \leftarrow \{0,1\}^n$, $\mathbf{u} \leftarrow \mathbb{T}_q^k$, the distributions of $(\mathbf{H}, \mathbf{Hz})$ and $(\mathbf{H}, \mathbf{u})$ are within statistical distance at most ε.*

A *distinguishing problem* P is defined by two distributions P_0 and P_1, and a solution to the problem is the ability to distinguish between these distributions. The *advantage* of an algorithm $\mathcal{A}$ with binary output on P is defined as

$$\mathrm{Adv}[\mathcal{A}] = |\Pr[\mathcal{A}(P_0)] - \Pr[\mathcal{A}(P_1)]| \,.$$

A reduction from a problem P to a problem Q is an efficient (i.e., polynomial-time) algorithm $\mathcal{A}^{\mathcal{B}}$ that solves P given access to an oracle $\mathcal{B}$ that solves Q. Most of our reductions (in fact all except the one in Lemma 2.15) are what we call "transformation reductions:" these reductions perform some transformation to the input and then apply the oracle to the result.

2.1 Lattices

An n-dimensional (full-rank) lattice $\Lambda \subseteq \mathbb{R}^n$ is the set of all integer linear combinations of some set of n linearly independent *basis* vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq \mathbb{R}^n$,

$$\Lambda = \mathcal{L}(\mathbf{B}) = \Big\{ \sum_{i \in [n]} z_i \mathbf{b}_i \ : \ \mathbf{z} \in \mathbb{Z}^n \Big\}.$$

The *dual lattice* of $\Lambda \subset \mathbb{R}^n$ is defined as $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \langle \Lambda, \mathbf{x} \rangle \subseteq \mathbb{Z}\}$.

The *minimum distance* (or *first successive minimum*) $\lambda_1(\Lambda)$ of a lattice Λ is the length of a shortest nonzero lattice vector, i.e., $\lambda_1(\Lambda) = \min_{\mathbf{0} \neq \mathbf{x} \in \Lambda} \|\mathbf{x}\|$. For an approximation ratio $\gamma = \gamma(n) \geq 1$, the GapSVP_γ is the problem of deciding, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$ and a number d, between the case where $\lambda_1(\mathcal{L}(\mathbf{B})) \leq d$ and the case where $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d$. We refer to [Kho10, Reg10b] for a recent account on the computational complexity of GapSVP_γ.

2.2 Gaussian measures

For $r > 0$, the n-dimensional Gaussian function $\rho_r : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho_r(\mathbf{x}) := \exp(-\pi \|\mathbf{x}\|^2 / r^2).$$

We extend this definition to sets, i.e., $\rho_r(A) = \sum_{\mathbf{x} \in A} \rho_r(\mathbf{x}) \in [0, +\infty]$ for any $A \subseteq \mathbb{R}^n$. The (spherical) continuous Gaussian distribution D_r is the distribution with density function proportional to ρ_r. More generally, for a matrix $\mathbf{B}$, we denote by $D_{\mathbf{B}}$ the distribution of $\mathbf{Bx}$ where $\mathbf{x}$ is sampled from D_1. When $\mathbf{B}$ is nonsingular, its probability density function is proportional to

$$\exp(-\pi \mathbf{x}^T (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{x}).$$

A basic fact is that for any matrices $\mathbf{B}_1, \mathbf{B}_2$, the sum of a sample from $D_{\mathbf{B}_1}$ and an independent sample from $D_{\mathbf{B}_2}$ is distributed like $D_{\mathbf{C}}$ for $\mathbf{C} = (\mathbf{B}_1 \mathbf{B}_1^T + \mathbf{B}_2 \mathbf{B}_2^T)^{1/2}$.

For an n-dimensional lattice Λ and a vector $\mathbf{u} \in \mathbb{R}^n$, we define the *discrete Gaussian distribution* $D_{\Lambda + \mathbf{u}, r}$ as the discrete distribution with support on the coset $\Lambda + \mathbf{u}$ whose probability mass function is proportional to ρ_r. There exists an efficient procedure that samples within negligible statistical distance of any (not too narrow) discrete Gaussian distribution ([GPV08, Theorem 4.1]; see also [Pei10]). In the next lemma, proved in Section 5, we modify this sampler so that the output is distributed exactly as a discrete Gaussian. This also allows us to sample from slightly narrower Gaussians. Strictly speaking, the lemma is not needed for our results, and we could use instead the original sampler from [GPV08]. Using our exact sampler leads to slightly cleaner proofs as well as a (miniscule) improvement in the parameters of our reductions, and we include it here mainly in the hope that it finds further applications in the future.

Lemma 2.3. *There is a probabilistic polynomial-time algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$, $\mathbf{c} \in \mathbb{R}^n$, and a parameter $r \geq \|\widetilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n + 4)/\pi}$, outputs a sample distributed according to $D_{\Lambda + \mathbf{c}, r}$.*

Here, $\widetilde{\mathbf{B}}$ denotes the Gram-Schmidt orthogonalization of $\mathbf{B}$, and $\|\widetilde{\mathbf{B}}\|$ is the length of the longest vector in it. We recall the definition of the *smoothing parameter* from [MR04].

Definition 2.4. *For a lattice Λ and positive real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.*

Lemma 2.5 ([GPV08, Lemma 3.1]). *For any $\varepsilon > 0$ and n-dimensional lattice Λ with basis $\mathbf{B}$,*

$$\eta_\varepsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \sqrt{\ln(2n(1+1/\varepsilon))/\pi}.$$

We now collect some known facts on Gaussian distributions and lattices.

Lemma 2.6 ([MR04, Lemma 4.1]). *For any n-dimensional lattice Λ, $\varepsilon > 0$, $r \geq \eta_\varepsilon(\Lambda)$, the distribution of $\mathbf{x} \bmod \Lambda$ where $\mathbf{x} \leftarrow D_r$ is within statistical distance $\varepsilon/2$ of the uniform distribution on cosets of Λ.*

Lemma 2.7 ([Reg05, Claim 3.8]). *For any n-dimensional lattice Λ, $\varepsilon > 0$, $r \geq \eta_\varepsilon(\Lambda)$, and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_r(\Lambda + \mathbf{c}) \in [\frac{1-\varepsilon}{1+\varepsilon}, 1] \cdot \rho_r(\Lambda)$.*

Lemma 2.8 ([Reg05, Claim 3.9]). *Let Λ be an n-dimensional lattice, let $\mathbf{u} \in \mathbb{R}^n$ be arbitrary, let $r, s > 0$ and let $t = \sqrt{r^2 + s^2}$. Assume that $rs/t = 1/\sqrt{1/r^2 + 1/s^2} \geq \eta_\varepsilon(\Lambda)$ for some $\varepsilon < 1/2$. Consider the continuous distribution Y on $\mathbb{R}^n$ obtained by sampling from $D_{\Lambda+\mathbf{u},r}$ and then adding a noise vector taken from D_s. Then, the statistical distance between Y and D_t is at most 4ε.*

Lemma 2.9 ([Reg05, Corollary 3.10]). *Let Λ be an n-dimensional lattice, let $\mathbf{u}, \mathbf{z} \in \mathbb{R}^n$ be arbitrary, and let $r, \alpha > 0$. Assume that $(1/r^2 + (\|\mathbf{z}\|/\alpha)^2)^{-1/2} \geq \eta_\varepsilon(\Lambda)$ for some $\varepsilon < 1/2$. Then the distribution of $\langle \mathbf{z}, \mathbf{v} \rangle + e$ where $\mathbf{v} \leftarrow D_{\Lambda+\mathbf{u},r}$ and $e \leftarrow D_\alpha$, is within statistical distance 4ε of D_β for $\beta = \sqrt{(r\|\mathbf{z}\|)^2 + \alpha^2}$.*

Lemma 2.10 (Special case of [Pei10, Theorem 3.1]). *Let Λ be a lattice and $r, s > 0$ be such that $s \geq \eta_\varepsilon(\Lambda)$ for some $\varepsilon \leq 1/2$. Then if we choose $\mathbf{x}$ from the continuous Gaussian D_r and then choose $\mathbf{y}$ from the discrete Gaussian $D_{\Lambda-\mathbf{x},s}$ then $\mathbf{x}+\mathbf{y}$ is within statistical distance 8ε of the discrete Gaussian $D_{\Lambda,(r^2+s^2)^{1/2}}$.*

2.3 Learning with Errors

For integers $n, q \geq 1$, an integer vector $\mathbf{s} \in \mathbb{Z}^n$, and a probability distribution ϕ on $\mathbb{R}$, let $A_{q,\mathbf{s},\phi}$ be the distribution over $\mathbb{T}_q^n \times \mathbb{T}$ obtained by choosing $\mathbf{a} \in \mathbb{T}_q^n$ uniformly at random and an error term e from ϕ, and outputting the pair $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{T}_q^n \times \mathbb{T}$.

Definition 2.11. *For integers $n, q \geq 1$, an error distribution ϕ over $\mathbb{R}$, and a distribution $\mathcal{D}$ over $\mathbb{Z}^n$, the (average-case) decision variant of the* LWE *problem, denoted* $\mathsf{LWE}_{n,q,\phi}(\mathcal{D})$*, is to distinguish given arbitrarily many independent samples, the uniform distribution over $\mathbb{T}_q^n \times \mathbb{T}$ from $A_{q,\mathbf{s},\phi}$ for a fixed $\mathbf{s}$ sampled from $\mathcal{D}$. The variant where the algorithm only gets a bounded number of samples $m \in \mathbb{N}$ is denoted* $\mathsf{LWE}_{n,m,q,\phi}(\mathcal{D})$.

Notice that the distribution $A_{q,\mathbf{s},\phi}$ only depends on $\mathbf{s} \bmod q$, and so one can assume without loss of generality that $\mathbf{s} \in \{0, \ldots, q-1\}^n$. Moreover, using a standard random self-reduction, for any distribution over secrets $\mathcal{D}$, one can reduce $\mathsf{LWE}_{n,q,\phi}(\mathcal{D})$ to $\mathsf{LWE}_{n,q,\phi}(U(\{0, \ldots, q-1\}^n))$, and we will occasionally use $\mathsf{LWE}_{n,q,\phi}$ to denote the latter (as is common in previous work). When the noise is a Gaussian with parameter $\alpha > 0$, i.e., $\phi = D_\alpha$, we use the shorthand $\mathsf{LWE}_{n,q,\alpha}(\mathcal{D})$. Since the case when $\mathcal{D}$ is uniform over $\{0,1\}^n$ plays an important role in this paper, we will denote it by $\mathsf{binLWE}_{n,q,\phi}$ (and by $\mathsf{binLWE}_{n,m,q,\phi}$ when the algorithm only gets m samples). Finally, as we show in the following lemma, one can efficiently reduce LWE to the case in which the secret is distributed according to the (discretized) error distribution and is hence somewhat short. This latter form of LWE, known as the "normal form," was first shown hard in [ACPS09] for the case of prime q. Here we observe that the proof extends to non-prime q, the new technical ingredient being Claim 2.13 below.

Lemma 2.12. *For any $q \geq 25$, $n, m \geq 1$, $\alpha > 0$, $\varepsilon < 1/2$ and $s \geq \sqrt{\ln(2n(1+1/\varepsilon)/\pi)}/q$, there is an efficient (transformation) reduction from $\mathsf{LWE}_{n,m,q,\alpha}$ to $\mathsf{LWE}_{n,m',q,\alpha}(\mathcal{D})$ where $m' = m - (16n + 4\ln\ln q)$ and $\mathcal{D} = D_{\mathbb{Z}^n, q(\alpha^2+s^2)^{1/2}}$, that turns advantage ζ into an advantage of at least $(\zeta - 8\varepsilon)/4$. In particular, assuming $\alpha \geq \sqrt{\ln(2n(1+1/\varepsilon)/\pi)}/q$, we can take $s = \alpha$, in which case $\mathcal{D} = D_{\mathbb{Z}^n, \sqrt{2}q\alpha}$.*

Proof. Consider the first $16n + 4\ln\ln q$ samples $(\mathbf{a}, b)$. Using Claim 2.13, with probability at least $1 - 2e^{-1} \geq 1/4$, we can efficiently find a subsequence of the samples such that the matrix $A_0 \in \mathbb{Z}_q^{n\times n}$ whose columns are formed by the $\mathbf{a}$ in the subset (scaled up by q) has an inverse $A_0^{-1} \in \mathbb{Z}_q^{n\times n}$ modulo q. If we cannot find such a subsequence, we abort. Let $\mathbf{b}_0 \in \mathbb{T}^n$ be the vector formed by the corresponding b in the subsequence. Let also $\mathbf{b}'_0 \in \mathbb{T}_q^n$ be $\mathbf{b}_0 + \mathbf{x}$ where $\mathbf{x}$ is chosen from $D_{q^{-1}\mathbb{Z}^n - \mathbf{b}_0, s}$. (Notice that the coset $q^{-1}\mathbb{Z}^n - \mathbf{b}_0$ is well defined because $\mathbf{b}_0$ is a coset of $\mathbb{Z}^n \subseteq q^{-1}\mathbb{Z}^n$.) From each of the remaining m' samples $(\mathbf{a}, b) \in \mathbb{T}_q^n \times \mathbb{T}$ we produce a pair

$$\left(\mathbf{a}' = A_0^{-1}\mathbf{a}, b' = b - \langle A_0^{-1} \cdot q\mathbf{a}, \mathbf{b}'_0\rangle\right) \in \mathbb{T}_q^n \times \mathbb{T}.$$

We then apply the given LWE oracle to the resulting m' pairs and output its result.

We now analyze the reduction. First notice that the construction of A_0 depends only on the $\mathbf{a}$ component of the input samples, and hence the probability of finding it is the same in case the input is uniform and in case it consists of LWE samples. It therefore suffices in the following to show that there is a distinguishing gap conditioned on successfully finding an A_0. To that end, first observe that if the input samples $(\mathbf{a}, b)$ are uniform in $\mathbb{T}_q^n \times \mathbb{T}$ then so are the output samples $(\mathbf{a}', b')$. Next consider the case that the input samples are distributed according to $A_{q,\mathbf{s},D_\alpha}$ for some $\mathbf{s} \in \mathbb{Z}^n$. Then since $s \geq \eta_\varepsilon(q^{-1}\mathbb{Z})$ by Lemma 2.5, using Lemma 2.10 we get that $\mathbf{b}'_0 = q^{-1}A_0^T\mathbf{s} + \mathbf{e}_0$ where $\mathbf{e}_0$ is distributed within statistical distance 8ε from $D_{q^{-1}\mathbb{Z}^n, (\alpha^2+s^2)^{1/2}}$. Therefore, for each output sample $(\mathbf{a}', b')$ we have

$$b' = b - \langle A_0^{-1} \cdot q\mathbf{a}, \mathbf{b}'_0\rangle = \langle \mathbf{a}, \mathbf{s}\rangle + e - \langle \mathbf{a}, \mathbf{s}\rangle - \langle A_0^{-1}q\mathbf{a}, \mathbf{e}_0\rangle = \langle -q\mathbf{e}_0, \mathbf{a}'\rangle + e,$$

where e is an independent error from D_α. Therefore, the output samples are distributed according to $A_{q,-q\mathbf{e}_0,D_\alpha}$, completing the proof. □

Claim 2.13. *For any $q \geq 25$, $n \geq 1$, and $t_1 \geq 4, t_2 \geq 1$, given a sequence of $t_1 n + t_2 \ln\ln q$ vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots$ chosen uniformly and independently from $\mathbb{Z}_q^n$, except with probability $e^{-t_1 n/16} + e^{-t_2/4}$, there exists a subsequence of n vectors such that the $n \times n$ matrix they form is invertible modulo q. Moreover, such a subsequence can be found efficiently.*

Proof. We consider the following procedure. Let k be a counter, initialized to 0, indicating the number of vectors currently in the subsequence, and let $A \in \mathbb{Z}_q^{n\times k}$ be the matrix whose columns are formed by the current subsequence. We also maintain a unimodular matrix $U \in \mathbb{Z}^{n\times n}$, initially set to the identity, satisfying the invariant that $U \cdot A \in \mathbb{Z}_q^{n\times k}$ has the following form: its top $k \times k$ submatrix is upper triangular with each diagonal coefficient coprime with q; its bottom $(n-k) \times k$ submatrix is zero. The procedure considers the vectors $\mathbf{a}_i$ one by one. For each vector $\mathbf{a}$, if it is such that the gcd of the last $n - k$ entries of $U\mathbf{a}$, call it g, is coprime with q, then it does the following: it adds $\mathbf{a}$ to the subsequence, computes (using, say, the extended GCD algorithm) a unimodular matrix V that acts as identity on the first k coordinates and for which the last $n - k$ coordinates of $VU\mathbf{a}$ are $(g, 0, \ldots, 0)$, replaces U with VU, and increments k.

It is easy to see that the procedure's output is correct if it reaches $k = n$. It therefore suffices to analyze the probability that this event happens. For this we use the following two facts to handle the cases $k < n - 1$

and $k = n - 1$, respectively. First, the probability that the gcd of two uniformly random numbers modulo q is coprime with q is

$$\prod_{p|q,\ p \text{ prime}} (1 - p^{-2}) \geq \prod_{p \text{ prime}} (1 - p^{-2}) = \zeta(2)^{-1} \approx 0.61,$$

where ζ is the Riemann zeta function. Second, the probability that one uniformly random number modulo q is coprime with q is $\varphi(q)/q$, where φ is Euler's totient function. By [BS96, Theorem 8.8.7], this probability is at least $(e^{\gamma} \ln \ln q + 3/(\ln \ln q))^{-1}$ where γ is Euler's constant, which for $q \geq 25$ is at least $(4 \ln \ln q)^{-1}$.

Using the (multiplicative) Chernoff bound, the first fact, and the fact that $U\mathbf{a}$ is uniform in $\mathbb{Z}_q^n$ since U is unimodular, we see that the probability that $k < n - 1$ after considering $t_1 n$ vector is at most $e^{-t_1 n/16}$. Moreover, once $k = n - 1$, using the second fact we get that the probability that after considering $t_2 \ln \ln q$ additional vectors we still have $k = n - 1$ is at most $e^{-t_2/4}$. □

Unknown (Bounded) Noise Rate. We also consider a variant of LWE in which the amount of noise is some unknown $\beta \leq \alpha$ (as opposed to exactly α), with β possibly depending on the secret $\mathbf{s}$. As the following lemma shows, this does not make the problem significantly harder.

Definition 2.14. *For integers $n, q \geq 1$ and $\alpha \in (0, 1)$, $\mathsf{LWE}_{n,q,\leq\alpha}$ is the problem of solving $\mathsf{LWE}_{n,q,\beta}$ for any $\beta = \beta(\mathbf{s}) \leq \alpha$.*

Lemma 2.15. *Let $\mathcal{A}$ be an algorithm for $\mathsf{LWE}_{n,m,q,\alpha}$ with advantage at least $\varepsilon > 0$. Then there exists an algorithm $\mathcal{B}$ for $\mathsf{LWE}_{n,m',q,\leq\alpha}$ using oracle access to $\mathcal{A}$ and with advantage at least $1/3$, where both m' and its running time are $\mathrm{poly}(m, 1/\varepsilon, n, \log q)$.*

The proof is standard (see, e.g., [Reg05, Lemma 3.7] for the analogous statement for the search version of LWE). The idea is to use Chernoff bound to estimate $\mathcal{A}$'s success probability on the uniform distribution, and then add noise in small increments to our given distribution and estimate $\mathcal{A}$'s behavior on the resulting distributions. If there is a gap between any of these and the uniform behavior, the input distribution is deemed non-uniform. The full proof is omitted.

Relation to Lattice Problems. Regev [Reg05] and Peikert [Pei09] showed quantum and classical reductions (respectively) from the worst-case hardness of the GapSVP problem to the search version of LWE. (We note that the quantum reduction in [Reg05] also shows a reduction from SIVP.) As mentioned in the introduction, the classical reduction only works when the modulus q is exponential in the dimension n. This is summarized in the following theorem, which is derived from [Reg05, Theorem 3.1] and [Pei09, Theorem 3.1].

Theorem 2.16. *Let $n, q \geq 1$ be integers and let $\alpha \in (0, 1)$ be such that $\alpha q \geq 2\sqrt{n}$. Then there exists a quantum reduction from worst-case n-dimensional $\mathsf{GapSVP}_{\widetilde{O}(n/\alpha)}$ to $\mathsf{LWE}_{n,q,\alpha}$. If in addition $q \geq 2^{n/2}$ then there is also a classical reduction between those problems.*

In order to obtain hardness of the *decision* version of LWE, which is the one we consider throughout the paper, one employs a search-to-decision reduction. Several such reductions appear in the literature (e.g., [Reg05, Pei09, MP12]). The most recent reduction by Micciancio and Peikert [MP12], which essentially subsumes all previous reductions, requires the modulus q to be smooth. Below we give the special case when the modulus is a power of 2, which suffices for our purposes. It follows from our results that (decision) LWE is hard not just for a smooth modulus q, as follows from [MP12], but actually for all moduli q, including prime moduli, with only a small deterioration in the noise (see Corollary 3.3).

Theorem 2.17 (Special case of [MP12, Theorem 3.1]). *Let q be a power of 2, and α satisfy $1/q < \alpha < 1/\omega(\sqrt{\log n})$. Then there exists an efficient reduction from search $\mathsf{LWE}_{n,q,\alpha}$ to (decision) $\mathsf{LWE}_{n,q,\alpha'}$ for $\alpha' = \alpha \cdot \omega(\log n)$.*

3 Modulus-Dimension Switching

The main results of this section are Corollaries 3.2 and 3.4 below. Both are special cases of the following technical theorem. We say that a distribution $\mathcal{D}$ over $\mathbb{Z}^n$ is (B, δ)-bounded for some reals $B, \delta \geq 0$ if the probability that $\mathbf{x} \leftarrow \mathcal{D}$ has norm greater than B is at most δ.

Theorem 3.1. *Let $m, n, n', q, q' \geq 1$ be integers, let $\mathbf{G} \in \mathbb{Z}^{n' \times n}$ be such that the lattice $\Lambda = \frac{1}{q'}\mathbf{G}^T\mathbb{Z}^{n'} + \mathbb{Z}^n$ has a known basis $\mathbf{B}$, and let $\mathcal{D}$ be an arbitrary (B, δ)-bounded distribution over $\mathbb{Z}^n$. Let $\alpha, \beta > 0$ and $\varepsilon \in (0, 1/2)$ satisfy*

$$\beta^2 \geq \alpha^2 + (4/\pi)\ln(2n(1 + 1/\varepsilon)) \cdot (\max\{q^{-1}, \|\widetilde{\mathbf{B}}\|\} \cdot B)^2.$$

Then there is an efficient (transformation) reduction from $\mathsf{LWE}_{n,m,q,\leq\alpha}(\mathcal{D})$ to $\mathsf{LWE}_{n',m,q',\leq\beta}(\mathbf{G} \cdot \mathcal{D})$ that reduces the advantage by at most $\delta + 14\varepsilon m$.

Here we use the notation $\|\widetilde{\mathbf{B}}\|$ from Lemma 2.3. We also note that if needed, the distribution on secrets produced by the reduction can always be turned into the uniform distribution on $\mathbb{Z}_{q'}^{n'}$, as mentioned after Definition 2.11. Also, we recall that there exists an elementary reduction from $\mathsf{LWE}_{n',q',\leq\beta}$ to $\mathsf{LWE}_{n',q',\beta}$ (see Lemma 2.15).

Here we state two important corollaries of the theorem. The first corresponds to just modulus reduction (the LWE dimension is preserved), and is obtained by letting $n' = n$, $\mathbf{G} = \mathbf{I}$ be the n-dimensional identity matrix, and $\mathbf{B} = \mathbf{I}/q'$. For example, we can take $q \geq q' \geq \sqrt{2\ln(2n(1 + 1/\varepsilon))} \cdot (B/\alpha)$ and $\beta = \sqrt{2}\alpha$, which corresponds to reducing an arbitrary modulus to almost B/α, while increasing the initial error rate α by just a small constant factor.

Corollary 3.2. *For any $m, n \geq 1$, $q \geq q' \geq 1$, (B, δ)-bounded distribution $\mathcal{D}$ over $\mathbb{Z}^n$, $\alpha, \beta > 0$ and $\varepsilon \in (0, 1/2)$ such that*

$$\beta^2 \geq \alpha^2 + (4/\pi)\ln(2n(1 + 1/\varepsilon)) \cdot (B/q')^2,$$

there is an efficient reduction from $\mathsf{LWE}_{n,m,q,\leq\alpha}(\mathcal{D})$ to $\mathsf{LWE}_{n,m,q',\leq\beta}(\mathcal{D})$ that reduces the advantage by at most $\delta + 14\varepsilon m$.

In particular, by using the normal form of LWE (Lemma 2.12), in which the secret has distribution $\mathcal{D} = D_{\mathbb{Z}^n,\sqrt{2}\alpha q}$, we can switch to a power-of-2 modulus with only a small loss in the noise rate, as described in the following corollary. Together with the known search-to-decision reduction (Theorem 2.17), this extends the known hardness of (decision) LWE to *any* modulus q. Here we use that $\mathcal{D} = D_{\mathbb{Z}^n,r}$ is $(Cr\sqrt{n\log(n/\delta)}, \delta)$-bounded for some universal constant $C > 0$, which follows by taking union bound over the n coordinates. (Alternatively, one could use that it is $(r\sqrt{n}, 2^{-n})$-bounded, as follows from [Ban93, Lemma 1.5], leading to a slightly tighter statement for large n.)

Corollary 3.3. *Let $\delta \in (0, 1/2)$, $m \geq n \geq 1$, $q' \geq 25$. Let also $q \in [q', 2q')$ be the smallest power of 2 not smaller than q' and $\alpha \geq \sqrt{\ln(2n(1 + 16/\delta)/\pi)}/q$. There exists an efficient (transformation) reduction from $\mathsf{LWE}_{n,m,q,\alpha}$ to $\mathsf{LWE}_{n,m',q',\leq\beta}$ where $m' = m - (16n + 4\ln\ln q)$ and*

$$\beta = C\alpha\sqrt{n}\sqrt{\log(n/\delta)\log(m/\delta)}$$

for some universal constant $C > 0$, that turns advantage of ζ into an advantage of at least $(\zeta - \delta)/4$.

Another corollary illustrates a modulus-dimension tradeoff. Assume $n = kn'$ for some $k \geq 1$, and let $q' = q^k$. Let $\mathbf{G} = \mathbf{I}_{n'} \otimes \mathbf{g}$, where $\mathbf{g} = (1, q, q^2, \ldots, q^{k-1})^T \in \mathbb{Z}^k$. We then have $\Lambda = q^{-k}\mathbf{G}^T\mathbb{Z}^{n'} + \mathbb{Z}^n$. A basis of Λ is given by

$$\mathbf{B} = \mathbf{I}_{n'} \otimes \begin{bmatrix} q^{-1} & q^{-2} & \cdots & q^{-k} \\ & q^{-1} & \cdots & q^{1-k} \\ & & \ddots & \vdots \\ & & & q^{-1} \end{bmatrix} \in \mathbb{R}^{n \times n};$$

this is since the column vectors of $\mathbf{B}$ belong to Λ and the determinants match. Orthogonalizing from left to right, we have $\widetilde{\mathbf{B}} = q^{-1}\mathbf{I}$ and so $\|\widetilde{\mathbf{B}}\| = q^{-1}$. We therefore obtain the following corollary, showing that we can trade off the dimension against the modulus, holding $n \log q = n' \log q'$ fixed. For example, letting $\mathcal{D} = D_{\mathbb{Z}^n, \alpha q}$ (corresponding to a secret in normal form, see Lemma 2.12), which is $(\alpha q\sqrt{n}, 2^{-n})$-bounded, the reduction increases the error rate by about a $\sqrt{n}$ factor.

Corollary 3.4. *For any $n, m, q \geq 1$, $k \geq 1$ that divides n, (B, δ)-bounded distribution $\mathcal{D}$ over $\mathbb{Z}^n$, $\alpha, \beta > 0$, and $\varepsilon \in (0, 1/2)$ such that*

$$\beta^2 \geq \alpha^2 + (4/\pi)\ln(2n(1 + 1/\varepsilon)) \cdot (B/q)^2,$$

there is an efficient reduction from $\mathsf{LWE}_{n,m,q,\leq\alpha}(\mathcal{D})$ to $\mathsf{LWE}_{n/k,m,q^k,\leq\beta}(\mathbf{G} \cdot \mathcal{D})$ that reduces the advantage by at most $\delta + 14\varepsilon m$, where $\mathbf{G} = \mathbf{I}_{n/k} \otimes (1, q, q^2, \ldots, q^{k-1})^T$.

Theorem 3.1 follows immediately from the following lemma.

Lemma 3.5. *Adopt the notation of Theorem 3.1, and let*

$$r \geq \max\{q^{-1}, \|\widetilde{\mathbf{B}}\|\} \cdot \sqrt{2\ln(2n(1 + 1/\varepsilon))/\pi}.$$

There is an efficient mapping from $\mathbb{T}_q^n \times \mathbb{T}$ to $\mathbb{T}_{q'}^{n'} \times \mathbb{T}$, which has the following properties:

- *If the input is uniformly random, then the output is within statistical distance 4ε from the uniform distribution.*
- *If the input is distributed according to $A_{q,\mathbf{s},D_\alpha}$ for some $\mathbf{s} \in \mathbb{Z}^n$ with $\|\mathbf{s}\| \leq B$, then the output distribution is within statistical distance 10ε from $A_{q',\mathbf{Gs},D_{\alpha'}}$, where $(\alpha')^2 = \alpha^2 + r^2(\|\mathbf{s}\|^2 + B^2) \leq \alpha^2 + 2(rB)^2$.*

Proof. The main idea behind the reduction is to encode $\mathbb{T}_q^n$ into $\mathbb{T}_{q'}^{n'}$, so that the mod-1 inner products between vectors in $\mathbb{T}_q^n$ and a short vector $\mathbf{s} \in \mathbb{Z}^n$, and between vectors in $\mathbb{T}_{q'}^{n'}$ and $\mathbf{Gs} \in \mathbb{Z}^{n'}$, are nearly equivalent. In a bit more detail, the reduction will map its input vector $\mathbf{a} \in \mathbb{T}_q^n$ (from the given LWE-or-uniform distribution) to a vector $\mathbf{a}' \in \mathbb{T}_{q'}^{n'}$, so that

$$\langle \mathbf{a}', \mathbf{Gs} \rangle = \langle \mathbf{G}^T\mathbf{a}', \mathbf{s} \rangle \approx \langle \mathbf{a}, \mathbf{s} \rangle \bmod 1$$

for any (unknown) $\mathbf{s} \in \mathbb{Z}^n$. To do this, it randomly samples $\mathbf{a}'$ so that $\mathbf{G}^T\mathbf{a}' \approx \mathbf{a} \bmod \mathbb{Z}^n$, where the approximation error will be a discrete Gaussian of parameter r.

We can now formally define the reduction, which works as follows. On an input pair $(\mathbf{a}, b) \in \mathbb{T}_q^n \times \mathbb{T}$, it does the following:

- Choose $\mathbf{f} \leftarrow D_{\Lambda-\mathbf{a},r}$ using Lemma 2.3 with basis $\mathbf{B}$, and let $\mathbf{v} = \mathbf{a} + \mathbf{f} \in \Lambda/\mathbb{Z}^n$. (The coset $\Lambda - \mathbf{a}$ is well defined since $\mathbf{a} = \bar{\mathbf{a}} + \mathbb{Z}^n$ is some coset of $\mathbb{Z}^n \subseteq \Lambda$.) Choose a uniformly random solution $\mathbf{a}' \in \mathbb{T}_{q'}^{n'}$ to the equation $\mathbf{G}^T\mathbf{a}' = \mathbf{v} \bmod \mathbb{Z}^n$. This can be done by computing a basis of the solution set $\mathbf{G}^T\mathbf{a}' = \mathbf{0} \bmod \mathbb{Z}^n$, and adding a uniform element from that set to an arbitrary solution to the equation $\mathbf{G}^T\mathbf{a}' = \mathbf{v} \bmod \mathbb{Z}^n$.
- Choose $e' \leftarrow D_{rB}$ and let $b' = b + e' \in \mathbb{T}$.
- Output $(\mathbf{a}', b')$.

We now analyze the reduction. First, if the distribution of the input is uniform, then it suffices to show that $\mathbf{a}'$ is (nearly) uniformly random, because both b and e' are independent of $\mathbf{a}'$, and $b \in \mathbb{T}$ is uniform. To prove this claim, notice that it suffices to show that the coset $\mathbf{v} \in \Lambda/\mathbb{Z}^n$ is (nearly) uniformly random, because each $\mathbf{v}$ has the same number of solutions $\mathbf{a}'$ to $\mathbf{G}^T\mathbf{a}' = \mathbf{v} \bmod \mathbb{Z}^n$. Next, observe that for any $\bar{\mathbf{a}} \in \mathbb{T}_q^n$ and $\bar{\mathbf{f}} \in \Lambda - \bar{\mathbf{a}}$, we have by Lemma 2.7 (using that $r \geq \eta_\varepsilon(\Lambda)$ by Lemma 2.5) that

$$\begin{aligned}\Pr[\mathbf{a} = \bar{\mathbf{a}} \wedge \mathbf{f} = \bar{\mathbf{f}}] &= q^{-n} \cdot \rho_r(\bar{\mathbf{f}})/\rho_r(\Lambda - \bar{\mathbf{a}}) \\ &\in C\big[1, \tfrac{1+\varepsilon}{1-\varepsilon}\big] \cdot \rho_r(\bar{\mathbf{f}}). \end{aligned} \tag{3.1}$$

where $C = q^{-n}/\rho_r(\Lambda)$ is a normalizing value that does not depend on $\bar{\mathbf{a}}$ or $\bar{\mathbf{f}}$. Therefore, by summing over all $\bar{\mathbf{a}}, \bar{\mathbf{f}}$ satisfying $\bar{\mathbf{a}} + \bar{\mathbf{f}} = \bar{\mathbf{v}}$, we obtain that for any $\bar{\mathbf{v}} \in \Lambda/\mathbb{Z}^n$,

$$\Pr[\mathbf{v} = \bar{\mathbf{v}}] \in C\big[1, \tfrac{1+\varepsilon}{1-\varepsilon}\big] \cdot \rho_r(q^{-1}\mathbb{Z}^n + \bar{\mathbf{v}}).$$

Since $r \geq \eta_\varepsilon(q^{-1}\mathbb{Z}^n)$ (by Lemma 2.5), Lemma 2.7 implies that $\Pr[\mathbf{v} = \bar{\mathbf{v}}] \in \big[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\big] C'$ for a constant C' that is independent of $\bar{\mathbf{v}}$. By Claim 2.1, this shows that $\mathbf{a}'$ is within statistical distance $1-((1-\varepsilon)/(1+\varepsilon))^2 \leq 4\varepsilon$ of the uniform distribution.

It remains to show that the reduction maps $A_{q,\mathbf{s},D_\alpha}$ to $A_{q',\mathbf{G}\mathbf{s},D_\beta}$. Let the input sample from the former distribution be $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s}\rangle + e)$, where $e \leftarrow D_\alpha$. As argued above, the output $\mathbf{a}'$ is (nearly) uniform over $\mathbb{T}_{q'}^{n'}$. So condition now on any fixed value $\overline{\mathbf{a}'} \in \mathbb{T}_{q'}^{n'}$ of $\mathbf{a}'$, and let $\bar{\mathbf{v}} = \mathbf{G}^T\overline{\mathbf{a}'} \bmod \mathbb{Z}^n$. We have

$$b' = \langle \mathbf{a}, \mathbf{s}\rangle + e + e' = \langle \overline{\mathbf{a}'}, \mathbf{G}\mathbf{s}\rangle + e + \langle -\mathbf{f}, \mathbf{s}\rangle + e' \bmod 1.$$

By Claim 2.1 and (3.1) (and noting that if $\mathbf{f} = \bar{\mathbf{f}}$ then $\mathbf{a} = \bar{\mathbf{v}} - \bar{\mathbf{f}} \bmod \mathbb{Z}^n$), the distribution of $-\mathbf{f}$ is within statistical distance $1-(1-\varepsilon)/(1+\varepsilon) \leq 2\varepsilon$ of $D_{q^{-1}\mathbb{Z}^n - \bar{\mathbf{v}}, r}$. By Lemma 2.9 (using $r \geq \sqrt{2}\eta_\varepsilon(q^{-1}\mathbb{Z}^n)$ and $\|\mathbf{s}\| \leq B$), the distribution of $\langle -\mathbf{f}, \mathbf{s}\rangle + e'$ is within statistical distance 6ε from D_t, where $t^2 = r^2(\|\mathbf{s}\|^2 + B^2)$. It therefore follows that $e + \langle -\mathbf{f}, \mathbf{s}\rangle + e'$ is within statistical distance 6ε from $D_{(t^2+\alpha^2)^{1/2}}$, as required. □

4 Hardness of LWE with Binary Secret

The following is the main theorem of this section.

Theorem 4.1. *Let $k, q \geq 1$, and $m \geq n \geq 1$ be integers, and let $\varepsilon \in (0, 1/2)$, $\alpha, \delta > 0$, be such that $n \geq (k+1)\log_2 q + 2\log_2(1/\delta)$, $\alpha \geq \sqrt{\ln(2n(1+1/\varepsilon))/\pi}/q$. There exist three (transformation) reductions from $\mathsf{LWE}_{k,m,q,\alpha}$ to $\mathsf{binLWE}_{n,m,q,\leq\sqrt{10n}\alpha}$, such that for any algorithm for the latter problem with advantage ζ, at least one of the reductions produces an algorithm for the former problem with advantage at least*

$$(\zeta - \delta)/(3m) - 41\varepsilon/2 - \sum_{p|q,\ p \text{ prime}} p^{-k-1}. \tag{4.1}$$

By combining Theorem 4.1 with the reduction in Corollary 3.2 (and noting that $\{0,1\}^n$ is $(\sqrt{n},0)$ bounded), we can replace the binLWE problem above with $\mathsf{binLWE}_{n,m,q',\beta}$ for any $q' \geq 1$ and $\xi > 0$ where

$$\beta := \left(10n\alpha^2 + \frac{4n}{\pi q'^2}\ln(2n(1+1/\xi))\right)^{1/2},$$

while decreasing the advantage in (4.1) by $14\xi m$. Recalling that LWE of dimension $k = \sqrt{n}$ and modulus $q = 2^{k/2}$ (assume k is even) is known to be classically as hard as $\sqrt{n}$-dimensional lattice problems (Theorems 2.16 and 2.17), this gives a formal statement of Theorem 1.1. The modulus q' can be taken almost as small as $\sqrt{n}$.

For most purposes the sum over prime factors of q in (4.1) is negligible. For instance, in deriving the formal statement of Theorem 1.1 above, we used a q that is a power of 2, in which case the sum is $2^{-k-1} = 2^{-\sqrt{n}-1}$, which is negligible. If needed, one can improve this by applying the modulus switching reduction (Corollary 3.3) before applying Theorem 4.1 in order to make q prime. (Strictly speaking, one also needs to apply Lemma 2.15 to replace the "unknown noise" variant of LWE given by Corollary 3.3 with the fixed noise variant.) This improves the advantage loss to $q^{-\sqrt{n}-1}$ which is roughly 2^{-n}.

In a high level, the proof of the theorem follows by combining three main steps. The first, given in Section 4.1, reduces LWE to a variant in which the first equation is errorless. The second, given in Section 4.2, reduces the latter to the intermediate problem extLWE, another variant of LWE in which some information on the noise elements is leaked. Finally, in Section 4.3, we reduce extLWE to LWE with $\{0,1\}$ secret. We note that the first reduction is relatively standard; it is the other two that we consider as the main contribution of this section. We now proceed with more details (see also Figure 1).

Proof. First, since $m \geq n$, Lemma 4.3 provides a transformation reduction from $\mathsf{LWE}_{k,m,q,\alpha}$ to first-is-errorless $\mathsf{LWE}_{k+1,n,q,\alpha}$, while reducing the advantage by at most 2^{-k+1}. Next, Lemma 4.7 with $\mathcal{Z} = \{0,1\}^n$, which is of quality $\xi = 2$ by Claim 4.6, reduces the latter problem to $\mathsf{extLWE}_{k+1,n,q,\sqrt{5}\alpha,\{0,1\}^n}$ while reducing the advantage by at most $33\varepsilon/2$. Then, Lemma 4.8 reduces the latter problem to $\mathsf{extLWE}^m_{k+1,n,q,\sqrt{5}\alpha,\{0,1\}^n}$, while losing a factor of m in the advantage. Finally, Lemma 4.9 provides three reductions to $\mathsf{binLWE}_{n,m,q,\leq\sqrt{10n}\alpha}$: two from the latter problem, and one from $\mathsf{LWE}_{k+1,m,q,\sqrt{5n}\alpha}$, guaranteeing that the sum of advantages is at least the original advantage minus $4m\varepsilon + \delta$. Together with the trivial reduction from $\mathsf{LWE}_{k,m,q,\alpha}$ to $\mathsf{LWE}_{k+1,m,q,\sqrt{5n}\alpha}$ (which incurs no loss in advantage), this completes the proof. □

4.1 First-is-errorless LWE

We first define a variant of LWE in which the first equation is given without error, and then show in Lemma 4.3 that it is still hard.

Definition 4.2. *For integers $n, q \geq 1$ and an error distribution ϕ over $\mathbb{R}$, the "first-is-errorless" variant of the* LWE *problem is to distinguish between the following two scenarios. In the first, the first sample is uniform over $\mathbb{T}_q^n \times \mathbb{T}_q$ and the rest are uniform over $\mathbb{T}_q^n \times \mathbb{T}$. In the second, there is an unknown uniformly distributed $\mathbf{s} \in \{0,\ldots,q-1\}^n$, the first sample we get is from $A_{q,\mathbf{s},\{0\}}$ (where $\{0\}$ denotes the distribution that is deterministically zero) and the rest are from $A_{q,\mathbf{s},\phi}$.*

Lemma 4.3. *For any $n \geq 2$, $m, q \geq 1$, and error distribution ϕ, there is an efficient (transformation) reduction from $\mathsf{LWE}_{n-1,m,q,\phi}$ to the first-is-errorless variant of $\mathsf{LWE}_{n,m,q,\phi}$ that reduces the advantage by at most $\sum_p p^{-n}$, with the sum going over all prime factors of q.*

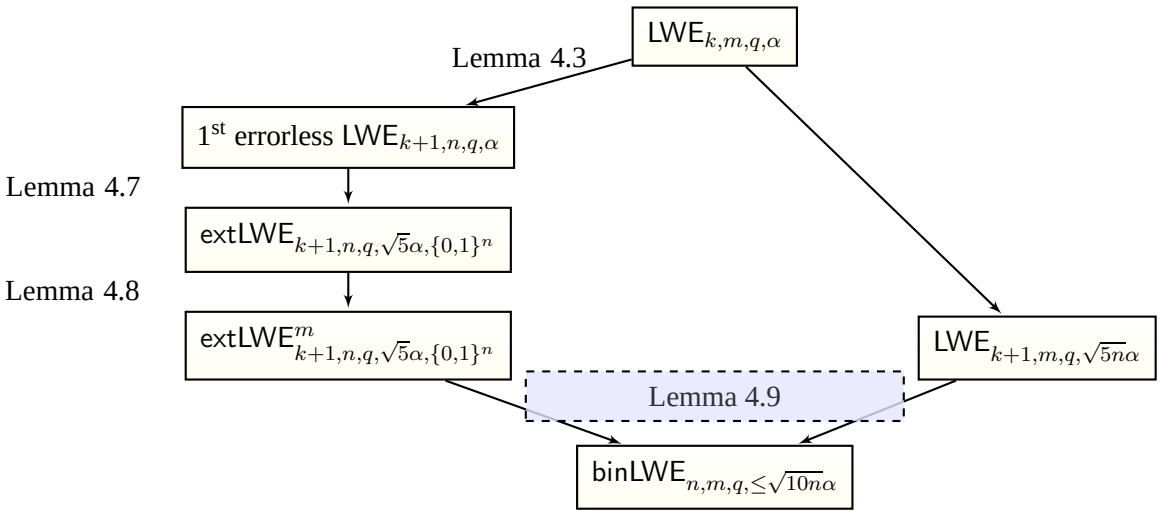


Figure 1: Summary of reductions used in Theorem 4.1

Notice that if q is prime the loss in advantage is at most q^{-n}. Alternatively, for any number q we can bound it by

$$\sum_{k\geq 2} k^{-n} \leq 2^{-n} + \int_2^\infty t^{-n}\mathrm{d}t \leq 2^{-n+2},$$

which might be good enough when n is large.

Proof. The reduction starts by choosing a vector $\mathbf{a}'$ uniformly at random from $\{0,\ldots,q-1\}^n$. Let r be the greatest common divisor of the coordinates of $\mathbf{a}'$. If it is not coprime to q, we abort. The probability that this happens is at most

$$\sum_{p \text{ prime},\ p|q} p^{-n}.$$

Assuming we do not abort, we proceed by finding a matrix $\mathbf{U} \in \mathbb{Z}^{n\times n}$ that is invertible modulo q and whose leftmost column is $\mathbf{a}'$. Such a matrix exists, and can be found efficiently. For instance, using the extended GCD algorithm, we find an $n \times n$ unimodular matrix $\mathbf{R}$ such that $\mathbf{R}\mathbf{a}' = (r, 0, \ldots, 0)^T$. Then $\mathbf{R}^{-1} \cdot \mathrm{diag}(r, 1, \ldots, 1)$ is the desired matrix. We also pick a uniform element $s_0 \in \{0,\ldots,q-1\}$. The reduction now proceeds as follows. The first sample it outputs is $(\mathbf{a}'/q, s_0/q)$. The remaining samples are produced by taking a sample $(\mathbf{a}, b)$ from the given oracle, picking a fresh uniformly random $d \in \mathbb{T}_q$, and outputting $(\mathbf{U}(d|\mathbf{a}), b + (s_0 \cdot d))$ with the vertical bar denoting concatenation. It is easy to verify correctness: given uniform samples, the reduction outputs uniform samples (with the first sample's b component uniform over $\mathbb{T}_q$), up to statistical distance 2^{-n+1}; and given samples from $A_{q,\mathbf{s},\phi}$, the reduction outputs one sample from $A_{q,\mathbf{s}',\{0\}}$ and the remaining samples from $A_{q,\mathbf{s}',\phi}$, up to statistical distance 2^{-n+1}, where $\mathbf{s}' = (\mathbf{U}^{-1})^T(s_0|\mathbf{s}) \bmod q$. This proves correctness since $\mathbf{U}$, being invertible modulo q, induces a bijection on $\mathbb{Z}_q^n$, and so $\mathbf{s}'$ is uniform in $\{0,\ldots,q-1\}^n$. □

4.2 Extended LWE

We next define the intermediate problem extLWE. (This definition is of an easier problem than the one considered in previous work [AP12], which makes our hardness result stronger.)

Definition 4.4. *For $n, m, q, t \geq 1$, $\mathcal{Z} \subseteq \mathbb{Z}^m$, and a distribution χ over $\frac{1}{q}\mathbb{Z}^m$, the* $\mathsf{extLWE}^t_{n,m,q,\chi,\mathcal{Z}}$ *problem is as follows. The algorithm gets to choose $\mathbf{z} \in \mathcal{Z}$ and then receives a tuple*

$$(\mathbf{A}, (\mathbf{b}_i)_{i\in[t]}, (\langle \mathbf{e}_i, \mathbf{z}\rangle)_{i\in[t]}) \in \mathbb{T}_q^{n\times m} \times (\mathbb{T}_q^m)^t \times (\tfrac{1}{q}\mathbb{Z})^t.$$

Its goal is to distinguish between the following two cases. In the first, $\mathbf{A} \in \mathbb{T}_q^{n\times m}$ is chosen uniformly, $\mathbf{e}_i \in \frac{1}{q}\mathbb{Z}^m$ are chosen from χ, and $\mathbf{b}_i = \mathbf{A}^T\mathbf{s}_i + \mathbf{e}_i \bmod 1$ where $\mathbf{s}_i \in \{0, \ldots, q-1\}^n$ are chosen uniformly. The second case is identical, except that the $\mathbf{b}_i$ are chosen uniformly in $\mathbb{T}_q^m$ independently of everything else.

When $t = 1$, we omit the superscript t. Also, when χ is $D_{q^{-1}\mathbb{Z}^m,\alpha}$ for some $\alpha > 0$, we replace the subscript χ by α. We note that a discrete version of LWE can be defined as a special case of extLWE by setting $\mathcal{Z} = \{0^m\}$. We next define a measure of quality of sets $\mathcal{Z}$.

Definition 4.5. *For a real $\xi > 0$ and a set $\mathcal{Z} \subseteq \mathbb{Z}^m$ we say that $\mathcal{Z}$ is of* quality ξ *if given any $\mathbf{z} \in \mathcal{Z}$, we can efficiently find a unimodular matrix $\mathbf{U} \in \mathbb{Z}^{m\times m}$ such that if $\mathbf{U}' \in \mathbb{Z}^{m\times(m-1)}$ is the matrix obtained from $\mathbf{U}$ by removing its leftmost column then all of the columns of $\mathbf{U}'$ are orthogonal to $\mathbf{z}$ and its largest singular value is at most ξ.*

The idea in this definition is that the columns of $\mathbf{U}'$ form a basis of the lattice of integer points that are orthogonal to $\mathbf{z}$, i.e., the lattice $\{\mathbf{b} \in \mathbb{Z}^m : \langle \mathbf{b}, \mathbf{z}\rangle = 0\}$. The quality measures how "short" we can make this basis.

Claim 4.6. *The set $\mathcal{Z} = \{0, 1\}^m$ is of quality 2.*

Proof. Let $\mathbf{z} \in \mathcal{Z}$ and assume without loss of generality that its first $k \geq 1$ coordinates are 1 and the remaining $m - k$ are 0. Then consider the upper bidiagonal matrix $\mathbf{U}$ whose diagonal is all 1s and whose diagonal above the main diagonal is $(-1, \ldots, -1, 0, \ldots, 0)$ with -1 appearing $k - 1$ times. The matrix is clearly unimodular and all the columns except the first one are orthogonal to $\mathbf{z}$. Moreover, by the triangle inequality, we can bound the operator norm of $\mathbf{U}$ by the sum of that of the diagonal 1 matrix and the off-diagonal matrix, both of which clearly have norm at most 1. □

Lemma 4.7. *Let $\mathcal{Z} \subseteq \mathbb{Z}^m$ be of quality $\xi > 0$. Then for any $n, q \geq 1$, $\varepsilon \in (0, 1/2)$, and $\alpha, r \geq (\ln(2m(1 + 1/\varepsilon))/\pi)^{1/2}/q$, there is a (transformation) reduction from the first-is-errorless variant of* $\mathsf{LWE}_{n,m,q,\alpha}$ *to* $\mathsf{extLWE}_{n,m,q,(\alpha^2\xi^2+r^2)^{1/2},\mathcal{Z}}$ *that reduces the advantage by at most $33\varepsilon/2$.*

Proof. We first describe the reduction. Assume we are asked to provide samples for some $\mathbf{z} \in \mathcal{Z}$. We compute a unimodular $\mathbf{U} \in \mathbb{Z}^{m\times m}$ for $\mathbf{z}$ as in Definition 4.5, and let $\mathbf{U}' \in \mathbb{Z}^{m\times(m-1)}$ be the matrix formed by removing the first column of $\mathbf{U}$. We then take m samples from the given distribution, resulting in $(\mathbf{A}, \mathbf{b}) \in \mathbb{T}_q^{n\times m} \times (\mathbb{T}_q \times \mathbb{T}^{m-1})$. We also sample a vector $\mathbf{f}$ from the m-dimensional continuous Gaussian distribution $D_{\alpha(\xi^2\mathbf{I}-\mathbf{U}'\mathbf{U}'^T)^{1/2}}$, which is well defined since $\xi^2\mathbf{I} - \mathbf{U}'\mathbf{U}'^T$ is a positive semidefinite matrix by our assumption on $\mathbf{U}$. The output of the reduction is the tuple

$$(\mathbf{A}' = \mathbf{A}\mathbf{U}^T, \mathbf{b}' + \mathbf{c}, \langle \mathbf{z}, \mathbf{f} + \mathbf{c}\rangle) \in \mathbb{T}_q^{n\times m} \times \mathbb{T}_q^m \times \tfrac{1}{q}\mathbb{Z}, \tag{4.2}$$

where $\mathbf{b}' = \mathbf{U}\mathbf{b} + \mathbf{f}$, and $\mathbf{c}$ is chosen from the discrete Gaussian distribution $D_{q^{-1}\mathbb{Z}^m-\mathbf{b}',r}$ (using Lemma 2.3).

We now prove the correctness of the reduction. Consider first the case that we get valid LWE equations, i.e., $\mathbf{A}$ is uniform in $\mathbb{T}_q^{n\times m}$ and $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \in \mathbb{T}^m$ where $\mathbf{s} \in \{0, \ldots, q-1\}^n$ is uniformly chosen, the first coordinate of $\mathbf{e} \in \mathbb{R}^m$ is 0, and the remaining $m - 1$ coordinates are chosen from D_α. Since $\mathbf{U}$ is

unimodular, $\mathbf{A}' = \mathbf{A}\mathbf{U}^T$ is uniformly distributed in $\mathbb{T}_q^{n\times m}$ as required. From now on we condition on an arbitrary $\mathbf{A}'$ and analyze the distribution of the remaining two components of (4.2). Next,

$$\mathbf{b}' = \mathbf{U}\mathbf{b} + \mathbf{f} = \mathbf{A}'^T\mathbf{s} + \mathbf{U}\mathbf{e} + \mathbf{f}.$$

Since $\mathbf{U}\mathbf{e}$ is distributed as a continuous Gaussian $D_{\alpha\mathbf{U}'}$, the vector $\mathbf{U}\mathbf{e}+\mathbf{f}$ is a distributed as a *spherical* continuous Gaussian $D_{\alpha\xi}$. Moreover, since $\mathbf{A}'^T\mathbf{s} \in \mathbb{T}_q^m$, the coset $q^{-1}\mathbb{Z}^m - \mathbf{b}'$ is identical to $q^{-1}\mathbb{Z}^m - (\mathbf{U}\mathbf{e}+\mathbf{f})$, so we can see $\mathbf{c}$ as being chosen from $D_{q^{-1}\mathbb{Z}^m-(\mathbf{U}\mathbf{e}+\mathbf{f}),r}$. Therefore, by Lemma 2.10 and using that $r \geq \eta_\varepsilon(q^{-1}\mathbb{Z}^m)$ by Lemma 2.5, the distribution of $\mathbf{U}\mathbf{e}+\mathbf{f}+\mathbf{c}$ is within statistical distance 8ε of $D_{q^{-1}\mathbb{Z}^m,(\alpha^2\xi^2+r^2)^{1/2}}$. This shows that the second component in (4.2) is also distributed correctly. Finally, for the third component, by our assumption on $\mathbf{U}$ and the fact that the first coordinate of $\mathbf{e}$ is zero,

$$\langle \mathbf{z}, \mathbf{f}+\mathbf{c}\rangle = \langle \mathbf{z}, \mathbf{U}\mathbf{e}+\mathbf{f}+\mathbf{c}\rangle,$$

and so the third component gives the inner product of the noise with $\mathbf{z}$, as desired.

We now consider the case where the input is uniform, i.e., that $\mathbf{A}$ is uniform in $\mathbb{T}_q^{n\times m}$ and $\mathbf{b}$ is independent and uniform in $\mathbb{T}_q \times \mathbb{T}^{m-1}$. We first observe that by Lemma 2.6, since $\alpha \geq \eta_{\varepsilon/m}(q^{-1}\mathbb{Z})$ (by Lemma 2.5), the distribution of $(\mathbf{A},\mathbf{b})$ is within statistical distance $\varepsilon/2$ of the distribution of $(\mathbf{A},\mathbf{e}'+\mathbf{e})$ where $\mathbf{e}'$ is chosen uniformly in $\mathbb{T}_q^m$, the first coordinate of $\mathbf{e}$ is zero, and its remaining $m-1$ coordinates are chosen independently from D_α. So from now on assume our input is $(\mathbf{A},\mathbf{e}'+\mathbf{e})$. The first component of (4.2) is uniform in $\mathbb{T}_q^{n\times m}$ as before, and moreover, it is clearly independent of the other two. Moreover, since $\mathbf{b}' = \mathbf{U}\mathbf{e}' + \mathbf{U}\mathbf{e} + \mathbf{f}$ and $\mathbf{U}\mathbf{e}' \in \mathbb{T}_q^m$, the coset $q^{-1}\mathbb{Z}^m - \mathbf{b}'$ is identical to $q^{-1}\mathbb{Z}^m - (\mathbf{U}\mathbf{e}+\mathbf{f})$, and so $\mathbf{c}$ is distributed identically to the case of a valid LWE equation, and in particular is independent of $\mathbf{e}'$. This establishes that the third component of (4.2) is correctly distributed; moreover, since $\mathbf{e}'$ is independent of the first and third components, and $\mathbf{U}\mathbf{e}'$ is uniform in $\mathbb{T}_q^m$ (since $\mathbf{U}$ is unimodular), we get that the second component is uniform and independent of the other two, as desired. □

We end this section by stating the standard reduction to the multi-secret ($t \geq 1$) case of extended LWE.

Lemma 4.8. *Let $n, m, q, \chi, \mathcal{Z}$ be as in Definition 4.4 with χ efficiently sampleable, and let $t \geq 1$ be an integer. Then there is an efficient (transformation) reduction from* $\mathsf{extLWE}_{n,m,q,\chi,\mathcal{Z}}$ *to* $\mathsf{extLWE}^t_{n,m,q,\chi,\mathcal{Z}}$ *that reduces the advantage by a factor of t.*

The proof is by a standard hybrid argument. We bring it here for the sake of completeness. We note that the distribution of the secret vector $\mathbf{s}$ needs to be sampleable but otherwise it plays no role in the proof. The lemma therefore naturally extends to any (sampleable) distribution of $\mathbf{s}$.

Proof. Let $\mathcal{A}$ be an algorithm for $\mathsf{extLWE}^t_{n,m,q,\chi,\mathcal{Z}}$, let $\mathbf{z}$ be the vector output by $\mathcal{A}$ in the first step (note that this is a random variable) and let H_i denote the distribution

$$\left(\mathbf{A}, \{\mathbf{b}_1,\ldots,\mathbf{b}_i,\mathbf{u}_{i+1},\ldots,\mathbf{u}_t\}, \mathbf{z}, \{\langle \mathbf{z},\mathbf{e}_i\rangle\}_{i\in[t]}\right) ,$$

where $\mathbf{u}_{i+1},\ldots,\mathbf{u}_t$ are sampled independently and uniformly in $\mathbb{T}_q^m$. Then by definition $\mathrm{Adv}[\mathcal{A}] = |\Pr[\mathcal{A}(H_0)] - \Pr[\mathcal{A}(H_t)]|$.

We now describe an algorithm $\mathcal{B}$ for $\mathsf{extLWE}_{n,m,q,\chi,\mathcal{Z}}$: First, $\mathcal{B}$ runs $\mathcal{A}$ to obtain $\mathbf{z}$ and sends it to the challenger as its own $\mathbf{z}$. Then, given an input $(\mathbf{A},\mathbf{d},\mathbf{z},y)$ for $\mathsf{extLWE}_{n,m,q,\chi,\mathcal{Z}}$, the distinguisher $\mathcal{B}$ samples $i^* \leftarrow [t]$, and in addition $\mathbf{s}_1,\ldots,\mathbf{s}_{i^*-1} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_1,\ldots,\mathbf{e}_{i^*-1},\mathbf{e}_{i^*+1},\ldots,\mathbf{e}_t \leftarrow \chi^m$, $\mathbf{u}_{i^*+1},\ldots,\mathbf{u}_t \leftarrow \mathbb{T}_q^m$. It sets $\mathbf{b}_i = \mathbf{A}^T\cdot\mathbf{s}_i + \mathbf{e}_i \pmod 1$, and sends the following to $\mathcal{A}$:

$$(\mathbf{A}, \{\mathbf{b}_1,\ldots,\mathbf{b}_{i^*-1},\mathbf{d},\mathbf{u}_{i^*+1},\ldots,\mathbf{u}_t\}, \mathbf{z}, \{\langle \mathbf{z},\mathbf{e}_1\rangle,\ldots,\langle \mathbf{z},\mathbf{e}_{i^*-1}\rangle, y, \langle \mathbf{z},\mathbf{e}_{i^*+1}\rangle,\ldots,\langle \mathbf{z},\mathbf{e}_t\rangle\}) .$$

Finally, $\mathcal{B}$ outputs the same output as $\mathcal{A}$ did.

Note that when the input to $\mathcal{B}$ is distributed as $P_0 = (\mathbf{A}, \mathbf{b}, \mathbf{z}, \mathbf{z}^T \cdot \mathbf{e})$ with $\mathbf{b} = \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \pmod 1$, then $\mathcal{B}$ feeds $\mathcal{A}$ with exactly the distribution H_{i^*}. On the other hand, if the input to $\mathcal{B}$ is $P_1 = (\mathbf{A}, \mathbf{u}, \mathbf{z}, \mathbf{z}^T \cdot \mathbf{e})$ with $\mathbf{u} \leftarrow \mathbb{T}_q^m$, then $\mathcal{B}$ feeds $\mathcal{A}$ with H_{i^*-1}.

Since i^* is uniform in $[t]$, we get that

$$\begin{aligned} t\,\mathrm{Adv}[\mathcal{B}] &= t\,|\Pr[\mathcal{B}(P_0)] - \Pr[\mathcal{B}(P_1)]| \\ &= \left| \sum_{i^* \in [t]} \Pr[\mathcal{A}(H_{i^*})] - \sum_{i^* \in [t]} \Pr[\mathcal{A}(H_{i^*-1})] \right| \\ &= |\Pr[\mathcal{A}(H_t)] - \Pr[\mathcal{A}(H_0)]| \\ &= \mathrm{Adv}[\mathcal{A}]\,, \end{aligned}$$

and the result follows. □

4.3 Reducing to binary secret

Lemma 4.9. *Let $k, n, m, q \in \mathbb{N}$, $\varepsilon \in (0, 1/2)$, and $\delta, \alpha, \beta, \gamma > 0$ be such that $n \geq k \log_2 q + 2\log_2(1/\delta)$, $\beta \geq \sqrt{2\ln(2n(1+1/\varepsilon))/\pi}/q$, $\alpha = \sqrt{2n}\beta$, $\gamma = \sqrt{n}\beta$. Then there exist three efficient (transformation) reductions to $\mathsf{binLWE}_{n,m,q,\leq\alpha}$ from $\mathsf{extLWE}^m_{k,n,q,\beta,\{0,1\}^n}$, $\mathsf{LWE}_{k,m,q,\gamma}$, and $\mathsf{extLWE}^m_{k,n,q,\beta,\{0^n\}}$, such that if $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ are the algorithms obtained by applying these reductions (respectively) to an algorithm $\mathcal{A}$, then*

$$\mathrm{Adv}[\mathcal{A}] \leq \mathrm{Adv}[\mathcal{B}_1] + \mathrm{Adv}[\mathcal{B}_2] + \mathrm{Adv}[\mathcal{B}_3] + 4m\varepsilon + \delta\,.$$

Pointing out the trivial (transformation) reduction from $\mathsf{extLWE}^m_{k,n,q,\beta,\{0,1\}^n}$ to $\mathsf{extLWE}^m_{k,n,q,\beta,\{0^n\}}$, the lemma implies the hardness of $\mathsf{binLWE}_{n,m,q,\leq\alpha}$ based on the hardness of $\mathsf{extLWE}^m_{k,n,q,\beta,\{0,1\}^n}$ and $\mathsf{LWE}_{k,m,q,\gamma}$.

We note that our proof is actually more general, and holds for any binary distribution of min-entropy at least $k \log_2 q + 2\log_2(1/\delta)$, and not just a uniform binary secret as in the definition of binLWE.

Proof. The proof follows by a sequence of hybrids. Let $k, n, m, q, \varepsilon, \alpha, \beta, \gamma$ be as in the lemma statement. We consider $\mathbf{z} \leftarrow \{0,1\}^n$ and $\mathbf{e} \leftarrow D^m_{\alpha'}$ for $\alpha' = \sqrt{\beta^2\|\mathbf{z}\|^2 + \gamma^2} \leq \sqrt{2n}\beta = \alpha$. In addition, we let $\mathbf{A} \leftarrow \mathbb{T}_q^{n \times m}$, $\mathbf{u} \leftarrow \mathbb{T}^m$, and define $\mathbf{b} := \mathbf{A}^T \cdot \mathbf{z} + \mathbf{e} \pmod 1$. We consider an algorithm $\mathcal{A}$ that distinguishes between $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{A}, \mathbf{u})$.

We let H_0 denote the distribution $(\mathbf{A}, \mathbf{b})$ and H_1 the distribution

$$H_1 = (\mathbf{A}, \mathbf{A}^T\mathbf{z} - \mathbf{N}^T\mathbf{z} + \hat{\mathbf{e}} \bmod 1),$$

where $\mathbf{N} \leftarrow D^{n \times m}_{q^{-1}\mathbb{Z},\beta}$ and $\hat{\mathbf{e}} \leftarrow D^m_\gamma$. Using $\|\mathbf{z}\| \leq \sqrt{n}$ and that $\beta \geq \sqrt{2}\eta_\varepsilon(\mathbb{Z}^n)/q$ (by Lemma 2.5), it follows by Lemma 2.9 that the statistical distance between $-\mathbf{N}^T\mathbf{z} + \hat{\mathbf{e}}$ and $D^m_{\alpha'}$ is at most $4m\varepsilon$. It thus follows that

$$|\Pr[\mathcal{A}(H_0)] - \Pr[\mathcal{A}(H_1)]| \leq 4m\varepsilon\,. \tag{4.3}$$

We define a distribution H_2 as follows. Let $\mathbf{B} \leftarrow \mathbb{T}_q^{k \times m}$ and $\mathbf{C} \leftarrow \mathbb{T}_q^{k \times n}$. Let $\hat{\mathbf{A}} := q\mathbf{C}^T \cdot \mathbf{B} + \mathbf{N} \pmod 1$. Finally,

$$H_2 = (\hat{\mathbf{A}}, \hat{\mathbf{A}}^T \cdot \mathbf{z} - \mathbf{N}^T\mathbf{z} + \hat{\mathbf{e}}) = (\hat{\mathbf{A}}, q\mathbf{B}^T \cdot \mathbf{C} \cdot \mathbf{z} + \hat{\mathbf{e}})\,.$$

We now argue that there exists an adversary $\mathcal{B}_1$ for problem $\mathsf{extLWE}^m_{k,n,q,\beta,\{0,1\}^n}$, such that

$$\mathrm{Adv}[\mathcal{B}_1] = |\Pr[\mathcal{A}(H_1)] - \Pr[\mathcal{A}(H_2)]|\,. \tag{4.4}$$

This is because H_1, H_2 can be viewed as applying the same efficient transformation on the distributions $(\mathbf{C}, \mathbf{A}, \mathbf{N}^T\mathbf{z})$ and $(\mathbf{C}, \hat{\mathbf{A}}, \mathbf{N}^T\mathbf{z})$ respectively. Since distinguishing the latter distributions is exactly the $\mathsf{extLWE}^m_{k,n,q,\beta,\{0,1\}^n}$ problem (where the columns of $q \cdot \mathbf{B}$ are interpreted as the m secret vectors), the distinguisher $\mathcal{B}_1$ follows by first applying the aforementioned transformation and then applying $\mathcal{A}$.

For the next hybrid, we define $H_3 = (\hat{\mathbf{A}}, \mathbf{B}^T \cdot \mathbf{s} + \hat{\mathbf{e}})$, for $\mathbf{s} \leftarrow \mathbb{Z}_q^k$. It follows that

$$|\Pr[\mathcal{A}(H_2)] - \Pr[\mathcal{A}(H_3)]| \leq \delta \tag{4.5}$$

by the leftover hash lemma (see Lemma 2.2), since H_2, H_3 can be derived from $(\mathbf{C}, q\mathbf{C} \cdot \mathbf{z})$ and $(\mathbf{C}, \mathbf{s})$ respectively, whose statistical distance is at most δ.

Our next hybrid makes the second component uniform: $H_4 = (\hat{\mathbf{A}}, \mathbf{u})$. There exists an algorithm $\mathcal{B}_2$ for $\mathsf{LWE}_{k,m,q,\gamma}$ such that

$$\mathrm{Adv}[\mathcal{B}_2] = |\Pr[\mathcal{A}(H_3)] - \Pr[\mathcal{A}(H_4)]| \,, \tag{4.6}$$

since H_3, H_4 can be computed efficiently from $(\mathbf{B}, \mathbf{B}^T\mathbf{s} + \hat{\mathbf{e}}), (\mathbf{B}, \mathbf{u})$.

Lastly, we change $\hat{\mathbf{A}}$ back to uniform: $H_5 = (\mathbf{A}, \mathbf{u})$. There exists an algorithm $\mathcal{B}_3$ for $\mathsf{extLWE}^m_{k,n,q,\beta,\{0^n\}}$ such that

$$\mathrm{Adv}[\mathcal{B}_3] = |\Pr[\mathcal{A}(H_4)] - \Pr[\mathcal{A}(H_5)]| \,. \tag{4.7}$$

Eq. (4.7) is derived very similarly to Eq. (4.4): We notice that H_4, H_5 can be viewed as applying the same efficient transformation on the distributions $(\mathbf{C}, \hat{\mathbf{A}})$ and $(\mathbf{C}, \mathbf{A})$ respectively. Since distinguishing the latter distributions is exactly the $\mathsf{extLWE}^m_{k,n,q,\beta,\{0^n\}}$ problem (where the columns of $q \cdot \mathbf{B}$ are interpreted as the m secret vectors), the distinguisher $\mathcal{B}_3$ follows by first applying the aforementioned transformation and then applying $\mathcal{A}$.

Putting together Eq. (4.3), (4.4), (4.5), (4.6), (4.7), the lemma follows. □

5 Exact Gaussian Sampler

In this section we prove Lemma 2.3. As in [GPV08], the proof consists of two parts. In the first we consider the one-dimensional case, and in the second we use it recursively to sample from arbitrary lattices. Our one-dimensional sampler is based on rejection sampling, just like the one in [GPV08]. Unlike [GPV08], we use the continuous normal distribution as the source distribution which allows us to avoid truncation, and as a result obtain an exact sample. Our second part uses the same recursive routine as in [GPV08], but adds a rejection sampling step to it in order to take care of the deviation of its output from the desired distribution.

5.1 The one-dimensional case

Here we show how to sample from the discrete Gaussian distribution on arbitrary cosets of one-dimensional lattices. We use a standard rejection sampling procedure (see, e.g. [Dev86, Page 117] for a very similar procedure).

By scaling, we can restrict without loss of generality to the lattice $\mathbb{Z}$, i.e., we consider the task of sampling from $D_{\mathbb{Z}+c,r}$ for a given coset representative $c \in [0,1)$ and parameter $r > 0$. The sampling procedure is as follows. Let $Z_0 = \int_c^\infty \rho_r(x)\mathrm{d}x$, and $Z_1 = \int_{-\infty}^{c-1} \rho_r(x)\mathrm{d}x$. These two numbers can be computed efficiently by expressing them in terms of the error function. Let $Z = Z_0 + Z_1 + \rho_r(c) + \rho_r(c-1)$. The algorithm repeats the following until it outputs an answer:

- With probability $\rho_r(c)/Z$ it outputs c;

- With probability $\rho_r(c-1)/Z$ it outputs $c-1$;
- With probability Z_0/Z it chooses x from the restriction of the continuous normal distribution D_r to the interval $[c,\infty)$. Let y be the smallest element in $\mathbb{Z}+c$ that is larger than x. With probability $\rho_r(y)/\rho_r(x)$ output y, and otherwise repeat;
- With probability Z_1/Z it chooses x from the restriction of the continuous normal distribution D_r to the interval $(-\infty,c-1]$. Let y be the largest element in $\mathbb{Z}+c$ that is smaller than x. With probability $\rho_r(y)/\rho_r(x)$ output y, and otherwise repeat.

Consider now one iteration of the procedure. The probability of outputting c is $\rho_r(c)/Z$, that of outputting $c-1$ is $\rho_r(c-1)/Z$, that of outputting $c+k$ for some $k \geq 1$ is

$$\frac{Z_0}{Z} \cdot \frac{1}{Z_0} \int_{c+k-1}^{c+k} \rho_r(x) \cdot \frac{\rho_r(c+k)}{\rho_r(x)} \mathrm{d}x = \frac{\rho_r(c+k)}{Z},$$

and similarly, that of outputting $c-1-k$ for some $k \geq 1$ is $\rho_r(c-1-k)/Z$. From this it follows immediately that conditioned on outputting something, the output distribution has support on $\mathbb{Z}+c$ and probability mass function proportional to ρ_r, and is therefore the desired discrete Gaussian distribution $D_{\mathbb{Z}+c,r}$. Moreover, the probability of outputting something is

$$\frac{\rho_r(\mathbb{Z}+c)}{Z} = \frac{\rho_r(\mathbb{Z}+c)}{Z_0+Z_1+\rho_r(c)+\rho_r(c-1)} \geq \frac{\rho_r(\mathbb{Z}+c)}{\rho_r(\mathbb{Z}+c)+\rho_r(c)+\rho_r(c-1)} \geq \frac{1}{2}.$$

Therefore at each iteration the procedure has probability of at least $1/2$ to terminate. As a result, the probability that the number of iterations is greater than t is at most 2^{-t}, and in particular, the expected number of iterations is at most 2.

5.2 The general case

For completeness, we start by recalling the SampleD procedure described in [GPV08]. This is a recursive procedure that gets as input a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$, a parameter $r > 0$, and a vector $\mathbf{c} \in \mathbb{R}^n$, and outputs a vector in $\Lambda + \mathbf{c}$ whose distribution is close to that of $D_{\Lambda+\mathbf{c},r}$. Let $\widetilde{\mathbf{b}_1}, \ldots, \widetilde{\mathbf{b}_n}$ be the Gram-Schmidt orthogonalization of $\mathbf{b}_1, \ldots, \mathbf{b}_n$, and let $\overline{\mathbf{b}_1}, \ldots, \overline{\mathbf{b}_n}$ be the normalized Gram-Schmidt vectors, i.e., $\overline{\mathbf{b}_i} = \widetilde{\mathbf{b}_i}/\|\widetilde{\mathbf{b}_i}\|$. The procedure is the following.

1. Let $\mathbf{c}_n \leftarrow \mathbf{c}$. For $i \leftarrow n, \ldots, 1$, do:
   (a) Choose v_i from $D_{\|\widetilde{\mathbf{b}_i}\|\mathbb{Z}+\langle \mathbf{c}_i, \overline{\mathbf{b}_i}\rangle, r}$ using the exact one-dimensional sampler.
   (b) Let $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i + (v_i - \langle \mathbf{c}_i, \overline{\mathbf{b}_i}\rangle) \cdot \mathbf{b}_i/\|\widetilde{\mathbf{b}_i}\| - v_i\overline{\mathbf{b}_i}$.
2. Output $\mathbf{v} := \sum_{i=1}^n v_i \overline{\mathbf{b}_i}$.

It is easy to verify that the procedure always outputs vectors in the coset $\Lambda+\mathbf{c}$. Moreover, the probability of outputting any $\mathbf{v} \in \Lambda + \mathbf{c}$ is

$$\prod_{i=1}^n \frac{\rho_r(v_i)}{\rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z} + \langle \mathbf{c}_i, \overline{\mathbf{b}_i}\rangle)} = \frac{\rho_r(\mathbf{v})}{\prod_{i=1}^n \rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z} + \langle \mathbf{c}_i, \overline{\mathbf{b}_i}\rangle)},$$

where $\mathbf{c}_i$ are the values computed in the procedure when it outputs $\mathbf{v}$. Notice that by Lemma 2.5 and our assumption on r, we have that $r \geq \eta_{1/(n+1)}(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z})$ for all i. Therefore, by Lemma 2.7, we have that for all $c \in \mathbb{R}$,

$$\rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z} + c) \in \left[1 - \frac{2}{n+2}, 1\right] \rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z}).$$

In order to get an exact sample, we combine the above procedure with rejection sampling. Namely, we apply SampleD to obtain some vector $\mathbf{v}$. We then output $\mathbf{v}$ with probability

$$\frac{\prod_{i=1}^n \rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z} + \langle \mathbf{c}_i, \overline{\mathbf{b}_i} \rangle)}{\prod_{i=1}^n \rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z})} \in \left(\left(1 - \frac{2}{n+2}\right)^n, 1 \right] \subseteq (e^{-2}, 1], \tag{5.1}$$

and otherwise repeat. This probability can be efficiently computed, as we will show below. As a result, in any given iteration the probability of outputting the vector $\mathbf{v} \in \Lambda + \mathbf{c}$ is

$$\frac{\rho_r(\mathbf{v})}{\prod_{i=1}^n \rho_r(\|\widetilde{\mathbf{b}_i}\|\mathbb{Z})}.$$

Since the denominator is independent of $\mathbf{v}$, we obtain that in any given iteration, conditioned on outputting something, the output is distributed according to the desired distribution $D_{\Lambda+\mathbf{c},r}$, and therefore this is also the overall output distribution of our sampler. Moreover, by (5.1), the probability of outputting something in any given iteration is at least e^{-2}, and therefore, the probability that the number of iterations is greater than t is at most $(1 - e^{-2})^t$, and in particular, the expected number of iterations is at most e^2.

It remains to show how to efficiently compute the probability in (5.1). By scaling, it suffices to show how to compute

$$\rho_r(\mathbb{Z} + c) = \sum_{k \in \mathbb{Z}} \exp(-\pi(k+c)^2/r^2)$$

for any $r > 0$ and $c \in [0, 1)$. If $r < 1$, the sum decays very fast, and we can achieve any desired t bits of accuracy in time $\mathrm{poly}(t)$, which agrees with our notion of efficiently computing a real number (following, e.g., the treatment in [Lov86, Section 1.4]). For $r \geq 1$, we use the Poisson summation formula (see, e.g., [MR04, Lemma 2.8]) to write

$$\rho_r(\mathbb{Z} + c) = r \cdot \sum_{k \in \mathbb{Z}} \exp(-\pi k^2 r^2 + 2\pi i c k) = r \cdot \sum_{k \in \mathbb{Z}} \exp(-\pi k^2 r^2) \cos(2\pi c k),$$

which again decays fast enough so we can compute it to within any desired t bits of accuracy in time $\mathrm{poly}(t)$.

Acknowledgments: We thank Elette Boyle, Adam Klivans, Vadim Lyubashevsky, Sasha Sherstov, Vinod Vaikuntanathan and Gilles Villard for useful discussions.

References

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115. 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.

[AD87] D. Aldous and P. Diaconis. Strong uniform times and finite random walks. *Adv. in Appl. Math.*, 8(1):69–97, 1987. ISSN 0196-8858. doi:10.1016/0196-8858(87)90006-6.

[AD97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293. 1997.

[AGV09] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495. 2009.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 1–32. Dept. Math., Seconda Univ. Napoli, Caserta, 2004. Preliminary version in STOC 1996.

[AP12] J. Alperin-Sheriff and C. Peikert. Circular and KDM security for identity-based encryption. In *Public Key Cryptography*, pages 334–352. 2012.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. 2012.

[Boy10] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517. 2010.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012.

[BS96] E. Bach and J. Shallit. *Algorithmic number theory. Vol. 1*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1996.

[BV96] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 1996. ISBN 3-540-61512-1.

[BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. 2011.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010.

[Dev86] L. Devroye. *Nonuniform random variate generation*. Springer-Verlag, New York, 1986. Available at `http://luc.devroye.org/rnbookindex.html`.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GGH96] O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(42), 1996.

[GHPS12] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN*, pages 19–37. 2012.

[GKPV10] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. 2010.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Kho10] S. Khot. Inapproximability results for computational problems on lattices. In P. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*. Springer-Verlag, New York, 2010.

[KS06] A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *FOCS*, pages 553–562. 2006.

[KTX08] A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389. 2008.

[Kup05] G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005.

[LM06] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155. 2006.

[LM09] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, pages 577–594. 2009.

[Lov86] L. Lovász. *An algorithmic theory of numbers, graphs and convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1986.

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339. 2011.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23. 2010.

[Lyu08] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, pages 162–179. 2008.

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. 2012.

[MM11] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484. 2011.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[MV03] D. Micciancio and S. P. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO*, pages 282–298. 2003.

[OPW11] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In *CRYPTO*, pages 525–542. 2011.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[PR06] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166. 2006.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571. 2008.

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196. 2008.

[Reg02] O. Regev. Quantum computation and lattice problems. *SIAM J. Comput.*, 33(3):738–760, 2004. Preliminary version in FOCS 2002.

[Reg03] O. Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004. Preliminary version in STOC 2003.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[Reg10a] O. Regev. The learning with errors problem. In *Proc. of 25th IEEE Annual Conference on Computational Complexity (CCC)*, pages 191–204. 2010.

[Reg10b] O. Regev. On the complexity of lattice problems with polynomial approximation factors. In P. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*. Springer-Verlag, New York, 2010.

How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE

Rikke Bendlin* Sara Krehbiel† Chris Peikert‡

Abstract

We develop secure *threshold* protocols for two important operations in lattice cryptography, namely, generating a hard lattice Λ together with a "strong" trapdoor, and sampling from a discrete Gaussian distribution over a desired coset of Λ using the trapdoor. These are the central operations of many cryptographic schemes: for example, they are exactly the key-generation and signing operations (respectively) for the GPV signature scheme, and they are the public parameter generation and private key extraction operations (respectively) for the GPV IBE. We also provide a protocol for trapdoor delegation, which is used in lattice-based hierarchical IBE schemes. Our work therefore directly transfers all these systems to the threshold setting.

Our protocols provide information-theoretic (i.e., statistical) security against adaptive corruptions in the UC framework, and they are private and robust against an optimal number of semi-honest or malicious parties. Our Gaussian sampling protocol is both noninteractive and efficient, assuming either a trusted setup phase (e.g., performed as part of key generation) or a sufficient amount of interactive but offline precomputation, which can be performed before the inputs to the sampling phase are known.

*Department of Computer Science, Aarhus University. Email: rikkeb@cs.au.dk. Supported by the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council). Part of this work was performed while visiting the Georgia Institute of Technology.

†School of Computer Science, College of Computing, Georgia Institute of Technology. Email: sarak@gatech.edu

‡School of Computer Science, College of Computing, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. Supported by the Alfred P. Sloan Foundation and the National Science Foundation under CAREER Award CCF-1054495. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

A *threshold* cryptographic scheme [DF89] is one that allows any quorum of h out of ℓ trustees to jointly perform some privileged operation(s) by following a specified protocol, and remains correct and secure even if up to some $t < h$ of the parties deviate from the protocol adversarially. For example, in a threshold signature scheme any h trustees can sign an agreed-upon message, and no t malicious players (who may even pool their knowledge and coordinate their actions) can prevent the signature from being produced, nor forge a valid signature on a new message. Similarly, a threshold encryption scheme requires at least h trustees to decrypt a ciphertext. Threshold cryptography is very useful for both distributing trust and increasing robustness in systems that perform high-value operations, such as certificate authorities (CAs) or private-key generators in identity-based encryption (IBE) systems.

Desirable efficiency properties in a threshold system include: (1) efficient local computation by the trustees; (2) low interaction—e.g., one broadcast message from each party—when performing the privileged operations; and (3) key sizes and public operations that are independent of the number of trustees. For example, while it might require several parties to *sign* a message, it is best if the signature can be *verified* without even being aware that it was produced in a distributed manner.

Over the years many elegant and rather efficient threshold systems have been developed. To name just a few representative works, there are simple variants of the ElGamal cryptosystem, Canetti and Goldwasser's [CG99] version of the CCA-secure Cramer-Shoup cryptosystem [CS98], and Shoup's [Sho00] version of the RSA signature scheme. These systems, along with almost all others in the literature, are based on number-theoretic problems related to either integer factorization or the discrete logarithm problem in cyclic groups. As is now well-known, Shor's algorithm [Sho97] would unfortunately render all these schemes insecure in a "post-quantum" world with large-scale quantum computers.

Lattice-based cryptography. Recently, *lattices* have been recognized as a viable foundation for quantum-resistant cryptography, and the past few years have seen the rapid growth of many rich lattice-based systems. A fruitful line of research, starting from the work of Gentry, Peikert and Vaikuntanathan (GPV) [GPV08], has resulted in secure lattice-based hash-and-sign signatures and (hierarchical) identity-based encryption schemes [CHKP10, ABB10], along with many more applications (e.g., [GKV10, BF11b, BF11a, AFV11]). All these schemes rely at heart on two nontrivial algorithms: the key-generation algorithm produces a lattice Λ together with a certain kind of "strong" trapdoor (e.g., a short basis of Λ) [Ajt99, AP09], while the signing/key-extraction algorithms use the trapdoor to randomly sample a short vector from a *discrete Gaussian distribution* over a certain coset $\Lambda + \mathbf{c}$, which is determined by the message or identity [GPV08]. Initially, both tasks were rather complicated algorithmically, and in particular the Gaussian sampling algorithm involved several adaptive iterations, so it was unclear whether either task could be efficiently and securely distributed among several parties. Recently, however, both key generation and Gaussian sampling have been simplified and made more efficient and parallel [Pei10, MP12]. This is the starting point for our work.

Our results. We give threshold protocols for the main nontrivial operations in lattice-based signature and (H)IBE schemes, namely: (1) generating a lattice Λ together with a strong trapdoor of the kind recently proposed in [MP12], (2) sampling from a discrete Gaussian distribution over a desired coset of Λ, and (3) delegating a trapdoor for a higher-dimensional extension of Λ. Since these are the only secret-key operations used in the signature and (H)IBE schemes of [GPV08, CHKP10, ABB10, MP12] and several other related works, our protocols can be plugged directly into all those schemes to distribute the signing algorithms and the (H)IBE private-key generators. In Section 4 we show how this is (straightforwardly) done for the simplest of these applications, namely, the GPV signature scheme [GPV08]; the GPV IBE scheme and other applications work similarly.

Our protocols have several desirable properties:

- They provide *information-theoretic* (i.e., statistical) security for *adaptive* corruptions. By information-theoretic security, we mean that the security of the key-generation and sampling protocols *themselves* relies on no computational assumption—instead, the application alone determines the assumption (usually, the Short Integer Solution assumption [Ajt96, MR04] for digital signatures, and Learning With Errors [Reg05] for identity-based encryption). We work in a version of the universal composability (UC) framework [Can01], specialized to the threshold setting, and as a result also get strong security guarantees for protocols under arbitrary composition.
- They are secure for an optimal threshold of semi-honest or active (malicious) parties, which is determined by the precise communication model and setup assumption. For example, we can tolerate $h - 1$ semi-honest parties assuming trusted setup (see below), or $t = h - 1$ malicious parties in a model with both broadcast and private channels, using the verifiable secret sharing scheme of [RB89]. (Recall that h is the number of (semi-)honest parties the protocol requires to execute successfully, and the robustness threshold t is an upper bound on the number of malicious parties.)
- The public key and trapdoor "quality" (i.e., the width of the discrete Gaussian that can be sampled using the trapdoor; smaller width means higher quality) are essentially the same as in the standalone setting. In particular, their sizes are independent of the number of trustees; the individual shares of the trapdoor are the same size as the trapdoor itself; and the protocols work for the same lattice parameters as in the standalone setting, up to small constant factors.
- They have *noninteractive* and very efficient *online* phases (corresponding to the signing or key-extraction operations), assuming either (1) a setup phase in which certain shares are distributed by a trusted party (e.g., as part of key generation), or (2) the parties themselves perform a sufficient amount of interactive precomputation in an offline phase (without relying on any trusted party). We provide protocols for these two settings in Section 3 and Appendix A, respectively.

Regarding the final item, the trusted setup model is the one used by Canetti and Goldwasser [CG99] for constructing threshold chosen ciphertext-secure threshold cryptosystems: as part of the key-generation process, a trusted party also distributes shares of some appropriately distributed secrets to the parties, which they can later use to perform an *a priori* bounded number of noninteractive threshold operations. Or, in lieu of a trusted party, the players can perform some interactive precomputation (offline, before the desired coset is known) to generate the needed randomness. The downside is that this precomputation is somewhat expensive, since the only solution we have for one important step (namely, sampling shares of a Gaussian-distributed value over $\mathbb{Z}$) is to use somewhat generic information-theoretic multiparty computation tools. On the plus side, the circuit for this sampling task is rather shallow, with depth just slightly super-constant $\omega(1)$, so the round complexity of the precomputation is not very high. We emphasize that the expensive precomputation is executed offline, before the applications decides which lattice cosets will be sampled from, and that the online protocols remain efficient and non-interactive.

Our protocols rely on the very simple form of the new type of strong trapdoor recently proposed in [MP12], and the parallel and offline nature of recent standalone Gaussian sampling algorithms [Pei10, MP12].[1] A key technical challenge is that the security of the sampling algorithms from [Pei10, MP12] crucially relies on the secrecy of some intermediate random variables known as "perturbations." However, in order to obtain a noninteractive protocol we need the parties to publicly reveal certain information about these perturbations. Fortunately, we can show that the leaked information is indeed simulatable, and so security is unharmed. See Section 3 and in particular Lemma 3.2 for further details.

[1] In particular, it appears very difficult to implement, in a noninteractive threshold fashion, iterative sampling algorithms like those from [Kle00, GPV08] which use the classical trapdoor notion of a short basis.

Open problems. In addition to simple, non-interactive protocols for discrete Gaussian sampling with trusted setup, in Appendix A we give efficient protocols for discrete Gaussian sampling that avoid both trusted setup and online interaction by using (offline) access to a functionality $\mathcal{F}_{\text{Samp}\mathbb{Z}}$, which produces shares of Gaussian-distributed values over the integers $\mathbb{Z}$ (see Appendix A.1 for details). We show how to instantiate $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ using a (somewhat inefficient) interactive protocol using generic MPC techniques. It remains an interesting open problem to design discrete Gaussian sampling protocols without trusted setup whose *offline precomputation* is efficient and/or non-interactive as well. An efficient realization of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ would yield such a solution, but there may be other routes as well.

Another intriguing problem is to give a simple and noninteractive threshold protocol for inverting the LWE function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$ (for short error vector $\mathbf{e}$) using a shared trapdoor. We find it surprising that, while in the standalone setting this inversion task is conceptually and algorithmically much simpler than Gaussian sampling, we have not yet been able to find a simple threshold protocol for it.[2] Such a protocol could, for example, be useful for obtaining threshold analogues of the chosen ciphertext-secure cryptosystems from [Pei09, MP12], without going through a generic IBE-to-CCA transformation [BCHK07].

Related work in threshold lattice cryptography. A few works have considered lattice cryptography in the threshold setting. For encryption schemes, Bendlin and Damgård [BD10] gave a threshold version of Regev's CPA-secure encryption scheme based on the learning with errors (LWE) problem [Reg05]. Related work by Myers *et al.* [MSs11] described threshold decryption for fully homomorphic cryptosystems. Xie *et al.* [XXZ11] gave a threshold CCA-secure encryption scheme from any lossy trapdoor function (and hence from lattices/LWE [PW08]), though its public key and encryption runtime grow at least linearly with the number of trustees. For signatures, Feng *et al.* [FGM10] gave a threshold signature scheme where signing proceeds sequentially through each trustee, making the scheme highly interactive; also, the scheme is based on NTRUSign, which has been broken [NR06]. Cayrel *et al.* [CLRS10] gave a lattice-based threshold *ring* signature scheme, in which at least t trustees are needed to create an *anonymous* signature. In that system, each trustee has its own public key, and verification time grows linearly with the number of trustees. In summary, lattice-based threshold schemes to date have either been concerned with distributing the *decryption* operation in public-key cryptosystems, and/or have lacked key efficiency properties typically asked of threshold systems (which our protocols do enjoy). Also, other important applications such as (H)IBE have yet to be realized in a threshold manner.

Organization. The remainder of the paper is organized as follows. In Section 2 we overview the relevant background on lattices, secret sharing, and the UC framework. In Section 3 we review the standalone key-generation and discrete Gaussian sampling algorithms of [MP12], present our functionalities for these algorithms in the threshold setting, and show how these functionalities can be implemented efficiently and noninteractively using trusted setup. At the end of Section 3 we additionally provide a functionality and protocol for trapdoor delegation. Finally, in Section 4 we detail a simple example application of our protocols, namely, a threshold version of the GPV signature scheme [GPV08] realizing the threshold signature functionality of [ADN06]. In the appendix, we remove the trusted setup assumption and show how to instead use offline interaction to implement all our functionalities.

2 Preliminaries

We denote the reals by $\mathbb{R}$ and the integers by $\mathbb{Z}$. For a positive integer ℓ, we let $[\ell] = \{1, \ldots, \ell\}$.

[2] We note that it is possible to give a threshold protocol using a combination of Gaussian sampling and trapdoor delegation [CHKP10, MP12], but it is obviously no simpler than Gaussian sampling alone.

A square symmetric real matrix Σ is *positive definite*, written $\Sigma > \mathbf{0}$, if $\mathbf{x}^t \Sigma \mathbf{x} > 0$ for all nonzero $\mathbf{x}$. Positive definiteness defines a partial ordering on symmetric matrices: we say that $\Sigma_1 > \Sigma_2$ if $(\Sigma_1 - \Sigma_2) > \mathbf{0}$. For any nonsingular matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$, the symmetric matrix $\Sigma = \mathbf{B}\mathbf{B}^t$ is positive definite. We say that $\mathbf{B}$ is a *square root* of $\Sigma > \mathbf{0}$, written $\mathbf{B} = \sqrt{\Sigma}$, if $\mathbf{B}\mathbf{B}^t = \Sigma$. Every $\Sigma > \mathbf{0}$ has a square root; moreover, the square root is unique up to right-multiplication by an orthogonal matrix, i.e., $\mathbf{B}' = \sqrt{\Sigma}$ if and only if $\mathbf{B}' = \mathbf{B}\mathbf{Q}$ for some orthogonal matrix $\mathbf{Q}$. A square root can be computed efficiently using, e.g., the Cholesky decomposition. The largest singular value (also called spectral norm or operator norm) of a real matrix $\mathbf{X}$ is defined as $s_1(\mathbf{X}) = \max_{\mathbf{u} \neq \mathbf{0}} \|\mathbf{X}\mathbf{u}\| / \|\mathbf{u}\|$. For convenience, we sometime write a scalar s to mean the scaled identity matrix $s\mathbf{I}$, whose dimension will be clear from context.

2.1 Continuous Gaussians

The n-dimensional Gaussian function $\rho \colon \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle).$$

Applying a linear transformation given by a nonsingular real matrix $\mathbf{B}$ yields the Gaussian function

$$\rho_{\mathbf{B}}(\mathbf{x}) := \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp\left(-\pi \cdot \left\langle \mathbf{B}^{-1}\mathbf{x}, \mathbf{B}^{-1}\mathbf{x} \right\rangle\right) = \exp\left(-\pi \cdot \mathbf{x}^t \Sigma^{-1} \mathbf{x}\right),$$

where $\Sigma = \mathbf{B}\mathbf{B}^t > \mathbf{0}$. Because $\rho_{\mathbf{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

Normalizing $\rho_{\sqrt{\Sigma}}$ by its total measure $\int_{\mathbb{R}^n} \rho_{\sqrt{\Sigma}}(\mathbf{x})\, d\mathbf{x} = \sqrt{\det \Sigma}$ over $\mathbb{R}^n$, we obtain the probability distribution function of the (continuous) *Gaussian distribution* $D_{\sqrt{\Sigma}}$. It is easy to check that a random variable $\mathbf{x}$ having distribution $D_{\sqrt{\Sigma}}$ can be written as $\sqrt{\Sigma} \cdot \mathbf{z}$, where $\mathbf{z}$ has spherical Gaussian distribution D_1. Therefore, the random variable $\mathbf{x}$ has *covariance*

$$\mathop{\mathrm{E}}_{\mathbf{x} \sim D_{\sqrt{\Sigma}}} \left[\mathbf{x} \cdot \mathbf{x}^t\right] = \sqrt{\Sigma} \cdot \mathop{\mathrm{E}}_{\mathbf{z} \sim D_1} \left[\mathbf{z} \cdot \mathbf{z}^t\right] \cdot \sqrt{\Sigma}^t = \sqrt{\Sigma} \cdot \frac{\mathbf{I}}{2\pi} \cdot \sqrt{\Sigma}^t = \frac{\Sigma}{2\pi},$$

by linearity of expectation. (The $\mathbf{I}/(2\pi)$ covariance of $\mathbf{z} \sim D_1$ arises from the independence of its entries, which are each distributed as D_1 in one dimension, and therefore have variance $1/(2\pi)$.) For convenience, in this paper we implicitly scale all covariance matrices by a 2π factor, and refer to Σ as the covariance matrix of $D_{\sqrt{\Sigma}}$.

2.2 Lattices and Discrete Gaussians

A *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^m$ for some $m \geq 0$. In this work we are only concerned with full-rank integer lattices, which are additive subgroups of $\mathbb{Z}^m$ with finite index. Most recent cryptographic applications use a particular family of so-called *q-ary* integer lattices, which contain $q\mathbb{Z}^m$ as a sublattice for some integer q, which in this work will always be bounded by $\mathrm{poly}(n)$. For positive integers n and q, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary, and define the full-rank m-dimensional q-ary lattice

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \bmod q\}.$$

For any $\mathbf{u} \in \mathbb{Z}_q^n$ admitting an integral solution $\mathbf{x} \in \mathbb{Z}^m$ to $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q$, define the coset (or shifted lattice)

$$\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) = \Lambda^{\perp}(\mathbf{A}) + \mathbf{x} = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \bmod q\}.$$

Note that for $n, m, q \leq 2$ and $m > Cn \log q$ for some fixed constant $C > 1$, the columns of a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ generate all of $\mathbb{Z}_q^n$ with all but $\mathrm{negl}(n)$ probability.

Let $\Lambda \subset \mathbb{R}^m$ be a lattice, let $\mathbf{c} \in \mathbb{R}^m$, and let $\Sigma > \mathbf{0}$ be a positive definite matrix. The *discrete Gaussian distribution* $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$ is simply the Gaussian distribution $D_{\sqrt{\Sigma}}$ restricted so that its support is the coset $\Lambda + \mathbf{c}$. That is, for all $\mathbf{x} \in \Lambda + \mathbf{c}$,

$$D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x})}{\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c})} \propto \rho_{\sqrt{\Sigma}}(\mathbf{x}).$$

A discrete Gaussian is said to be *spherical* with parameter $s > 0$ if its covariance matrix is $s^2\mathbf{I}$.

We recall an important definition and some useful properties of discrete Gaussian distributions on lattices. For $\epsilon > 0$, the *smoothing parameter* [MR04] $\eta_\epsilon(\Lambda)$ of a lattice Λ is defined as the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$, where Λ^* is the dual lattice (whose precise definition we will not need here). Here we generalize the smoothing parameter to non-spherical Gaussians; note that this definition is consistent with the partial ordering on positive definite matrices.

Definition 2.1 (Smoothing parameter). *Let $\Sigma > \mathbf{0}$ be any positive definite matrix. We say that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ if $\eta_\epsilon(\sqrt{\Sigma}^{-1} \cdot \Lambda) \leq 1$, or equivalently, if $\rho_{\sqrt{\Sigma^{-1}}}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$.*

The following lemma is a slight generalization of [Reg05, Claim 3.8] (see also [MR04, Lemma 4.1]) to non-spherical Gaussians, obtained by applying a linear transformation to the Gaussian function and lattice. Informally, it says that every coset of Λ has essentially the same mass under $\rho_{\sqrt{\Sigma}}$, when $\sqrt{\Sigma}$ exceeds the smoothing parameter of Λ. The corollary then follows by a routine calculation.

Lemma 2.2. *For any m-dimensional lattice Λ, real $\epsilon > 0$, $r \geq \eta_\epsilon(\Lambda)$, and $\mathbf{c} \in \mathbb{R}^m$, we have $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \in [1 \pm \epsilon] \cdot Z_{\Lambda,r}$, where $Z_{\Lambda,r}$ depends only on Λ and r (not $\mathbf{c}$).*

Corollary 2.3. *Let $\Lambda' \subseteq \Lambda$ be full-rank lattices, and let $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda')$ for some $\epsilon > 0$. For $\mathbf{x} \leftarrow D_{\Lambda,\sqrt{\Sigma}}$, the marginal distribution of $\mathbf{c} = \mathbf{x} \bmod \Lambda'$ is within statistical distance $\epsilon/2$ from uniform over Λ/Λ', and the conditional distribution of $\mathbf{x}$ given $\mathbf{c}$ is $D_{\Lambda'+\mathbf{c},\sqrt{\Sigma}}$.*

The following special case of [MP12, Lemma 2.4] says that for uniformly random $\mathbf{A}$ and appropriate parameters, the lattice $\Lambda^\perp(\mathbf{A})$ has small smoothing parameter with very high probability.

Lemma 2.4. *Let $n, m, q \geq 2$ be positive integers and $C > 1$ be a fixed constant such that $m > Cn \log q$, and let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be uniformly random. For any fixed $\omega_n = \omega(\sqrt{\log n})$ there exists some $\epsilon = \mathrm{negl}(n)$ such that $\eta_\epsilon(\Lambda^\perp(\mathbf{A})) \leq \omega_n$ except with probability $2^{-\Omega(n)}$.*

Finally, we need the "convolution lemma" of [Pei10, Theorem 3.1].

Lemma 2.5. *Let $\Sigma_1, \Sigma_2 > \mathbf{0}$ be positive definite matrices, with $\Sigma = \Sigma_1 + \Sigma_2 > \mathbf{0}$ and $\Sigma^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} > \mathbf{0}$. Let Λ_1, Λ_2 be lattices such that $\sqrt{\Sigma_1} \geq \eta_\epsilon(\Lambda_1)$ and $\sqrt{\Sigma_2} \geq \eta_\epsilon(\Lambda_2)$ for some positive $\epsilon \leq 1/2$, and let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^m$ be arbitrary. In the following experiment:*

$$\textit{choose } \mathbf{x}_2 \leftarrow D_{\Lambda_2+\mathbf{c}_2,\sqrt{\Sigma_2}}\textit{, then choose } \mathbf{x}_1 \leftarrow \mathbf{x}_2 + D_{\Lambda_1+\mathbf{c}_1-\mathbf{x}_2,\sqrt{\Sigma_1}},$$

the marginal distribution of $\mathbf{x}_1$ is within statistical distance 8ϵ of $D_{\Lambda_1+\mathbf{c}_1,\sqrt{\Sigma}}$.

Throughout the paper we often attach a factor $\omega_n = \omega_n(n) = \omega(\sqrt{\log n})$, which represents an arbitrary fixed function that grows asymptotically faster than $\sqrt{\log n}$, to Gaussian parameters $\sqrt{\Sigma}$ (or ω_n^2 to covariance matrices Σ). In exposition we usually omit reference to these factors, but we always retain them where needed in formal expressions.

2.3 Secret Sharing for Additive Groups

In this work we will need to distribute secret lattice points among multiple players, so that any sufficiently large number of players is able to reconstruct the points, but smaller subsets collectively get no information about the secret. Because a lattice Λ is an *infinite* additive group (and in particular is not a field), it is not immediately amenable to standard secret-sharing techniques like those of [Sha79]. Fortunately, for our purposes it will suffice to share elements of a suitable *finite* quotient group $G = \Lambda/\bar{\Lambda}$, where the sublattice $\bar{\Lambda} \subseteq \Lambda$ is "sparse" enough in Λ that an element of G identifies an element of Λ with enough specificity for our applications. There is a rich theory of secret sharing for arbitrary additive groups and modules, e.g., [DF94, Feh98]. Here we recall the relevant material in enough generality for our purposes.

Let G be a finite abelian (additive) group with identity element 0. The *exponent* of G, denoted $e(G)$, is the smallest positive integer m such that $mg = g + g + \cdots + g = 0$ for every $g \in G$. We have that G is a module over the ring $R = \mathbb{Z}_{e(g)}$, which gives the following form of Shamir's $(t+1)$-out-of-ℓ secret-sharing scheme [Sha79]. Let $t < \ell$ be positive integers, where t denotes a bound on the number of corrupt players out of ℓ total. Suppose that G is an R-module for some ring R which has efficiently computable operations and $\ell + 1$ known elements $U = \{r_0 = 0, r_1, \ldots, r_\ell\} \subseteq R$ such that $r_i - r_j$ is invertible in R (i.e., a unit) for every $i \neq j$. For example, in our protocols we will have $e(G) = q^d$ for some public integers $q \geq 2$ and $d \geq 1$, so we can take $R = \mathbb{Z}_{q^d}$ and $r_i = i \bmod q^d$, as long as ℓ is smaller than every prime divisor of q. When this condition does not hold, we can use an extension ring instead, as described below.

To share a value $g \in G$, one chooses a formal polynomial $f(X) = \sum_{j=0}^{t} f_j X^j \in G[X]$ of degree at most t, where $f_0 = g$ and the $f_i \in G$ for $i \geq 1$ are uniformly random and independent. Player $i \in [\ell]$ is publicly associated with the value $r_i \in R$, and gets the share $s_i = f(r_i) = \sum_{j=0}^{t-1} r_i^j f_j \in G$. Usually we let f be implicit, denoting the ith player's share as $[\![g]\!]^i$ and the tuple of all shares by $[\![g]\!]$. Note that the product group G^k is also an additive group with exponent $e(G)$, so we can share vectors or matrices with entries in G as above, using the same ring $\mathbb{Z}_{e(G)}$. (Equivalently, this is just an independent entry-wise sharing.)

The above scheme has several important properties (whose proofs are straightforward; see, e.g., [Feh98]):

- It is *ideal*: the shares $s_i \in G$ belong to the same set as the shared value $g \in G$.
- It is *perfectly secret*: for any shared value $g \in G$, any tuple of up to t shares s_i is distributed uniformly.
- It is *perfectly correct* and *robust*: any $t+1$ shares s_i of g (along with their corresponding evaluation points r_i) can be used to efficiently recover $f(X)$, and hence $g = f_0 = f(0)$, by interpolation.

  Moreover, given at least $3t+1$ values s'_i (along with the corresponding evaluation points r_i), where at least $2t+1$ are correct shares $s'_i = f(r_i)$ of g and the remaining t may be arbitrary, one can efficiently recover $f(X)$ and hence $g = f_0$ using, e.g., the Welch-Berlekamp algorithm for unambiguous decoding of Reed-Solomon codes. (The algorithm is usually described for codes defined over finite fields, but its proof of correctness goes through without modification in our setting.)
- It is *homomorphic*: if $g, g' \in G$ have respective shares $s_i = f(r_i), s'_i = f'(r_i)$ for $i \in [\ell]$, then $s_i + s'_i = (f + f')(r_i)$ and $rs_i = (rf)(r_i)$ are respective shares of $g + g'$ and rg for any $r \in R$. Moreover, let $G' \subseteq G$ be a subgroup; then $\bar{s}_i = s_i \bmod G'$ are shares of $\bar{g} = g \bmod G'$, via the polynomial $\bar{f}(X) = f(X) \bmod G'[X]$. Additionally, if $g \in G'$, then $s_i - \bar{s}_i \in G'$ are shares of g.

Secret sharing with extension rings. The above scheme works when the number of parties ℓ is less than every prime divisor of $e(G)$. When this is not the case (e.g., when using $q = 2^k$, which is a convenient choice for the trapdoor construction described in Section 3.1), we can instead share elements from the vector group G^k, which is a module over a certain *extension ring* of $\mathbb{Z}_{e(G)}$ that has a suitable set U of size p^k, where p is the smallest prime divisor of $e(G)$. By choosing $k \geq \log_p(\ell + 1)$, we can share elements of G among ℓ players using shares in G^k, or even amortize the sharing of up to k elements in G at a time.

In brief, we use the extension ring $R = \mathbb{Z}_{e(G)}[X]/F(X)$ for any monic degree-k polynomial $F(X) = \sum_{i=0}^{k} F_i X^i \in \mathbb{Z}_{e(G)}[X]$ that is irreducible modulo every prime dividing $e(G)$. Then it can be verified that G^k is an R-module, where multiplication $R \times G^k \to G^k$ is defined by the rule $X \cdot (g_0, \ldots, g_{k-1}) = (0, g_0, \ldots, g_{k-2}) - (F_0 \cdot g_{k-1}, \ldots, F_{k-1} \cdot g_{k-1})$. An element of R is a unit if and only if it is nonzero modulo every prime integer divisor of $e(G)$, so letting p be the smallest such divisor, the polynomial residues in R with coefficients in $\{0, \ldots, p-1\}$ give us p^k elements $r_i \in R$ such that $r_i - r_j$ is a unit for all $i \neq j$, as needed. See, for example, [DF94] or [Feh98, Chapter 3] for full details.

Verifiable secret sharing. To recover a shared value, our protocols instruct honest parties to broadcast their respective shares and then reconstruct the value from the announced shares. As mentioned above, the Welch-Berlekamp algorithm efficiently reconstructs the shared value given at least $2t + 1$ correct shares and up to t incorrect ones (which may come from malicious parties), which means we can tolerate any $t < \ell/3$ malicious parties. Assuming appropriate communication channels, it is possible to improve this threshold to any $t < \ell/2$ malicious parties by using a verifiable secret sharing (VSS) protocol, e.g., the one of [RB89]. The share-distribution and reconstruction steps of our protocols can be straightforwardly modified to use VSS, but we omit these modifications for simplicity of exposition.

2.4 UC Framework

We frame our results in the Universal Composability (UC) framework [Can00, Can01]. In the UC framework, security is defined by considering a probabilistic polynomial-time (PPT) machine $\mathcal{Z}$, called the environment. In coordination with an adversary that may corrupt some of the players, $\mathcal{Z}$ chooses inputs and observes the outputs of a protocol executed in one of two worlds: a "real" world in which the parties interact with each other in some specified protocol π while a dummy adversary $\mathcal{A}$ (controlled by $\mathcal{Z}$) corrupts players and controls their interactions with honest players, and an "ideal" world in which the players interact directly with a *functionality* $\mathcal{F}$, while a simulator $\mathcal{S}$ (communicating with $\mathcal{Z}$) corrupts players and controls their interactions with $\mathcal{F}$. The views of the environment in these executions are respectively denoted $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, and the protocol is said to realize the functionality if these two views are indistinguishable. In this work we are concerned solely with statistical indistinguishability (which is stronger than the computational analogue), denoted by the relation $\overset{s}{\approx}$.

Definition 2.6. *A protocol* π statistically realizes *a functionality* $\mathcal{F}$ *(or alternatively, is a UC-secure implementation of* $\mathcal{F}$*) if for any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *such that for all PPT environments* $\mathcal{Z}$*, we have* $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} \mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.

The *universal composition theorem* [Can01] informally states that any UC-secure protocol remains secure under concurrent general composition. This allows for the modular design of functionalities and protocols which can be composed to produce secure higher-level protocols. Our functionalities implicitly use standard conventions like delayed public and private outputs, corruptions, etc, which are addressed in detail in [Can00, Can01].

UC framework for threshold protocols. We consider a specialized case of the UC framework that is appropriate for modeling threshold protocols. All of our functionalities are called with a session ID of the form $sid = (\mathcal{P}, sid')$, where $\mathcal{P}$ is a set of ℓ parties representing the individual trustees in the threshold protocol. We prove security against adversaries that may adaptively corrupt a certain bounded number of the parties over the entire lifetime of a protocol, and consider both the semi-honest case (in which corrupted parties still execute the protocol faithfully) and the malicious case. At the time of corruption, the entire

view of the player to that point (and beyond) is revealed to the adversary; in particular, we do not assume secure erasures. For robustness, we additionally require that when the environment issues a command to a functionality/protocol, it always does so for at least h honest parties in the same round.

Many of our protocols require the parties to maintain and use consistent local states, corresponding to certain shared random variables that are consumed by the protocols. We note that synchronizing their local states may be nontrivial, if not every party is involved with executing every command. For this reason we assume some mechanism for coordinating local state, such as hashing as suggested in [CG99], which deals with similar synchronization issues.

3 Threshold Key Generation, Gaussian Sampling, and Trapdoor Delegation

In this section, we present UC functionalities and protocols for generating a lattice with a shared trapdoor, for sampling from a coset of that lattice, and for securely delegating a trapdoor of a higher-dimensional extension of the lattice. As an example application of these functionalities, in Section 4 we describe threshold variants of the GPV signature and IBE schemes of [GPV08]; other signature and (H)IBE schemes (e.g., [CHKP10, ABB10, MP12]) can be adapted similarly (where delegation is needed for HIBE).

We start in Section 3.1 by recalling the recent standalone (non-threshold) key generation and discrete Gaussian sampling algorithms of [MP12], which form the basis of our protocols. In Section 3.2 we present the two main functionalities $\mathcal{F}_{\text{KG}}$ (key generation) and $\mathcal{F}_{\text{GS}}$ (Gaussian sampling) corresponding to the standalone algorithms.

- Since key generation tends to be rare in applications, $\mathcal{F}_{\text{KG}}$ can be realized using trusted setup; alternatively, in Appendix A we realize $\mathcal{F}_{\text{KG}}$ via an interactive protocol without trusted setup.
- To realize $\mathcal{F}_{\text{GS}}$, we define two lower-level "helper" functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$, and give in Section 3.3 an efficient noninteractive protocol that realizes $\mathcal{F}_{\text{GS}}$ using access to them. Section 3.2 realizes the helper functionalities noninteractively using trusted setup, and Appendix A realizes them using offline precomputation (instead of trusted setup).
- Finally, in Section 3.4 we give a functionality $\mathcal{F}_{\text{DelTrap}}$ and protocol for trapdoor delegation.

3.1 Trapdoors and Standalone Algorithms

We recall the notion of a (strong) lattice trapdoor and associated algorithms recently introduced by Micciancio and Peikert [MP12] (see that paper for full details and proofs). Let n and q be positive integers and $k = \lceil \lg q \rceil$. Define the "gadget" vector $\mathbf{g} = (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$ and matrix

$$\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t = \begin{bmatrix} \cdots \mathbf{g}^t \cdots & & & \\ & \cdots \mathbf{g}^t \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^t \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}.$$

The k-dimensional lattice $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$, and hence also the nk-dimensional lattice $\Lambda^\perp(\mathbf{G})$, has smoothing parameter bounded by $s_{\mathbf{g}} \cdot \omega_n$, where $s_{\mathbf{g}} \leq \sqrt{5}$ is a known constant. There are efficient algorithms that, given any desired syndrome $u \in \mathbb{Z}_q$, sample from a discrete Gaussian distribution over the coset $\Lambda_u^\perp(\mathbf{g}^t)$ for any given parameter $s \geq s_{\mathbf{g}} \cdot \omega_n$. Since $\Lambda^\perp(\mathbf{G}) \subset \mathbb{Z}^{nk}$ is the direct sum of n copies of $\Lambda^\perp(\mathbf{g}^t)$, discrete Gaussian sampling over a desired coset $\Lambda_{\mathbf{u}}^\perp(\mathbf{G})$ (with parameter $s \geq s_{\mathbf{g}} \cdot \omega_n$) can be accomplished by concatenating n independent samples over appropriate cosets of $\Lambda^\perp(\mathbf{g}^t)$.

Definition 3.1 ([MP12]). *Let $m \geq nk$ be an integer and define $\bar{m} = m - nk$. For $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we say that $\mathbf{R} \in \mathbb{Z}_q^{\bar{m} \times nk}$ is a* trapdoor *for $\mathbf{A}$ with* tag $\mathbf{H}^* \in \mathbb{Z}_q^{n \times n}$ *if $\mathbf{A} \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] = \mathbf{H}^* \cdot \mathbf{G}$. The* quality *of the trapdoor is defined to be the spectral norm $s_1(\mathbf{R})$.*

Note that $\mathbf{H}^*$ is uniquely determined and efficiently computable from $\mathbf{R}$, because $\mathbf{G}$ contains the n-by-n identity as a submatrix. Note also that if $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H}^*$, then it is also a trapdoor for $\mathbf{A}_\mathbf{H} := \mathbf{A} - [\mathbf{0} \mid \mathbf{HG}]$ with tag $\mathbf{H}^* - \mathbf{H} \in \mathbb{Z}_q^{n \times n}$.

The key-generation algorithm of [MP12] produces a parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{R}$ having desired tag $\mathbf{H}^*$. It does so by choosing (or being given) a uniformly random $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ and a random $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times nk}$ having small $s_1(\mathbf{R})$, and outputs $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}^* \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$. For sufficiently large $m \geq Cn \lg q$ (where C is a universal constant) and appropriate distribution of $\mathbf{R}$, the output matrix $\mathbf{A}$ is uniformly random, up to $\mathrm{negl}(n)$ statistical distance.

The discrete Gaussian sampling algorithm of [MP12] is an instance of the "convolution" approach from [Pei10]. It works in two phases:

1. In the *offline* "perturbation" phase, it takes as input a parity-check matrix $\mathbf{A}$, a trapdoor $\mathbf{R}$ for $\mathbf{A}$ with some tag $\mathbf{H}^* \in \mathbb{Z}_q^{n \times n}$, and a Gaussian parameter $s \geq Cs_1(\mathbf{R})$ (where C is some universal constant). It chooses one or more Gaussian perturbation vectors $\mathbf{p} \in \mathbb{Z}^m$ (one for each future call to the online sampling step) having non-spherical covariance $\Sigma_\mathbf{p}$ that depends only on s and the trapdoor $\mathbf{R}$.
2. In the *online* "syndrome correction" phase, it is given a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ and a tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. As long as $\mathbf{H}^* - \mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is invertible, it chooses $\mathbf{z} \in \mathbb{Z}^{nk}$ having Gaussian distribution with parameter $s_\mathbf{g} \cdot \omega_n$ over an appropriate coset of $\Lambda^\perp(\mathbf{G})$, and outputs $\mathbf{x} = \mathbf{p} + \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \mathbf{z} \in \Lambda_\mathbf{u}^\perp(\mathbf{A}_\mathbf{H})$, where $\mathbf{p}$ is a fresh perturbation from the offline step.

Informally, the perturbation covariance $\Sigma_\mathbf{p}$ of $\mathbf{p}$ is carefully designed to cancel out the trapdoor-revealing covariance of $\mathbf{y} = \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \mathbf{z}$, so that their sum has a (public) spherical Gaussian distribution. More formally, the output $\mathbf{x}$ has distribution within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda_\mathbf{u}^\perp(\mathbf{A}_\mathbf{H}), s \cdot \omega_n}$, and in particular does not reveal any information about the trapdoor $\mathbf{R}$ (aside from an upper bound s on $s_1(\mathbf{R})$, which is public).

We emphasize that for security, it is essential that none of the intermediate values $\mathbf{p}$, $\mathbf{z}$ or $\mathbf{y} = \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \mathbf{z}$ be revealed, otherwise they could be correlated with $\mathbf{x}$ to leak information about the trapdoor $\mathbf{R}$ that could lead to an attack like the one given in [NR06].

3.2 Functionalities for Threshold Sampling

Here we present ideal functionalities corresponding to the above two algorithms. The key-generation and Gaussian sampling functionalities $\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{GS}}$ are specified in Figure 1 and Figure 2, respectively; they internally execute the standalone algorithms described above.

To realize $\mathcal{F}_{\text{KG}}$, in the trusted setup model (as used in [CG99]) we can simply let the trusted party play the role of $\mathcal{F}_{\text{KG}}$, because key generation is a one-time setup. To realize $\mathcal{F}_{\text{GS}}$, for the purpose of modularity we define two lower-level functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ (Figures 3 and 4), which generate the perturbation and syndrome-correction components, respectively, as in the standalone sampling algorithm. The $\mathcal{F}_{\text{GS}}$, $\mathcal{F}_{\text{Perturb}}$, and $\mathcal{F}_{\text{Correct}}$ functionalities are all initialized with a bound B on the number of Gaussian samples that they will produce in their lifetimes. This is because the trusted setup (or offline precomputation) phases of our protocols need to prepare sufficient randomness so that the online phases can be noninteractive. (If the bound B is reached, then the parties can just initialize new copies of $\mathcal{F}_{\text{GS}}$, $\mathcal{F}_{\text{Perturb}}$, and $\mathcal{F}_{\text{Correct}}$ using the same arguments from $\mathcal{F}_{\text{KG}}$.)

Functionality $\mathcal{F}_{\text{KG}}$

Generate: Upon receiving $(\mathsf{gen}, sid, \bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}, \mathbf{H}^* \in \mathbb{Z}_q^{n \times n}, z)$ from at least h honest parties in $\mathcal{P}$:

- Choose $\mathbf{R} \leftarrow D_{\mathbb{Z}, z \cdot \omega_n}^{\bar{m} \times nk}$ and compute a sharing $[\![\mathbf{R}]\!]$ over $\mathbb{Z}_q$. Let $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}^* \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.
- Send $(\mathsf{gen}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i)$ to each party i in $\mathcal{P}$, and $(\mathsf{gen}, sid, \mathbf{A}, \mathbf{H}^*, z)$ to the adversary.

Figure 1: Key generation functionality

Functionality $\mathcal{F}_{\text{GS}}$

Initialize: Upon receiving $(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^*, s, B)$ from at least h honest parties i in $\mathcal{P}$:

- Reconstruct $\mathbf{R}$ and store sid, $\mathbf{A}$, $\mathbf{R}$, $\mathbf{H}^*$, s, and B.
- Send (init, sid) to each party in $\mathcal{P}$, and $(\mathsf{init}, sid, \mathbf{A}, \mathbf{H}^*, s, B)$ to the adversary.

Sample: Upon receiving $(\mathsf{sample}, sid, \mathbf{H} \in \mathbb{Z}_q^{n \times n}, \mathbf{u} \in \mathbb{Z}_q^n)$ from at least h honest parties in $\mathcal{P}$, if $\mathbf{H}^* - \mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is invertible and fewer than B calls to sample have already been made:

- Sample $\mathbf{x} \leftarrow D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}_{\mathbf{H}}), s \cdot \omega_n}$ using the algorithm from [MP12] with trapdoor $\mathbf{R}$.
- Send $(\mathsf{sample}, sid, \mathbf{x})$ to all parties in $\mathcal{P}$, and $(\mathsf{sample}, sid, \mathbf{H}, \mathbf{u}, \mathbf{x})$ to the adversary.

Figure 2: Gaussian sampling functionality

We next describe the helper functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$, and describe how they can be realized efficiently and noninteractively using trusted setup.

3.2.1 Perturbation

Our perturbation functionality $\mathcal{F}_{\text{Perturb}}$ (Figure 3) corresponds to the offline perturbation phase of the standalone sampling algorithm. The perturb command does not take any inputs, so its results can be precomputed offline, before the command is actually invoked. The possibility of precomputation introduces one subtlety in the definition of the functionality, however. Notice that the functionality asks the adversary for share values $[\![\mathbf{p}]\!]^i$ for the corrupted parties, then generates shares for the honest parties that are consistent with those shares; clearly this does not affect the secrecy of $\mathbf{p}$. This formulation is needed for proving security of a protocol that precomputes shares of perturbations before any real calls to perturb are made (we give such a protocol in Appendix A.4). It allows the simulator to choose shares on its own when simulating the precomputation, and ensures that the functionality later distributes shares that are consistent with the simulation. Observe that with trusted setup, $\mathcal{F}_{\text{Perturb}}$ can be trivially realized by just precomputing and distributing shares of B samples in the initialization phase, which the parties then consume in the online phase.

Note that $\mathcal{F}_{\text{Perturb}}$ distributes shares $[\![\mathbf{p}]\!]^i$ of a perturbation $\mathbf{p}$ to the players, which themselves do not reveal any information about $\mathbf{p}$ to the adversary, just as in the standalone Gaussian sampling algorithm. However, in order for the perturbation to be useful in the later syndrome-correction phase, the parties will need to know (and so $\mathcal{F}_{\text{Perturb}}$ reveals) some partial information about $\mathbf{p}$, namely, the syndromes $\bar{\mathbf{w}} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}] \cdot \mathbf{p} \in \mathbb{Z}_q^n$ and $\mathbf{w} = [\mathbf{0} \mid \mathbf{G}] \cdot \mathbf{p} \in \mathbb{Z}_q^n$. This is the main significant difference with the standalone setting, in which these same syndromes are calculated internally but never revealed. Informally, Lemma 3.2 below shows that the syndromes are uniformly random (up to negligible error), and hence can be simulated without knowing $\mathbf{p}$. Furthermore, $\mathbf{p}$ will still be a usable perturbation even after $\bar{\mathbf{w}}, \mathbf{w}$ are revealed, because it has an appropriate (non-spherical) Gaussian parameter which sufficiently exceeds the smoothing parameter of an appropriate lattice. This fact will be used later in the proof of security for our $\mathcal{F}_{\text{GS}}$ realization.

Functionality $\mathcal{F}_{\text{Perturb}}$

Initialize: Upon receiving $(\mathsf{init}, sid, \mathbf{A}_{-\mathbf{H}^*} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}], [\![\mathbf{R}]\!]^i, s, B)$ from at least h honest parties i in $\mathcal{P}$:

- Reconstruct $\mathbf{R}$ to compute covariance matrix $\Sigma_{\mathbf{p}} = s^2 - s_{\mathbf{g}}^2 \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \left[\begin{smallmatrix}\mathbf{R}^t & \mathbf{I}\end{smallmatrix}\right]$ and store sid, $\mathbf{A}_{-\mathbf{H}^*}$, and $\Sigma_{\mathbf{p}}$.
- Send (init, sid) to all parties in $\mathcal{P}$ and $(\mathsf{init}, sid, \mathbf{A}_{-\mathbf{H}^*}, s, B)$ to the adversary.

Perturb: Upon receiving $(\mathsf{perturb}, sid)$ from at least h honest parties in $\mathcal{P}$, if fewer than B calls to perturb have already been made:

- Choose $\mathbf{p} \leftarrow D_{\mathbb{Z}^m, \sqrt{\Sigma_{\mathbf{p}}} \cdot \omega_n}$.
- Compute $\bar{\mathbf{w}} = \mathbf{A}_{-\mathbf{H}^*} \cdot \mathbf{p} \in \mathbb{Z}_q^n$ and $\mathbf{w} = [\mathbf{0} \mid \mathbf{G}] \cdot \mathbf{p} \in \mathbb{Z}_q^n$.
- Send $(\mathsf{perturb}, sid, \bar{\mathbf{w}}, \mathbf{w})$ to the adversary, and receive back shares $[\![\mathbf{p}]\!]^i \in \mathbb{Z}_q^m$ for each currently corrupted party i in $\mathcal{P}$.
- Generate a uniformly random sharing $[\![\mathbf{p}]\!]$ consistent with the shares received in the previous step.
- Send $(\mathsf{perturb}, sid, [\![\mathbf{p}]\!]^i, \bar{\mathbf{w}}, \mathbf{w})$ to each party i in $\mathcal{P}$.

Figure 3: Perturbation functionality

Lemma 3.2. *Let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ be uniformly random for $\bar{m} = m - nk \geq n \lg q + \omega(\log n)$, let*

$$\mathbf{B} = \begin{bmatrix} \bar{\mathbf{A}} & -\bar{\mathbf{A}}\mathbf{R} \\ & \mathbf{G} \end{bmatrix} = (\bar{\mathbf{A}} \oplus \mathbf{G}) \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{2n \times (\bar{m}+nk)}$$

(where $\oplus$ denotes the direct sum), and let $\Lambda = \Lambda^\perp(\mathbf{B})$. Then with all but $\mathrm{negl}(n)$ probability over the choice of $\bar{\mathbf{A}}$, we have $\eta_\epsilon(\Lambda^\perp(\mathbf{B})) \leq \sqrt{5}(s_1(\mathbf{R}) + 1) \cdot \omega_n$ for some $\epsilon = \mathrm{negl}(n)$.

In particular, for $\mathbf{p} \leftarrow D_{\mathbb{Z}^m, \sqrt{\Sigma_{\mathbf{p}}}}$ where $\sqrt{\Sigma_{\mathbf{p}}} \geq 6(s_1(\mathbf{R}) + 1) \cdot \omega_n \geq 2\eta_\epsilon(\Lambda^\perp(\mathbf{B}))$, the syndrome $\mathbf{u} = (\bar{\mathbf{w}}, \mathbf{w}) = \mathbf{B}\mathbf{p} \in \mathbb{Z}_q^{2n}$ is $\mathrm{negl}(n)$-far from uniform, and the conditional distribution of $\mathbf{p}$ given $\mathbf{u}$ is $D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{B}), \sqrt{\Sigma_{\mathbf{p}}}}$.

Proof. By Lemma 2.4, we have $\eta_{\epsilon'}(\Lambda^\perp(\bar{\mathbf{A}})) \leq 2 \cdot \omega_n$ (with overwhelming probability) for some $\epsilon' = \mathrm{negl}(n)$. Also as shown in [MP12], we have $\eta_{\epsilon'}(\Lambda^\perp(\mathbf{G})) \leq \sqrt{5} \cdot \omega_n$ (see Section 3.1). This implies that

$$\eta_\epsilon(\Lambda^\perp(\bar{\mathbf{A}} \oplus \mathbf{G})) \leq \sqrt{5} \cdot \omega_n$$

where $(1 + \epsilon) = (1 + \epsilon')^2$, and in particular $\epsilon = \mathrm{negl}(n)$.

Since $\mathbf{T} = \left[\begin{smallmatrix}\mathbf{I} & -\mathbf{R}\\ & \mathbf{I}\end{smallmatrix}\right]$ is unimodular with inverse $\mathbf{T}^{-1} = \left[\begin{smallmatrix}\mathbf{I} & \mathbf{R}\\ & \mathbf{I}\end{smallmatrix}\right]$, it is easy to verify that $\Lambda^\perp(\mathbf{B}) = \mathbf{T}^{-1} \cdot \Lambda^\perp(\bar{\mathbf{A}} \oplus \mathbf{G})$, and hence

$$\eta_\epsilon(\Lambda^\perp(\mathbf{B})) \leq s_1(\mathbf{T}^{-1}) \cdot \eta_\epsilon(\Lambda^\perp(\bar{\mathbf{A}} \oplus \mathbf{G})) \leq \sqrt{5}(s_1(\mathbf{R}) + 1) \cdot \omega_n. \qquad \square$$

3.2.2 Syndrome Correction

Our functionality $\mathcal{F}_{\text{Correct}}$ (Figure 4) corresponds to the syndrome-correction step of the standalone sampling algorithm. Because its output $\mathbf{y}$ must lie in a certain coset $\Lambda_{\mathbf{v}}^\perp(\mathbf{A})$, where $\mathbf{v}$ depends on the desired final syndrome $\mathbf{u}$, the functionality must be invoked online. As indicated in the overview, the standalone algorithm samples $\mathbf{z} \leftarrow \Lambda_{\mathbf{v}}^\perp(\mathbf{G})$ and defines $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \mathbf{z}$. The functionality does the same, but outputs only shares of $\mathbf{y}$ to their respective owners. This ensures that no information about $\mathbf{y}$ is revealed to the adversary. (Note that the input syndrome $\mathbf{v}$ itself is not revealed in the standalone algorithm, but in our setting $\mathbf{v}$ is determined

Functionality $\mathcal{F}_{\text{Correct}}$

Initialize: Upon receiving $(\mathsf{init}, sid, [\![\mathbf{R}]\!]^i, B)$ from at least h honest parties i in $\mathcal{P}$:

- Reconstruct $\mathbf{R}$ and store sid, $\mathbf{R}$, and B.
- Send (init, sid) to all parties in $\mathcal{P}$ and (init, sid, B) to the adversary.

Correct: Upon receiving $(\mathsf{correct}, sid, \mathbf{v})$ from at least h honest parties in $\mathcal{P}$, if fewer than B calls to correct have already been made:

- Sample $\mathbf{z} \leftarrow D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}), s_{\mathbf{g}} \cdot \omega_n}$ and compute $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \mathbf{z}$.
- Send $(\mathsf{correct}, sid, \mathbf{v})$ to the adversary, receive shares $[\![\mathbf{y}]\!]^i \in \mathbb{Z}_q^m$ for each currently corrupted party i, and generate a uniformly random sharing $[\![\mathbf{y}]\!]$ consistent with these shares.
- Send $(\mathsf{correct}, sid, [\![\mathbf{y}]\!]^i)$ to each party i in $\mathcal{P}$.

Figure 4: Syndrome correction functionality

entirely by public information, so it is known to the adversary.) Also, just like $\mathcal{F}_{\text{Perturb}}$, the functionality asks the simulator for shares for the corrupted parties, to make precomputation simulatable.

Realizing $\mathcal{F}_{\text{Correct}}$ with a noninteractive protocol relies crucially on the *parallel* and *offline* nature of the corresponding step in the standalone algorithm. In particular, we use the fact that without knowing $\mathbf{v}$ in advance, the algorithm can precompute *partial* samples for each of the $q = \text{poly}(n)$ scalar values $v \in \mathbb{Z}_q$, and then linearly combine n such partial samples to answer a query for a full syndrome $\mathbf{v} \in \mathbb{Z}_q^n$.

In the trusted setup model, the protocol realizing $\mathcal{F}_{\text{Correct}}$ is as follows.

1. In the offline phase, a trusted party uses the trapdoor $\mathbf{R}$ (with tag $\mathbf{H}^*$) to distribute shares as follows. For each $j \in [n]$ and $v \in \mathbb{Z}_q$, the party initializes queues $Q^i_{j,v}$ for each party i, does the following B times, and then gives each of the resulting queues $Q^i_{j,v}$ to party i.

   - Sample $\mathbf{z}_{j,v} \leftarrow D_{\Lambda^{\perp}_{v}(\mathbf{g}^t), s_{\mathbf{g}} \cdot \omega_n}$.
   - Compute $\mathbf{y}_{j,v} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] (\mathbf{e}_j \otimes \mathbf{z}_{j,v})$, where $\mathbf{e}_j \in \mathbb{Z}^n$ denotes the jth standard basis vector. Note that
   $$\mathbf{A}_{\mathbf{H}} \cdot \mathbf{y}_{j,v} = (\mathbf{H}^* - \mathbf{H})\mathbf{G} \cdot (\mathbf{e}_j \otimes \mathbf{z}_{j,v}) = (\mathbf{H}^* - \mathbf{H})(v \cdot \mathbf{e}_j),$$
   where as always, $\mathbf{A}_{\mathbf{H}} = \mathbf{A} - [\mathbf{0} \mid \mathbf{HG}]$ for any $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$.
   - Generate a sharing for $\mathbf{y}_{j,v}$, and add $[\![\mathbf{y}_{j,v}]\!]^i$ to queue $Q^i_{j,v}$ for each party $i \in \mathcal{P}$.

2. In the online phase, upon receiving $(\mathsf{correct}, sid, \mathbf{v})$, each party i dequeues an entry $[\![\mathbf{y}_{j,v_j}]\!]^i$ from Q_{j,v_j} for each $j \in [n]$, and locally outputs $[\![\mathbf{y}]\!]^i = \sum_{j \in [n]} [\![\mathbf{y}_{j,v_j}]\!]^i$. Note that by linearity and the homomorphic properties of secret sharing, the shares $[\![\mathbf{y}]\!]^i$ recombine to $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \mathbf{z} \in \mathbb{Z}^m$ for some Gaussian-distributed $\mathbf{z}$ of parameter $s_{\mathbf{g}} \cdot \omega_n$, such that $\mathbf{A}_{\mathbf{H}} \cdot \mathbf{y} = (\mathbf{H}^* - \mathbf{H}) \cdot \mathbf{v} \in \mathbb{Z}_q^n$.

Without trusted setup, we give in Appendix A.3 an efficient protocol for $\mathcal{F}_{\text{Correct}}$ that operates in a similar way, populating the local queues $Q^i_{j,v}$ during the offline phase in a distributed manner using standard share-blinding and multiplication functionalities, among others.

3.2.3 Legal Uses of the Functionalities

Putting the key-generation and Gaussian sampling operations into separate functionalities $\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{GS}}$ (and $\mathcal{F}_{\text{DelTrap}}$ for delegation, in Section 3.4 below), and realizing $\mathcal{F}_{\text{GS}}$ using the helper functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$, aids modularity and simplifies the analysis of our protocols. However, as a side effect it also raises a

technical issue in the UC framework, since environments can in general provide functionalities with arbitrary inputs, even on behalf of honest users. The issue is that the functionalities are all designed to be initialized with some *common, valid* state—namely, shares of a trapdoor $\mathbf{R}$ for a matrix $\mathbf{A}$ as produced by $\mathcal{F}_{\text{KG}}$ on valid inputs—but it might be expensive or impossible for the corresponding protocols to check the consistency and validity those shares. Moreover, such checks would be unnecessary in the typical case where an application protocol, such as a threshold signature scheme, initializes the functionalities as intended.[3]

Therefore, we prove UC security for a restricted class of environments $\mathscr{Z}$ that always initialize our functionalities with valid arguments. In particular, environments in $\mathscr{Z}$ can instruct parties to instantiate $\mathcal{F}_{\text{KG}}$ only with appropriate arguments $\bar{\mathbf{A}}, z$. Similarly, $\mathcal{F}_{\text{GS}}$ (and $\mathcal{F}_{\text{DelTrap}}$) can be initialized only with a matrix $\mathbf{A}$, tag $\mathbf{H}^*$, and shares of a trapdoor $\mathbf{R}$ matching those of a prior call to the gen command of $\mathcal{F}_{\text{KG}}$, and with a sufficiently large Gaussian parameter $s \geq Cs_1 \cdot \omega_n$, where s_1 is a high-probability upper bound on $s_1(\mathbf{R})$ for the trapdoor $\mathbf{R}$ generated by $\mathcal{F}_{\text{KG}}$. The functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ (which in any case are not intended for direct use by applications) also must be initialized using a prior output of $\mathcal{F}_{\text{KG}}$.

More formally, an environment $\mathcal{Z}$ is said to be in the class $\mathscr{Z}$ if it satisfies the following conditions specific to our functionalities:

- When $\mathcal{Z}$ instructs honest parties to run the gen command of $\mathcal{F}_{\text{KG}}$, the input matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ and parameter z must correspond with a statistically secure instantiation of the trapdoor generator from [MP12]. Concretely, $\bar{\mathbf{A}}$ must be statistically close to uniformly random, and the Gaussian parameter z and dimension $\bar{m}$ must jointly be sufficiently large. As shown in [MP12], one valid instantiation is to let $z \geq 2\omega_n$ and $\bar{m} \geq Cn \lg q$ for any fixed constant $C > 1$.

- When $\mathcal{Z}$ instructs honest parties to run the init commands of $\mathcal{F}_{\text{GS}}$, $\mathcal{F}_{\text{Perturb}}$, or $\mathcal{F}_{\text{Correct}}$, the matrix $\mathbf{A}$, tag $\mathbf{H}^*$, and shares $[\![\mathbf{R}]\!]^i$ provided as the parties' inputs must match those of a prior call to $\mathcal{F}_{\text{KG}}$.gen. In addition, these init commands must all use the same Gaussian parameter s, which must be sufficiently large relative to the Gaussian parameter z and dimension $\bar{m}$ used in that call to $\mathcal{F}_{\text{KG}}$.gen. Specifically, we require $s \geq Cz(\sqrt{\bar{m}} + \sqrt{n \log q}) \cdot \omega_n$ for a certain universal constant C. By the results of [MP12], this guarantees that with overwhelming probability over the choice of $\mathbf{R}$, we have $s \geq C' s_1(\mathbf{R}) \cdot \omega_n$ for some universal constant C', which ensures that our π_{GS} protocol produces the proper distribution.

We emphasize that these restrictions on the environment are not actually limiting in any meaningful way, since our functionalities are only intended to serve as subroutines in higher-level applications, e.g. threshold signatures and (H)IBE. When designing a protocol ϕ that uses these functionalities (see, e.g., Section 4) one simply needs to ensure that ϕ does so in a manner consistent with the above conditions. Then composing ϕ with our protocols π_{KG}, π_{GS}, π_{Perturb}, and π_{Correct} (which we prove secure against environments in $\mathscr{Z}$) will yield a secure protocol against *any* t-limited environment.

3.3 Gaussian Sampling Protocol

Figure 5 defines a protocol π_{GS} that realizes the Gaussian sampling functionality $\mathcal{F}_{\text{GS}}$ in the $(\mathcal{F}_{\text{Perturb}}, \mathcal{F}_{\text{Correct}})$-hybrid model. Its sample command simply makes one call to each of the main commands of $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$, adjusting the requested syndrome as necessary to ensure that the syndrome of the final output is the desired one. (This is done exactly as in the standalone algorithm.) The shares of the perturbation $\mathbf{p}$ and syndrome-correction term $\mathbf{y}$ are then added locally and announced, allowing the players to reconstruct the final output $\mathbf{x} = \mathbf{p} + \mathbf{y}$. The security of π_{GS} is formalized in Theorem 3.3, and proved via the simulator $\mathcal{S}_{\text{GS}}$ in Figure 6.

[3] This issue is not limited to our setting, and can arise any time the key-generation and secret-key operations of a threshold scheme are put into separate functionalities. We note that using "joint state" [CR03] does not appear to resolve the issue, because it only allows multiple instances of the *same* protocol to securely share some joint state.

An essential point is that given the helper functionalities, the protocol π_{GS} is completely *noninteractive*, i.e., no messages are exchanged among the parties, except when broadcasting their shares of the final output. Similarly, recall that our realizations of $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ are also noninteractive, either when using trusted setup or offline precomputation (see Appendix A). In other words, in the fully realized sampling protocol, the parties can sample from any desired coset using only local computation, plus one broadcast of the final output shares. We emphasize that this kind of noninteractivity is nontrivial, because the number of possible cosets is exponentially large.

Protocol π_{GS} in the ($\mathcal{F}_{\text{Perturb}}$, $\mathcal{F}_{\text{Correct}}$)-hybrid model

Initialize: On input $(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^*, s, B)$, party i stores $\mathbf{H}^*$, calls $\mathcal{F}_{\text{Perturb}}(\mathsf{init}, sid, \mathbf{A}_{-\mathbf{H}^*}, [\![\mathbf{R}]\!]^i, s, B)$ and $\mathcal{F}_{\text{Correct}}(\mathsf{init}, sid, [\![\mathbf{R}]\!]^i, B)$, and outputs (init, sid).

Sample: On input $(\mathsf{sample}, sid, \mathbf{H}, \mathbf{u})$, if $\mathbf{H}^* - \mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is invertible, and if fewer than B calls to sample have already been made, then party i does:

- Call $\mathcal{F}_{\text{Perturb}}(\mathsf{perturb}, sid)$ and receive $(\mathsf{perturb}, sid, [\![\mathbf{p}]\!]^i, \bar{\mathbf{w}}, \mathbf{w})$.
- Compute $\mathbf{v} = (\mathbf{H}^* - \mathbf{H})^{-1}(\mathbf{u} - \bar{\mathbf{w}}) - \mathbf{w} \in \mathbb{Z}_q^n$.
- Call $\mathcal{F}_{\text{Correct}}(\mathsf{correct}, sid, \mathbf{v})$ and receive $(\mathsf{correct}, sid, [\![\mathbf{y}]\!]^i)$.
- Broadcast $[\![\mathbf{x}]\!]^i = [\![\mathbf{p}]\!]^i + [\![\mathbf{y}]\!]^i$ and reconstruct $\mathbf{x} = \mathbf{p} + \mathbf{y}$ from the announced shares.
- Output $(\mathsf{sample}, sid, \mathbf{x})$.

Figure 5: Gaussian sampling protocol

Simulator $\mathcal{S}_{\text{GS}}$

Initialize: Upon receiving $(\mathsf{init}, sid, \mathbf{A}, \mathbf{H}^*, s, B)$ from $\mathcal{F}_{\text{GS}}$, reveal to $\mathcal{Z}$ (init, sid) as outputs of both $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ to each currently corrupted party and any party that is corrupted in the future.

Sample: Upon receiving $(\mathsf{sample}, sid, \mathbf{H}, \mathbf{u}, \mathbf{x})$ from $\mathcal{F}_{\text{GS}}$:

- Choose uniform and independent $\bar{\mathbf{w}}, \mathbf{w} \in \mathbb{Z}_q^n$ and compute $\mathbf{v} = (\mathbf{H}^* - \mathbf{H})^{-1}(\mathbf{u} - \bar{\mathbf{w}}) - \mathbf{w} \in \mathbb{Z}_q^n$.
- On behalf of $\mathcal{F}_{\text{Perturb}}$, send $(\mathsf{perturb}, sid, \bar{\mathbf{w}}, \mathbf{w})$ to $\mathcal{Z}$ and receive back shares $[\![\mathbf{p}]\!]^i$ for each currently corrupted party i in $\mathcal{P}$. Generate a uniformly random sharing $[\![\mathbf{p}]\!]$ of $\mathbf{p} = \mathbf{0}$ consistent with these shares. Send $(\mathsf{perturb}, sid, [\![\mathbf{p}]\!]^i, \bar{\mathbf{w}}, \mathbf{w})$ to each corrupted party i in $\mathcal{P}$ on behalf of $\mathcal{F}_{\text{Perturb}}$.
- On behalf of $\mathcal{F}_{\text{Correct}}$, send $(\mathsf{correct}, sid, \mathbf{v})$ to $\mathcal{Z}$ and receive back shares $[\![\mathbf{y}]\!]^i$ for each currently corrupted party i in $\mathcal{P}$. Generate a uniformly random sharing $[\![\mathbf{y}]\!]$ of $\mathbf{y} = \mathbf{x}$ consistent with these shares. Send $(\mathsf{correct}, sid, [\![\mathbf{y}]\!]^i)$ to each corrupted party i in $\mathcal{P}$ on behalf of $\mathcal{F}_{\text{Correct}}$.
- Broadcast $[\![\mathbf{x}]\!]^i = [\![\mathbf{p}]\!]^i + [\![\mathbf{y}]\!]^i$ on behalf of each honest party i.

Corruption: When $\mathcal{Z}$ requests to corrupt party i, for each previous call to sample, reveal the corresponding messages $(\mathsf{perturb}, sid, [\![\mathbf{p}]\!]^i, \bar{\mathbf{w}}, \mathbf{w})$ and $(\mathsf{correct}, sid, [\![\mathbf{y}]\!]^i)$ to party i on behalf of $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$, respectively.

Figure 6: Simulator for π_{GS}

Theorem 3.3. *Protocol π_{GS} statistically realizes $\mathcal{F}_{GS}$ in the ($\mathcal{F}_{Perturb}$, $\mathcal{F}_{Correct}$)-hybrid model for t-limited environments in $\mathscr{Z}$.*

Proof sketch. The simulator $\mathcal{S}_{\text{GS}}$ in Figure 6 maintains consistent sharings of $\mathbf{p} = \mathbf{0}$ and $\mathbf{y} = \mathbf{x}$ for each call to sample, and it releases player i's shares of these values (on behalf of $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$) upon corruption

of player i. The fact that $\mathbf{p}$ and $\mathbf{y}$ in $\mathcal{S}_{\text{GS}}$ are from incorrect distributions is not detectable (even statistically) by the environment $\mathcal{Z}$, because it sees at most t shares of each, and the shares are consistent with announced shares of $\mathbf{x} = \mathbf{p} + \mathbf{y}$.

The only other significant issues relate to (1) the syndromes $\bar{\mathbf{w}}, \mathbf{w}$ output publicly by $\mathcal{F}_{\text{Perturb}}$ in the $(\mathcal{F}_{\text{Perturb}}, \mathcal{F}_{\text{Correct}})$-hybrid world, versus the simulator's choices of those values on behalf of $\mathcal{F}_{\text{Perturb}}$ in the ideal world; and (2) the distribution (conditioned on any fixed $\bar{\mathbf{w}}, \mathbf{w}$) of the final output $\mathbf{x}$ in both worlds. For item (1), as proved in Lemma 3.2, in the hybrid world the syndromes $\bar{\mathbf{w}}, \mathbf{w}$ are jointly uniform and independent (up to negligible statistical distance) over the choice of $\mathbf{p}$ by $\mathcal{F}_{\text{Perturb}}$, just as they are when produced by the simulator. Moreover, conditioned on any fixed values of $\bar{\mathbf{w}}, \mathbf{w}$, the distribution of $\mathbf{p}$ in the hybrid world is a discrete Gaussian with covariance $\Sigma_{\mathbf{p}}$ over a certain lattice coset $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{B})$, and the actual value of $\mathbf{p}$ from this distribution is perfectly hidden by the secret-sharing scheme.

For item (2), the above facts imply that in the hybrid world, $\mathbf{x} = \mathbf{p} + \mathbf{y}$ has spherical discrete Gaussian distribution $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A_H}),s}$, just as the output $\mathbf{x}$ of $\mathcal{F}_{\text{GS}}$ does in the ideal world (up to negligible statistical error in both cases). The proof is essentially identical to that of the "convolution lemma" from [MP12], which guarantees the correctness of the standalone sampling algorithm (as run by $\mathcal{F}_{\text{GS}}$ in the ideal world). The only slight difference is that in the hybrid world, $\mathbf{p}$'s distribution (conditioned on any fixed values of $\bar{\mathbf{w}}, \mathbf{w}$) is a discrete Gaussian with parameter $\sqrt{\Sigma_{\mathbf{p}}}$ over a coset of $\Lambda^{\perp}(\mathbf{B})$, instead of over $\mathbb{Z}^m$ as in the standalone algorithm. Fortunately, Lemma 3.2 says that $\sqrt{\Sigma_{\mathbf{p}}} \geq 2\eta_{\epsilon}(\Lambda^{\perp}(\mathbf{B}))$, and this is enough to adapt the proof from [MP12] to the different distribution of $\mathbf{p}$.

Finally, by the homomorphic properties of secret sharing, the shares $[\![\mathbf{p}]\!]^i + [\![\mathbf{y}]\!]^i$ announced by the honest parties are jointly distributed exactly as a fresh sharing of $\mathbf{x}$ as produced by the simulator. We conclude that the hybrid and real views are statistically indistinguishable, as desired. □

3.4 Trapdoor Delegation

The trapdoor delegation functionality $\mathcal{F}_{\text{DelTrap}}$ given in Figure 7 corresponds to the algorithm DelTrap for delegating a lattice trapdoor in [MP12], which is used in hierarchical IBE schemes. The functionality is initialized with shares of a trapdoor $\mathbf{R}$ for some $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For an extended matrix $\mathbf{A}' = [\mathbf{A_H} | \mathbf{A}_1] \in \mathbb{Z}_q^{n \times (m+nk)}$ (where $\mathbf{A_H} = \mathbf{A} - [\mathbf{0} \mid \mathbf{HG}]$) and tag $\mathbf{H}' \in \mathbb{Z}_q^{n \times n}$, $\mathcal{F}_{\text{DelTrap}}$ outputs shares of a trapdoor $\mathbf{R}'$ for $\mathbf{A}'$ with tag $\mathbf{H}'$, where the distribution of $\mathbf{R}'$ is Gaussian and in particular is independent of $\mathbf{R}$.

A realization of $\mathcal{F}_{\text{DelTrap}}$ in the $\mathcal{F}_{\text{GS}}$-hybrid model is given by π_{DelTrap} in Figure 8. It is entirely straightforward, since the standalone algorithm from [MP12] just draws several Gaussian samples over appropriate (publicly computable) cosets of $\Lambda^{\perp}(\mathbf{A})$, so we omit the proof of security.

4 Threshold Signatures and IBE

Here we apply our protocols in a straightforward manner to give threshold versions of the signature and identity-based encryption schemes from [GPV08]. Other signature and (H)IBE schemes that use key-generation and Gaussian sampling as "black boxes" can be similarly adapted to the threshold setting.

The GPV schemes. For security parameter n, modulus q and message space $\mathcal{M}$, the GPV signature scheme uses a hash function $H \colon \mathcal{M} \to \mathbb{Z}_q^n$, which is modeled as a random oracle, and two algorithms GenTrap and SampleD. At a high level, $\mathsf{GenTrap}(n, q, m)$ generates a nearly uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{R}$. Using these, $\mathsf{SampleD}(\mathbf{A}, \mathbf{R}, \mathbf{u}, s)$ generates a Gaussian sample (for any sufficiently large

Functionality $\mathcal{F}_{\text{DelTrap}}$

Initialize: Upon receiving $(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^*, s, B)$ from at least h honest parties in $\mathcal{P}$:

- Reconstruct trapdoor $\mathbf{R}$ and its invertible tag $\mathbf{H}$ for $\mathbf{A}$, and store sid, $\mathbf{A}$, $\mathbf{R}$, $\mathbf{H}^*$ and s.
- Send (init, sid) to each party in $\mathcal{P}$, and $(\mathsf{init}, sid, \mathbf{A}, s, B)$ to the adversary.

Delegate: Upon receiving $(\mathsf{delegate}, sid, \mathbf{H}, \mathbf{A}_1, \mathbf{H}')$ from at least h honest parties in $\mathcal{P}$:

- If $\mathbf{H}^* - \mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is invertible, using the Gaussian sampling algorithm from [MP12] with trapdoor $\mathbf{R}$, sample each column of $\mathbf{R}'$ independently from a discrete Gaussian with parameter s over the appropriate coset of $\Lambda^{\perp}(\mathbf{A}_{\mathbf{H}})$, so that $\mathbf{A}_{\mathbf{H}} \cdot \mathbf{R}' = \mathbf{H}' \cdot \mathbf{G} - \mathbf{A}_1$.
- Compute a sharing $[\![\mathbf{R}']\!]$ over $\mathbb{Z}_q$, and send $(\mathsf{delegate}, sid, [\![\mathbf{R}']\!]^i)$ to each party i, and $(\mathsf{delegate}, sid, \mathbf{H}, \mathbf{A}_1, \mathbf{H}')$ to the adversary.

Figure 7: Functionality for delegating a lattice trapdoor

Protocol π_{DelTrap} in the $\mathcal{F}_{\text{GS}}$-hybrid model

Initialize: On input $(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^*, s, B)$, call $\mathcal{F}_{\text{GS}}(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^*, s, Bnk)$.

Delegate: On input $(\mathsf{delegate}, sid, \mathbf{H}, \mathbf{A}_1, \mathbf{H}')$, party i does the following:

- For each $j = 1, \ldots, nk$, call $\mathcal{F}_{\text{GS}}(\mathsf{sample}, sid, \mathbf{H}, \mathbf{u})$ where $\mathbf{u}$ is the jth column of $\mathbf{H}'\mathbf{G} - \mathbf{A}_1$; receive $(\mathsf{sample}, sid, \mathbf{r}'_j)$ from $\mathcal{F}_{\text{GS}}$ and let $\mathbf{r}'_j$ be the jth column of $\mathbf{R}'$.
- Output $(\mathsf{sample}, sid, \mathbf{R}')$.

Figure 8: Protocol for delegating a lattice trapdoor

parameter s) over the lattice coset $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$. Ignoring the exact selection of parameters, the stateful version of the signature scheme consists of the following three algorithms:

- $\mathsf{KeyGen}(1^n)$: Let $(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{GenTrap}(n, q, m)$ and output verification key $vk = \mathbf{A}$ and signing key $sk = \mathbf{R}$.
- $\mathsf{Sign}(sk, \mu \in \mathcal{M})$: If (μ, σ) is already in local storage, output the signature σ. Otherwise, let $\mathbf{x} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{R}, H(\mu), s)$ and store (μ, σ). Output the signature $\sigma = \mathbf{x}$.
- $\mathsf{Verify}(vk, \mu, \sigma = \mathbf{x})$: If $\mathbf{A}\mathbf{x} = H(\mathbf{m})$ and $\mathbf{x}$ is sufficiently short, then accept; otherwise, reject.

See [GPV08] for the proof of (strong) unforgeability under worst-case lattice assumptions.

In the GPV identity-based encryption scheme, the setup algorithm is the same as KeyGen above, and the master public and secret keys are simply the verification and signing keys above. The secret key for an individual identity is a signature on that identity. Since we are concerned only with thresholdizing the signing and key-extraction algorithms, the details of the encryption and decryption algorithms are unchanged and irrelevant here, so we need only give threshold version of KeyGen and Sign.

Thresholdizing. In order to obtain a threshold signature scheme, KeyGen and Sign must be done in a distributed way, so that the signing key $sk = \mathbf{R}$ is distributed among the participating parties and a valid signature σ can only be produced by a quorum of participating parties. In Figure 9 we recall (from [ADN06]) a formal functionality for threshold signatures. (Recall that $\mathcal{P}$ is the set of trustees, or parties authorized to receive shares of the signing key.)

Functionality $\mathcal{F}_{\text{TSig}}$

Generate: Upon receiving (gen, sid, B) from at least h honest parties in $\mathcal{P}$, send (gen, sid, B) to the adversary, receive and record verification key v, and send (gen, sid, v) to each party in $\mathcal{P}$.

Sign: Upon receiving (sign, sid, m) from at least h honest parties in $\mathcal{P}$, if fewer than B calls to sign have already been made:

- Send (sign, sid, m) to the adversary and receive signature σ.
- If there is no record of $(m, \sigma, v, 0)$, record $(m, \sigma, v, 1)$ and send $(\mathsf{sign}, sid, m, \sigma)$ to each party in $\mathcal{P}$.

Verify: Upon receiving $(\mathsf{ver}, sid, m, \sigma, v')$ from any party $p \in \mathcal{P}$:

- Send $(\mathsf{ver}, sid, m, \sigma, v')$ to the adversary and receive $(\mathsf{ver}, sid, m, \phi)$.
- If $v' = v$ and $(m, \sigma, v, 1)$ is recorded, then send $(\mathsf{ver}, sid, m, 1)$ to p.
- If $v' = v$ and there is no recorded $(m, \sigma', v, 1)$, then record $(m, \sigma, v, 0)$ and send $(\mathsf{ver}, sid, m, 0)$ to p.
- If some $(m, \sigma, v', 1)$ is recorded, then send $(\mathsf{ver}, sid, m, 1)$ to p.
- If some $(m, \sigma, v', 0)$ is recorded, then send $(\mathsf{ver}, sid, m, 0)$ to p.
- Otherwise, record (m, σ, v', ϕ) and send $(\mathsf{ver}, sid, m, \sigma, \phi)$ to p.

Figure 9: Threshold signature functionality

To construct a protocol for threshold GPV signatures we need threshold analogues of GenTrap and SampleD; these are the functionalities $\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{GS}}$ (from Section 3), respectively. $\mathcal{F}_{\text{KG}}$ produces $\mathbf{A}$ as usual, but each party i receives a share $[\![\mathbf{R}]\!]^i$ of the trapdoor. To produce a signature, each party i in a quorum of signers simply calls $\mathcal{F}_{\text{GS}}$.sample with his share $[\![\mathbf{R}]\!]^i$, and this allows them to collectively produce a signature σ.

In Figure 10 we present a protocol for threshold GPV signatures in the $(\mathcal{F}_{\text{KG}}, \mathcal{F}_{\text{GS}})$-hybrid model. Note that it obeys all the constraints on the usage of $\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{GS}}$ described in Section 3.2.3. Its security is easily proved using the correspondence between $\mathcal{F}_{\text{KG}}$ and KeyGen, and $\mathcal{F}_{\text{GS}}$ and Sign, so we state Theorem 4.1 without proof.

Theorem 4.1. *The protocol $\pi_{ThreshGPV}$ securely realizes $\mathcal{F}_{TSig}$, assuming the unforgeability of the GPV signature scheme (with the same parameters) under chosen-message attacks.*

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ADN06] J. F. Almansa, I. Damgård, and J. B. Nielsen. Simplified threshold RSA with adaptive and proactive security. In *EUROCRYPT*, pages 593–611. 2006.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*. 2011. To appear.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9. 1999.

Protocol $\pi_{\text{ThreshGPV}}$ in the $(\mathcal{F}_{\text{KG}}, \mathcal{F}_{\text{GS}})$-hybrid model

Generate: On input (gen, sid, B):

- The parties run an information-theoretically secure coin-flipping protocol to choose a uniformly random $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$, and let $\mathbf{H}^* = \mathbf{I}_n$ (the n-by-n identity matrix).
- Each party i calls $\mathcal{F}_{\text{KG}}(\mathsf{gen}, sid, \bar{\mathbf{A}}, \mathbf{H}^*, z = 2 \cdot \omega_n)$ and receives $(\mathsf{gen}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i)$.
- Each party i calls $\mathcal{F}_{\text{GS}}(\mathsf{init}, sid, \mathbf{A}, [\![\mathbf{R}]\!]^i, \mathbf{H}^* = \mathbf{I}_n, s = Cz \cdot O(\sqrt{n \log q}) \cdot \omega_n, B)$ and outputs $(\mathsf{gen}, sid, \mathbf{A})$.

Sign: On input (sign, sid, m), if fewer than B calls to sign have already been made, then party i does:

- First check if (sid, m, σ) is already in local storage. If so, output $(\mathsf{sign}, sid, m, \sigma)$.
- Otherwise, compute $\mathbf{u} = H(m)$, call $\mathcal{F}_{\text{GS}}(\mathsf{sample}, sid, \mathbf{H} = \mathbf{0}, \mathbf{u})$, and receive $(\mathsf{sample}, sid, \bar{\mathbf{x}} \in \mathbb{Z}_q^m)$. Interpret $\bar{\mathbf{x}}$ as the unique integer vector $\mathbf{x} \in \mathbb{Z}^m$ with entries in $[-q/2, q/2)$, store $(sid, m, \sigma = \mathbf{x})$ locally, and output $(\mathsf{sign}, sid, m, \sigma)$.

Verify: Upon receiving $(\mathsf{ver}, sid, m, \sigma = \mathbf{x}, v')$:

- Each party i checks that $v' = \mathbf{A}$, that $\mathbf{A}\mathbf{x} = H(m)$, and that $\|\mathbf{x}\| \leq s\sqrt{m}$. If so, then party i outputs $(\mathsf{ver}, sid, m, 1)$, otherwise it outputs $(\mathsf{ver}, sid, m, 0)$.

Figure 10: Threshold GPV Signatures

[ALR11] G. Asharov, Y. Lindell, and T. Rabin. Perfectly-secure multiplication for any $< n/3$. In *CRYPTO*, pages 240–258. 2011.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, April 2011. Preliminary version in STACS 2009.

[BCHK07] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BD10] R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, pages 201–218. 2010.

[BF11a] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, pages 149–168. 2011.

[BF11b] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography*, pages 1–16. 2011.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. 1988.

[Can00] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. 2001.

[CG99] R. Canetti and S. Goldwasser. An efficient *threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, pages 90–106. 1999.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010.

[CLRS10] P.-L. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. In *Proceedings of the First international conference on Progress in cryptology: cryptology and information security in Latin America*, LATINCRYPT'10, pages 255–272. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-14711-9, 978-3-642-14711-1.

[CR03] R. Canetti and T. Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281. 2003.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25. 1998.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315. 1989.

[DF94] Y. Desmedt and Y. Frankel. Perfect homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM J. Discrete Math.*, 7(4):667–679, 1994.

[Feh98] S. Fehr. *Span Programs over Rings and How to Share a Secret from a Module*. Master's thesis, ETH Zurich, Institute for Theoretical Computer Science, 1998.

[FGM10] T. Feng, Y. Gao, and J. Ma. Changeable threshold signature scheme based on lattice theory. *International Conference on E-Business and E-Government*, 0:1311–1315, 2010. doi:http://doi.ieeecomputersociety.org/10.1109/ICEE.2010.335.

[GKV10] S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT*, pages 395–412. 2010.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[Kle00] P. N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. 2000.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*. 2012.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[MSs11] S. Myers, M. Sergi, and a. shelat. Threshold fully homomorphic encryption and secure computation. Cryptology ePrint Archive, Report 2011/454, 2011. `http://eprint.iacr.org/`.

[NR06] P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009. Preliminary version in Eurocrypt 2006.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196. 2008.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85. 1989.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[Sha79] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sho00] V. Shoup. Practical threshold signatures. In *EUROCRYPT*, pages 207–220. 2000.

[XXZ11] X. Xie, R. Xue, and R. Zhang. Efficient threshold encryption from lossy trapdoor functions. In *PQCrypto*, pages 163–178. 2011.

A Protocols Without Trusted Setup

Here we show how to realize threshold key generation $\mathcal{F}_{\text{KG}}$ and discrete Gaussian sampling $\mathcal{F}_{\text{GS}}$ without relying on any trusted setup. Given the protocol in Section 3.3, for $\mathcal{F}_{\text{GS}}$ it suffices to realize $\mathcal{F}_{\text{Correct}}$ and $\mathcal{F}_{\text{Perturb}}$. The rest of the section is organized as follows:

- In Section A.1 we formalize three low-level utility functionalities $\mathcal{F}_{\text{Blind}}$, $\mathcal{F}_{\text{Mult}}$, and $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ used by several of our protocols, and we describe how these functionalities are realized.
- In Section A.2 we give a protocol realizing $\mathcal{F}_{\text{KG}}$ using the utility functionalities. This simple protocol and security analysis are representative of the techniques we use (in more complex ways) in later protocols as well.
- In Section A.3 we give a realization of $\mathcal{F}_{\text{Correct}}$ that uses an additional utility functionality $\mathcal{F}_{\text{Gadget}}$, which we define and realize there.
- Finally, in Section A.4 we realize $\mathcal{F}_{\text{Perturb}}$ using a simple extension of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$, which we also realize there.

A.1 Utility Functionalities

We first present the low-level utility functionalities.

Blinding. The blinding functionality $\mathcal{F}_{\text{Blind}}$ (Figure 11) simply accepts shares of some value over an arbitrary additive group G, and distributes fresh shares of the same value. Our later protocols will use blinding and the homomorphic properties of secret sharing to reveal the values of shared secrets modulo lattices, and nothing more.

Realizations of $\mathcal{F}_{\text{Blind}}$ in various communication models are standard. For example, to realize it against semi-honest corruptions with private channels is very simple: simply add sufficiently many player-generated sharings of 0 to the original shares. For malicious corruptions one can use, e.g., subprotocols of the BGW [BGW88, ALR11] or RB [RB89] protocols, which run in a constant number of rounds. We leave the

Functionality $\mathcal{F}_{\text{Blind}}$

Blind: Upon receiving $(\mathsf{blind}, sid, [\![x]\!]^i \in G)$ from at least h honest parties in $\mathcal{P}$:

- Reconstruct x and generate a fresh sharing $[\![y]\!]$ (over G) of $y = x$.
- Send $(\mathsf{blind}, sid, [\![y]\!]^i)$ to each party i in $\mathcal{P}$, and (blind, sid, G) to the adversary.

Figure 11: Blinding functionality

implementation of $\mathcal{F}_{\text{Blind}}$ unspecified and simply work in the $\mathcal{F}_{\text{Blind}}$-hybrid model where needed. We remark that our protocols use $\mathcal{F}_{\text{Blind}}$ only during initialization, so the interaction required to implement it is limited to the offline phase.

Multiplication. The multiplication functionality $\mathcal{F}_{\text{Mult}}$ (Figure 12) takes shares of two values x, y in a ring $\mathbb{Z}_{q^d}$ and returns fresh shares of their product $x \cdot y$ (modulo q^d) to the respective parties. By the homomorphic properties of secret sharing, this generalizes immediately (via local computation alone) to products of vectors and/or matrices $\mathbf{X}, \mathbf{Y}$, so we write the functionality to support this more general capability. To realize $\mathcal{F}_{\text{Mult}}$ one can use any statistically secure protocol, such as the constant-round protocols of [BGW88, RB89, ALR11]; we leave this choice unspecified and simply work in the $\mathcal{F}_{\text{Mult}}$-hybrid model where needed.

Functionality $\mathcal{F}_{\text{Mult}}$

Multiply: Upon receiving $(\mathsf{mult}, sid, [\![\mathbf{X}]\!]^i \in \mathbb{Z}_{q^d}^{h \times \ell}, [\![\mathbf{Y}]\!]^i \in \mathbb{Z}_{q^d}^{\ell \times w})$ from at least h honest parties i in $\mathcal{P}$:

- Reconstruct $\mathbf{X}$, $\mathbf{Y}$ from the shares $[\![\mathbf{X}]\!]^i$, $[\![\mathbf{Y}]\!]^i$, respectively.
- Generate a fresh sharing $[\![\mathbf{Z}]\!]$ of $\mathbf{Z} = \mathbf{X} \cdot \mathbf{Y} \in \mathbb{Z}_{q^d}^{h \times w}$.
- Send $(\mathsf{mult}, sid, [\![\mathbf{Z}]\!]^i)$ to each party i in $\mathcal{P}$, and $(\mathsf{mult}, sid, h \times \ell, \ell \times w, d)$ to the adversary.

Figure 12: Multiplication functionality

Sampling integers. Several of our protocols rely on a low-level functionality $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ (Figure 13) for sampling discrete Gaussians over the integers $\mathbb{Z}$. At a high level, the sample command produces shares of a discrete Gaussian variable $x \in \mathbb{Z}$ with a given parameter, where the sharing is over the additive group $\mathbb{Z}_{q^d}$, (i.e., with d digits of precision), and distributes these shares $[\![x]\!]^i$ to the respective parties. Later on in Section A.4.1 we will extend $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ with some additional commands, but for now we only need the sample command.

Functionality $\mathcal{F}_{\text{Samp}\mathbb{Z}}$

Sample: Upon receiving $(\mathsf{sample}, sid, h \times w, z, d)$ from at least h honest parties in $\mathcal{P}$:

- Sample $\mathbf{X} \leftarrow D_{\mathbb{Z}, z \cdot \omega_n}^{h \times w}$ and generate a fresh sharing $[\![\mathbf{X}]\!]$ over $\mathbb{Z}_{q^d}$.
- Send $(\mathsf{sample}, sid, [\![\mathbf{X}]\!]^i)$ to each party i in $\mathcal{P}$ and $(\mathsf{sample}, sid, h \times w, z, d)$ to the adversary.

Figure 13: Simplified integer sampling functionality

Because we do not know of any highly efficient algorithms for sampling discrete Gaussians, our realization uses the general "inverse transform" sampling algorithm, and implements it securely using multiparty

computation tools. Recall that inverse sampling involves a close approximation of the cumulative distribution function. To sample, one chooses a uniformly random $u \in [0, 1)$ and looks up the corresponding output value in the table. An arithmetic circuit implementing this algorithm can be written as an AND of several interval tests on the input u, so the depth of the circuit is roughly the precision (number of digits) of the entries in the lookup table, and the width of the circuit is roughly the number of entries in the table. (Other trade-offs between depth and width are possible as well.)

Importantly, we can implement the inverse sampling method for discrete Gaussians using a table of size proportional to $q = \mathrm{poly}(n)$ for very large parameters z, even though the distribution has support size proportional to z. This is because for $z \in [q^j, q^{j+1})$, the discrete Gaussian of parameter z can be decomposed using Lemma 2.5 as a convolution of j discrete Gaussians over $q^j\mathbb{Z}, q^{j-1}\mathbb{Z}, \ldots, \mathbb{Z}$ having respective parameters roughly $z, z/q, \ldots, z/q^j$. (Note that each parameter can be chosen to be larger than the smoothing parameter of the respective lattice.) Each of these distributions is highly concentrated on only $\mathrm{poly}(n)$ outputs.

Finally, we emphasize that our higher-level protocols use $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ only in their key-generation or initialization phases, and only with fixed, public Gaussian parameters, so any inefficiencies in a realization of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ are limited to the offline phase.

A.2 Realizing $\mathcal{F}_{\text{KG}}$

The protocol π_{KG} (Figure 14) realizing $\mathcal{F}_{\text{KG}}$ in the $(\mathcal{F}_{\text{Blind}}, \mathcal{F}_{\text{Samp}\mathbb{Z}})$-hybrid model is straightforward, given the homomorphic properties of the secret-sharing scheme and the simple operation of the standalone trapdoor generator, which just multiplies a public uniform matrix $\bar{\mathbf{A}}$ with a secret Gaussian-distributed matrix $\mathbf{R}$. The parties first get shares of a Gaussian-distributed trapdoor $\mathbf{R}$ using $\mathcal{F}_{\text{Samp}\mathbb{Z}}$, then announce *blinded* shares of $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R} \bmod q$ and reconstruct $\mathbf{A}_1$ to determine the public key $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1]$. The blinding is needed so that the announced shares reveal only $\mathbf{A}_1$, and nothing more about the honest parties' shares $[\![\mathbf{R}]\!]^i$ themselves.

Protocol π_{KG} in the $(\mathcal{F}_{\text{Blind}}, \mathcal{F}_{\text{Samp}\mathbb{Z}})$-hybrid model

Generate: On input $(\mathsf{gen}, sid, \bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}, \mathbf{H}^* \in \mathbb{Z}_q^{n \times n}, z)$, party i does:

- Call $\mathcal{F}_{\text{Samp}\mathbb{Z}}(\mathsf{sample}, sid, \bar{m} \times nk, z, 1)$ and receive $(\mathsf{sample}, sid, [\![\mathbf{R}]\!]^i)$.
- Call $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, -\bar{\mathbf{A}}[\![\mathbf{R}]\!]^i)$ and receive $(\mathsf{blind}, sid, [\![\mathbf{A}_1]\!]^i)$.
- Broadcast $[\![\mathbf{A}_1]\!]^i$ and reconstruct $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R}$ from the announced shares.
- Output $(\mathsf{gen}, sid, \mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}^* \cdot \mathbf{G} + \mathbf{A}_1], [\![\mathbf{R}]\!]^i)$.

Figure 14: Key generation protocol

A simulator $\mathcal{S}_{\text{KG}}$ for demonstrating the security of π_{KG} is provided in Figure 15. Essentially, security boils down to the fact that the announced blinded shares $-\bar{\mathbf{A}}[\![\mathbf{R}]\!]^i$ in the protocol π_{KG} form a uniformly random, and independent of the honest parties' outputs $[\![\mathbf{R}]\!]^i$, sharing of $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R}$, which is exactly what $\mathcal{S}_{\text{KG}}$ constructs to simulate the broadcast messages. A full proof is a straightforward application of this observation, and of the privacy, robustness, and homomorphic properties of the secret-sharing scheme, so we omit it.

Theorem A.1. *Protocol π_{KG} statistically realizes $\mathcal{F}_{KG}$ in the $(\mathcal{F}_{Blind}, \mathcal{F}_{Samp\mathbb{Z}})$-hybrid model for t-limited environments in $\mathscr{Z}$.*

Simulator $\mathcal{S}_{\text{KG}}$

Generate: Upon receiving $(\mathsf{gen}, sid, \mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}^* \cdot \mathbf{G} + \mathbf{A}_1], \mathbf{H}^*, z)$ from $\mathcal{F}_{\text{KG}}$:

- Generate a fresh sharing of $\mathbf{A}_1$ over $\mathbb{Z}_q$.
- For all currently corrupted parties i, and whenever $\mathcal{Z}$ later requests to corrupt a party i, receive the share $[\![\mathbf{R}]\!]^i$ from $\mathcal{F}_{\text{KG}}$, and reveal to $\mathcal{Z}$ the following functionality outputs to party i:
  - $(\mathsf{sample}, sid, [\![\mathbf{R}]\!]^i)$ on behalf of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$;
  - $(\mathsf{blind}, sid, [\![\mathbf{A}_1]\!]^i)$ on behalf of $\mathcal{F}_{\text{Blind}}$.
- Broadcast $[\![\mathbf{A}_1]\!]^i$ on behalf of each honest party i.

Figure 15: Simulator for π_{KG}

A.3 Realizing $\mathcal{F}_{\text{Correct}}$

Recall that $\mathcal{F}_{\text{Correct}}$ samples and distributes shares of a (non-spherical Gaussian) vector $\mathbf{y}$ from a desired coset of $\Lambda^\perp(\mathbf{A})$. Section 3.2.2 describes how to realize $\mathcal{F}_{\text{Correct}}$ with trusted setup, partly by precomputing shares of samples from each coset of $\Lambda^\perp(\mathbf{g}^t)$. We next describe how these shares can be obtained from a utility functionality called $\mathcal{F}_{\text{Gadget}}$ (Figure 16), and how it can easily be realized.

A.3.1 Gadget Functionality

The functionality $\mathcal{F}_{\text{Gadget}}$ (Figure 16) relates to the special gadget vector $\mathbf{g}$ and lattice $\Lambda^\perp(\mathbf{g}^t)$, as defined in [MP12] and reviewed in Section 3.1. Recall that we have a fixed public vector $\mathbf{g}^t = [1, 2, 4, \ldots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$ for $k = \lceil \lg q \rceil$, which defines a full-rank lattice $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$ of determinant q whose smoothing parameter is bounded by $s_{\mathbf{g}} \cdot \omega_n$, where $s_{\mathbf{g}} \leq \sqrt{5}$ is a known constant. The functionality generates shares of a discrete Gaussian over the coset $\Lambda_v^\perp(\mathbf{g}^t)$ for any desired $v \in \mathbb{Z}_q$, by running any of the efficient algorithms described in [MP12].

Functionality $\mathcal{F}_{\text{Gadget}}$

Sample coset: Upon receiving $(\mathsf{cosetsample}, sid, v \in \mathbb{Z}_q)$ from at least h honest parties in $\mathcal{P}$:

- Sample $\mathbf{z} \leftarrow D_{\Lambda_v^\perp(\mathbf{g}^t), s_{\mathbf{g}}\cdot\omega_n}$ and generate a uniformly random sharing $[\![\mathbf{z}]\!]$.
- Send $(\mathsf{cosetsample}, sid, [\![\mathbf{z}]\!]^i)$ to each party i in $\mathcal{P}$, and send $(\mathsf{cosetsample}, sid, v)$ to the adversary.

Figure 16: Functionality for operations related to the gadget lattice $\Lambda^\perp(\mathbf{g}^t)$

Realizing $\mathcal{F}_{\text{Gadget}}$ is straightforward in the $(\mathcal{F}_{\text{Samp}\mathbb{Z}}, \mathcal{F}_{\text{Blind}})$-hybrid model, using the homomorphic properties of secret sharing: essentially, the parties request shares of a Gaussian-distributed $\mathbf{z} \in \mathbb{Z}^k$ from $\mathcal{F}_{\text{Samp}\mathbb{Z}}$, then broadcast blinded shares of the syndrome $u = \langle \mathbf{g}, \mathbf{z} \rangle \bmod q$ and recover u, repeating until $u = v$. (The blinding is needed so that nothing more than the syndrome is revealed about $\mathbf{z}$.) Implemented naively as in Figure 17, the expected number of trials (which may be performed in parallel) is almost exactly $q = \text{poly}(n)$, because the syndrome u is negligibly far from uniform since $\mathbf{z}$'s Gaussian parameter is at least the smoothing parameter of $\Lambda^\perp(\mathbf{g}^t)$. Alternatively, shares of samples having the wrong syndrome can be stored away and used as needed later on. Note that in any case, $\mathcal{F}_{\text{Gadget}}$ is only ever called in the offline phase of π_{Correct} (Figure 19), so efficiency is not a top priority here.

A simulator $\mathcal{S}_{\text{Gadget}}$ for demonstrating the security of our protocol is provided in Figure 18. For a t-limited

Protocol π_{Gadget} in the $(\mathcal{F}_{\text{Samp}\mathbb{Z}}, \mathcal{F}_{\text{Blind}})$-hybrid model

Sample coset: On input $(\mathsf{cosetsample}, sid, v \in \mathbb{Z}_q)$, party i does:

- Call $\mathcal{F}_{\text{Samp}\mathbb{Z}}(\mathsf{sample}, sid, k \times 1, s_{\mathbf{g}}, 1)$ and receive $(\mathsf{sample}, sid, [\![\mathbf{z}]\!]^i)$.
- Call $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, \langle \mathbf{g}, [\![\mathbf{z}]\!]^i \rangle \bmod q)$ and receive $(\mathsf{blind}, sid, [\![u]\!]^i)$.
- Broadcast $[\![u]\!]^i$ and reconstruct $u = \langle \mathbf{g}, \mathbf{z} \rangle$ from the broadcast shares.
- If $u = v$, output $(\mathsf{cosetsample}, sid, [\![\mathbf{z}]\!]^i)$. Otherwise, repeat.

Figure 17: Protocol for gadget operations

Simulator $\mathcal{S}_{\text{Gadget}}$

Sample coset: Upon receiving $(\mathsf{cosetsample}, sid, v)$ from $\mathcal{F}_{\text{Gadget}}$:

- Choose a uniformly random $u \in \mathbb{Z}_q$, and generate fresh sharings of u and of $\mathbf{z} = \mathbf{0} \in \mathbb{Z}_q^k$.
- For each currently corrupted party i, reveal to $\mathcal{Z}$ the following functionality outputs to party i:
  - $(\mathsf{sample}, sid, [\![\mathbf{z}]\!]^i)$ on behalf of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$;
  - $(\mathsf{blind}, sid, [\![u]\!]^i)$ on behalf of $\mathcal{F}_{\text{Blind}}$.
- Broadcast $[\![u]\!]^i$ on behalf of all honest parties i.
- Unless $u = v$, repeat.

Corruption: When $\mathcal{Z}$ requests to corrupt party i, for each previous call to $\mathsf{cosetsample}$, reveal the corresponding messages $(\mathsf{sample}, sid, [\![\mathbf{z}]\!]^i)$ and $(\mathsf{blind}, sid, [\![u]\!]^i)$ to party i on behalf of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ and $\mathcal{F}_{\text{Blind}}$, respectively.

Figure 18: Simulator for π_{Gadget}

adversary, the value of $\mathbf{z} = \mathbf{0}$ and honest parties' shares remain information theoretically hidden. The announced (blinded) shares of u in the protocol π_{Gadget} form a uniformly random (and independent of the honest parties' outputs $[\![\mathbf{z}]\!]^i$) sharing of the (nearly) uniformly random syndrome $u = \langle \mathbf{g}, \mathbf{z} \rangle$, which is exactly what $\mathcal{S}_{\text{Gadget}}$ constructs to simulate the broadcast messages. A full proof is a straightforward application of these observations and of the privacy, robustness, and homomorphic properties of secret sharing.

A.3.2 Protocol and Security Analysis

The protocol π_{Correct} in the $(\mathcal{F}_{\text{Mult}}, \mathcal{F}_{\text{Gadget}})$-hybrid model is defined formally in Figure 19. In the initialization step, for each $j \in [n]$ and $v \in \mathbb{Z}_q$ the parties populate each of their local queues $Q_{j,v}$ with at least B entries, in the following way: each party i uses $\mathcal{F}_{\text{Gadget}}$, its shares of the trapdoor $\mathbf{R}$, and $\mathcal{F}_{\text{Mult}}$ to obtain a share of $\mathbf{y}_{j,v} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right](\mathbf{e}_j \otimes \mathbf{z}_{j,v})$ for Gaussian-distributed $\mathbf{z}_{j,v} \in \Lambda_v^{\perp}(\mathbf{g}^t)$, and places the share in a queue $Q_{j,v}$. (Regarding the arguments to the call to $\mathcal{F}_{\text{Mult}}$, note that $\left[\begin{smallmatrix}[\![\mathbf{R}]\!]^i\\ \mathbf{I}\end{smallmatrix}\right]$ is a valid ith share of $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$, via a constant sharing polynomial for $\mathbf{I}$, and $\mathbf{e}_{j,v} \otimes [\![\mathbf{z}_j]\!]^i$ is similarly a valid ith share of $\mathbf{e}_j \otimes \mathbf{z}_{j,v}$.) To later answer a $\mathsf{correct}$ request for syndrome $\mathbf{v} \in \mathbb{Z}_q^n$, each party just draws a share from each of $Q_{1,v_1}, \ldots, Q_{n,v_n}$ and sums these shares. By the homomorphic properties of secret sharing, this yields a share of $\mathbf{y} = \sum_{j \in [n]} \mathbf{y}_{j,v_j} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ for Gaussian $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n) \in \Lambda_{\mathbf{v}}^{\perp}(\mathbf{G})$, as desired.

Theorem A.2. *Protocol π_{Correct} statistically realizes $\mathcal{F}_{\text{Correct}}$ in the $(\mathcal{F}_{\text{Mult}}, \mathcal{F}_{\text{Gadget}})$-hybrid model for t-limited environments in $\mathscr{Z}$.*

Protocol π_{Correct} in the $(\mathcal{F}_{\text{Mult}}, \mathcal{F}_{\text{Gadget}})$-hybrid model

Initialize: On input $(\mathsf{init}, sid, [\![\mathbf{R}]\!]^i, B)$, party i does:

- Locally store $(sid, [\![\mathbf{R}]\!]^i)$ and initialize local queues $Q_{j,v}$ for each $j \in [n]$ and $v \in \mathbb{Z}_q$.
- For each $j \in [n]$, while there exists some $v \in \mathbb{Z}_q$ such that $Q_{j,v}$ has fewer than B entries:
  - Call $\mathcal{F}_{\text{Gadget}}(\mathsf{cosetsample}, sid, v)$ and receive $(\mathsf{cosetsample}, sid, [\![\mathbf{z}_{j,v}]\!]^i)$.
  - Call $\mathcal{F}_{\text{Mult}}(\mathsf{mult}, sid, \begin{bmatrix} [\![\mathbf{R}]\!]^i \\ \mathbf{I} \end{bmatrix}, \mathbf{e}_j \otimes [\![\mathbf{z}_{j,v}]\!]^i)$ and receive $(\mathsf{mult}, sid, [\![\mathbf{y}_{j,v}]\!]^i)$, where $\mathbf{y}_{j,v} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} (\mathbf{e}_j \otimes \mathbf{z}_{j,v})$.
  - Place $[\![\mathbf{y}_{j,v}]\!]^i$ in local queue $Q_{j,v}$.
- Output (init, sid).

Correct: On input $(\mathsf{correct}, sid, \mathbf{v})$, if fewer than B calls to correct have already been made, party i does:

- For each $j \in [n]$, dequeue an entry $[\![\mathbf{y}_{j,v}]\!]^i$ from Q_{j,v_j}.
- Locally compute $[\![\mathbf{y}]\!]^i = \sum_{j \in [n]} [\![\mathbf{y}_{j,v}]\!]^i$.
- Output $(\mathsf{correct}, sid, [\![\mathbf{y}]\!]^i)$.

Figure 19: Syndrome correction protocol

Simulator $\mathcal{S}_{\text{Correct}}$

Initialize: Upon receiving (init, sid, B) from $\mathcal{F}_{\text{Correct}}$:

- Initialize empty lists $Q_{j,v}$ for each $j \in [n]$ and $v \in \mathbb{Z}_q$.
- For each $j \in [n]$, while there exists some $v \in \mathbb{Z}_q$ such that $Q_{j,v}$ has fewer than B unused entries:
  - Generate a fresh sharing $[\![\mathbf{z}_{j,v}]\!]$ of $\mathbf{z}_{j,v} = \mathbf{0} \in \mathbb{Z}_q^k$, and send $(\mathsf{cosetsample}, sid, [\![\mathbf{z}_{j,v}]\!]^i)$ on behalf of $\mathcal{F}_{\text{Gadget}}$ to each currently corrupted party i in $\mathcal{P}$.
  - Generate a fresh sharing $[\![\mathbf{y}_{j,v}]\!]$ for $\mathbf{y}_{j,v} = \mathbf{0} \in \mathbb{Z}_q^m$, and send $(\mathsf{mult}, sid, [\![\mathbf{y}_{j,v}]\!]^i)$ on behalf of $\mathcal{F}_{\text{Mult}}$ to each currently corrupted i in $\mathcal{P}$.
  - Store $([\![\mathbf{z}_{j,v}]\!], [\![\mathbf{y}_{j,v}]\!])$ as an unused entry at the end of list $Q_{j,v}$.

Correct: Upon receiving $(\mathsf{correct}, sid, \mathbf{v})$ from $\mathcal{F}_{\text{Correct}}$:

- For each $j \in [n]$, look up the next unused entry $([\![\mathbf{z}_{j,v_j}]\!], [\![\mathbf{y}_{j,v_j}]\!])$ from Q_{j,v_j}, and mark it as used for this call to correct. For each currently corrupted party i in $\mathcal{P}$, send $[\![\mathbf{y}]\!]^i = \sum_{j \in [n]} [\![\mathbf{y}_{j,v}]\!]^i$ to $\mathcal{F}_{\text{Correct}}$ as the desired share for party i.

Corruption: When $\mathcal{Z}$ requests to corrupt party i,

- Receive party i's share $[\![\mathbf{y}]\!]^i$ for each previous call of the form $(\mathsf{correct}, sid, \mathbf{v})$. Look up the n corresponding (used) entries $([\![\mathbf{z}_{j,v_j}]\!], [\![\mathbf{y}_{j,v_j}]\!])$ in Q_{j,v_j}, and update the value $[\![\mathbf{y}_{n,v_n}]\!]^i$ so that $[\![\mathbf{y}]\!]^i = \sum_{j \in [n]} [\![\mathbf{y}_{j,v_j}]\!]^i$.
- For all entries $([\![\mathbf{z}_{j,v}]\!], [\![\mathbf{y}_{j,v}]\!])$, both used and unused, in each list $Q_{j,v}$, reveal to $\mathcal{Z}$ the messages $(\mathsf{cosetsample}, sid, [\![\mathbf{z}_{j,v}]\!]^i)$ and $(\mathsf{mult}, sid, [\![\mathbf{y}_{j,v}]\!]^i)$ to party i on behalf of $\mathcal{F}_{\text{Gadget}}$ and $\mathcal{F}_{\text{Mult}}$, respectively.

Figure 20: Simulator for π_{Correct}

Proof sketch. A simulator $\mathcal{S}_{\text{Correct}}$ for demonstrating the security of π_{Correct} is provided in Figure 20. The only subtlety lies in the fact that outputs from helper functionalities during precomputation must be simulated before knowing which parties will be corrupted when the corresponding correct calls are made later on. As mentioned in Section 3.2.2, this is why we designed $\mathcal{F}_{\text{Correct}}$ to ask the adversary for shares for the corrupted

parties: the simulator generates its own shares when simulating the precomputation, and provides them to the functionality upon request. Then security boils down to the fact that even though $\mathbf{y}_{j,v_j} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right](\mathbf{e}_j \otimes \mathbf{z}_{j,v_j})$, all shares of $\mathbf{z}_{j,v_j}$ (that the adversary sees) are uniform and independent of the corresponding shares of $\mathbf{y}_{j,v_j}$ since $\mathcal{F}_{\text{Mult}}$ blinds its output, so the queuing strategy employed by π_{Correct} indeed produces shares of $\mathbf{y}$ with the desired distribution. □

A.4 Realizing $\mathcal{F}_{\text{Perturb}}$

Recall that $\mathcal{F}_{\text{Perturb}}$ (Figure 3) distributed shares of of perturbations $\mathbf{p}$ drawn from the discrete Gaussian distribution $D_{\mathbb{Z}^m,\sqrt{\Sigma_{\mathbf{p}}}}$, where the covariance $\Sigma_{\mathbf{p}}$ depends on the trapdoor $\mathbf{R}$. In the standalone setting, this is straightforward: first generate a continuous Gaussian $\mathbf{p}' \in \mathbb{R}^m$ with covariance $\Sigma_{\mathbf{p}} - \mathbf{I} \cdot \omega_n^2$, then randomly round each coordinate of $\mathbf{p}'$ to a nearby integer (see [Pei10] for details). In the threshold setting, generating a good perturbation seems quite a bit more difficult, because neither $\mathbf{p}$ nor its covariance $\Sigma_{\mathbf{p}}$ can be revealed, since they leak the trapdoor $\mathbf{R}$. Fortunately, we can give a distributed protocol that emulates the standalone rounding procedure up to sufficient precision.

A.4.1 Extending $\mathcal{F}_{\text{Samp}\mathbb{Z}}$

We first extend the functionality $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ (originally defined in Section A.1) with two additional commands, cosetsample and rround, which support randomized rounding of a shared value $x \in q^{-j}\mathbb{Z}$ to the integers $\mathbb{Z}$. Note that while cosetsample and rround are defined for (scalings of) the integer lattice $\mathbb{Z}$, the commands immediately generalize to vectors and matrices, component-wise. (This is simply because spherical Gaussians over cosets of $\mathbb{Z}^{h\times w}$ are just product distributions of Gaussians over cosets of $\mathbb{Z}$.) This extended functionality is defined formally in Figure 21.

Functionality $\mathcal{F}_{\text{Samp}\mathbb{Z}}$

Sample: Upon receiving $(\mathsf{sample}, sid, h \times w, z, d)$ from at least h honest parties in $\mathcal{P}$:

- Sample $\mathbf{X} \leftarrow D^{h\times w}_{\mathbb{Z},z\cdot\omega_n}$ and generate a fresh sharing $[\![\mathbf{X}]\!]$ over $\mathbb{Z}_{q^d}$.
- Send $(\mathsf{sample}, sid, [\![\mathbf{X}]\!]^i)$ to each party i in $\mathcal{P}$ and $(\mathsf{sample}, sid, h \times w, z, d)$ to the adversary.

Sample coset: Upon receiving $(\mathsf{cosetsample}, sid, v \in q^{-j}\mathbb{Z}/q^{-j+1}\mathbb{Z}, z \geq q^{-j+1})$ from at least h honest parties in $\mathcal{P}$:

- Sample $x \leftarrow D_{q^{-j+1}\mathbb{Z}+v,z\cdot\omega_n}$ and let $c = x \bmod \mathbb{Z}$.
- Generate a fresh sharing $[\![x]\!]$ over $q^{-j}\mathbb{Z}/q\mathbb{Z}$.
- Send $(\mathsf{cosetsample}, sid, [\![x]\!]^i, c)$ to each party i in $\mathcal{P}$, and $(\mathsf{cosetsample}, sid, c, z)$ to the adversary.

Randomized round: Upon receiving $(\mathsf{rround}, sid, [\![x]\!]^i \in q^{-d}\mathbb{Z}/q\mathbb{Z})$ from at least h honest parties in $\mathcal{P}$:

- Reconstruct x and let $c = x \bmod \mathbb{Z}$. Sample an integer $z \leftarrow x + D_{\mathbb{Z}-c,d\cdot\omega_n}$.
- Generate a fresh sharing $[\![z]\!]$ over $\mathbb{Z}_q$, and send $(\mathsf{rround}, sid, [\![z]\!]^i, c)$ to each party i in $\mathcal{P}$, and $(\mathsf{rround}, sid, c)$ to the adversary.

Figure 21: Full integer sampling functionality (which replaces Figure 13)

A protocol $\pi_{\text{Samp}\mathbb{Z}}$ realizing $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ in the $\mathcal{F}_{\text{Blind}}$-hybrid model is given in Figure 22. We elaborate informally on the implementation of the two additional commands, noting that implementation of the sample command was discussed in Section A.1. We omit a formal security proof for $\pi_{\text{Samp}\mathbb{Z}}$, but we remark that while

the protocol is somewhat more complicated than the key-generation protocol of Section A.2, its security is based straightforwardly on the same main observations, plus Lemma 2.5.

Coset sampling. The cosetsample command (which exists mainly to support the randomized-rounding command, described next) generates a discrete Gaussian variable x with given parameter $z \geq q^{-j+1}$ over the (possibly very dense) lattice $q^{-j}\mathbb{Z}$, such that x's least significant base-q digit is a specified v. It then distributes shares $[\![x]\!]^i$ to the respective parties, along with the value $c = x \bmod \mathbb{Z}$, which also goes to the adversary.

Naïvely implementing cosetsample in our protocol is very simple: the parties just use sample to generate a Gaussian value $q^j x \in \mathbb{Z}$ with parameter $q^j z \cdot \omega_n$, shared over $\mathbb{Z}_{q^{j+1}}$, then reveal their blinded shares $x \bmod \mathbb{Z}$ to reconstruct c, repeating until the least-significant digit is v. Because $z \geq q^{-j+1}$ is large enough, x's least significant digit is nearly uniform by Corollary 2.3, and the expected number of trials is almost exactly $q = \mathrm{poly}(n)$. Of course, this procedure throws away many samples; a more efficient implementation would precompute many trials and store the results according to least-significant digit, so that the online phase of the command becomes just a noninteractive table lookup. For simplicity, we formally define only the naive implementation.

Randomized rounding. The rround command takes shares of a value $x \in q^{-d}\mathbb{Z}$ (represented modulo $q\mathbb{Z}$) and rounds it to an integer $z \in \mathbb{Z}$ using Gaussian rounding. It returns shares $[\![z]\!]^i$ to the respective parties, along with the coset $c = x \bmod \mathbb{Z}$ of the original input, which also goes to the adversary. In our protocol, the parties broadcast their blinded shares of $c = x \bmod \mathbb{Z}$ to reconstruct c, then use d calls to cosetsample to round x one digit at a time, from least- to most-significant digit. Note that each call to cosetsample alters the more-significant digits of $x \bmod \mathbb{Z}$, but these changes are public.

A.4.2 Protocol and Security Analysis

In brief, our perturbation protocol starts by generating a sharing of a sufficiently high-precision approximation $\mathbf{P} \approx \sqrt{\Sigma_{\mathbf{p}}}$ with some d digits of precision in its fractional part (i.e., the entries of $\mathbf{P}$ are in $q^{-d}\mathbb{Z}$). The sharing of $\mathbf{P}$ can be precomputed as part of the key-generation phase, using general multiparty computation. To generate a perturbation vector $\mathbf{p}$, the protocol first generates a sharing of a high-precision Gaussian random variable $\mathbf{p}' \in q^{-d}\mathbb{Z}^m$ having covariance $\mathbf{P}\mathbf{P}^t \approx \Sigma_{\mathbf{p}}$. It does this by invoking $\mathcal{F}_{\text{Samp}\mathbb{Z}}$ to generate shares of a Gaussian-distributed $\mathbf{z} \in \mathbb{Z}^m$, and then invoking $\mathcal{F}_{\text{Mult}}$ to get a fresh sharing of $\mathbf{p}' = \mathbf{P}\mathbf{z}$. The parties then randomize-round their shared $\mathbf{p}' \in q^{-d}\mathbb{Z}^m$ to a shared final perturbation vector $\mathbf{p} \in \mathbb{Z}^m$, using the rround command of $\mathcal{F}_{\text{Samp}\mathbb{Z}}$. (Recall from A.4.1 that this command reveals $\mathbf{c} = \mathbf{p}' \bmod \mathbb{Z}^m$ publicly.) Finally, using the secret-sharing homomorphisms the parties also reconstruct the two syndromes $\bar{\mathbf{w}}$ and $\bar{\mathbf{w}}$. (Recall that these are eventually needed by the full Gaussian sampling protocol π_{GS}.) Note that once the sharing of $\mathbf{P}$ is computed once and for all, the only trapdoor-dependent work is the relatively efficient call to $\mathcal{F}_{\text{Mult}}$.

For analyzing security, the essence of the argument is that the public residue $\mathbf{c} = \mathbf{p}' \bmod \mathbb{Z}^m$ returned by $\mathcal{F}_{\text{Samp}\mathbb{Z}}$.rround is (nearly) uniformly random, and hence simulatable without knowing $\mathbf{p}'$, because $\mathbf{p}'$ is drawn from a Gaussian whose parameter exceeds the smoothing parameter of $\mathbb{Z}^m$. All the remaining functionalities simply return independent and properly blinded shares of intermediate values to their respective owners, and so are trivial to simulate. Therefore, we omit a formal simulator and security proof for π_{Perturb}, which are tedious (though straightforward given the above intuition).

Protocol $\pi_{\text{Samp}\mathbb{Z}}$ in the $\mathcal{F}_{\text{Blind}}$-hybrid model

Sample: On input $(\mathsf{sample}, sid, h \times w, z, d)$, party i does:

- With the other parties, run an inverse sampling protocol (see A.1 for elaboration) to generate private output $[\![\mathbf{X}]\!]^i$, where $\mathbf{X} \leftarrow D^{h\times w}_{\mathbb{Z},z\cdot\omega_n}$.
- Output $(\mathsf{sample}, sid, [\![\mathbf{X}]\!]^i)$.

Sample coset: On input $(\mathsf{cosetsample}, sid, v \in q^{-j}\mathbb{Z}/q^{-j+1}\mathbb{Z}, z \geq q^{-j+1})$, party i does:

- Call $(\mathsf{sample}, sid, 1 \times 1, q^j z, j+1)$ and receive $(\mathsf{sample}, sid, q^j [\![x]\!]^i)$ with $x \in q^{-j}\mathbb{Z}$.
- Call $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, [\![x]\!]^i \bmod \mathbb{Z})$ and receive $(\mathsf{blind}, sid, [\![c]\!]^i)$.
- Announce $[\![c]\!]^i$ and reconstruct c from other parties' announced shares.
- If $c \bmod q^{-j+1} = v$, output $(\mathsf{cosetsample}, sid, [\![x]\!]^i)$. Otherwise, repeat.

Randomized round: On input $(\mathsf{rround}, sid, [\![x]\!]^i \in q^{-d}\mathbb{Z}/q\mathbb{Z})$, party i does:

- Call $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, [\![x]\!]^i \bmod \mathbb{Z})$ and receive a fresh share $[\![c]\!]^i$ of $c = x \bmod \mathbb{Z}$. Broadcast $[\![c]\!]^i$.
- Reconstruct $c \in q^{-d}\mathbb{Z}/\mathbb{Z}$ from the announced shares.
- Let $v = c$ and $[\![z]\!]^i = [\![x]\!]^i$. For $j = d, \ldots, 1$:
  - Call $(\mathsf{cosetsample}, sid, -v \bmod q^{-j+1}\mathbb{Z}, \sqrt{d})$ as a subroutine, and receive back $(\mathsf{cosetsample}, sid, [\![x']\!]^i \in q^{-j}\mathbb{Z}/q\mathbb{Z}, c' \in q^{-j}\mathbb{Z}/\mathbb{Z})$.
  - Let $v \leftarrow v + c' \in q^{-j+1}\mathbb{Z}/\mathbb{Z}$ and $[\![z]\!]^i \leftarrow [\![z]\!]^i + [\![x']\!]^i \in q^{-j}\mathbb{Z}/q\mathbb{Z}$, which is an ith share of $z + x' \in q^{-j+1}\mathbb{Z}/q\mathbb{Z}$.
  - Truncate $[\![z]\!]^i$ to lie in $q^{-j+1}\mathbb{Z}/q\mathbb{Z}$ (without changing the underlying shared value) as described in Section 2.3, i.e., let $[\![z]\!]^i \leftarrow [\![z]\!]^i - ([\![z]\!]^i \bmod q^{-j+1})$.
- Output $(\mathsf{rround}, sid, [\![z]\!]^i, c)$.

Figure 22: Integer sampling protocol

Protocol π_{Perturb} in the $(\mathcal{F}_{\text{Blind}}, \mathcal{F}_{\text{Mult}}, \mathcal{F}_{\text{Samp}\mathbb{Z}})$-hybrid model

In what follows, define the quotient ring $G_d = q^{-d}\mathbb{Z}/q\mathbb{Z}$.

Initialize: On input $(\mathsf{init}, sid, \mathbf{A}_{-\mathbf{H}^*}, [\![\mathbf{R}]\!]^i, s, B)$, party i does:

- With the other parties, run a statistically secure multiparty computation protocol to compute (as a private output) $[\![\mathbf{P}]\!]^i$ shared over G_d, where $\Sigma_{\mathbf{p}} = s^2 - s_{\mathbf{g}}^2 \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \left[\begin{smallmatrix}\mathbf{R}^t & \mathbf{I}\end{smallmatrix}\right]$, and $\mathbf{P} \approx \sqrt{\Sigma_{\mathbf{p}} - d^2 \cdot \omega_n^2}$
- Locally store sid, $\mathbf{A}_{-\mathbf{H}^*}$, and $[\![\mathbf{P}]\!]^i$, and initialize a local queue Q.
- While Q has fewer than B entries:
  - Call $\mathcal{F}_{\text{Samp}\mathbb{Z}}(\mathsf{sample}, sid, m \times 1, 1, d+1)$ and receive $(\mathsf{sample}, sid, [\![\mathbf{z}]\!]^i)$ for some $\mathbf{z} \leftarrow D^m_{\mathbb{Z},\omega_n}$ that is shared over $\mathbb{Z}_{q^{d+1}}$.
  - Call $\mathcal{F}_{\text{Mult}}(\mathsf{mult}, sid, q^d \cdot [\![\mathbf{P}]\!]^i, [\![\mathbf{z}]\!]^i)$ and receive $(\mathsf{mult}, sid, q^d \cdot [\![\mathbf{p}']\!]^i)$ where $\mathbf{p}' = \mathbf{P}\mathbf{z} \in G_d^m$. (Above we are multiplying and dividing shares by q^d simply to compute an isomorphism between G_d and $\mathbb{Z}_{q^{d+1}}$, because $\mathcal{F}_{\text{Mult}}$ expects to receive and return shares over the latter ring.)
  - Call $\mathcal{F}_{\text{Samp}\mathbb{Z}}(\mathsf{rround}, sid, [\![\mathbf{p}']\!]^i)$ and receive $(\mathsf{rround}, sid, [\![\mathbf{p}]\!]^i \in \mathbb{Z}_q^m, \mathbf{c} = \mathbf{p}' \bmod \mathbb{Z}^m)$, where $\mathbf{p}$ has distribution $\mathbf{p} + D_{\mathbb{Z}^m - \mathbf{p}', d\cdot\omega_n} \in \mathbb{Z}^m$.
  - Call $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, [\mathbf{0} \mid \mathbf{G}] \cdot [\![\mathbf{p}]\!]^i)$ and $\mathcal{F}_{\text{Blind}}(\mathsf{blind}, sid, \mathbf{A}_{-\mathbf{H}^*} \cdot [\![\mathbf{p}]\!]^i)$ and receive back $[\![\mathbf{w}]\!]^i$ and $[\![\bar{\mathbf{w}}]\!]^i$, respectively. Broadcast these shares.
  - Reconstruct $\mathbf{w}$ and $\bar{\mathbf{w}}$ from the announced shares, and put $([\![\mathbf{p}]\!]^i, \bar{\mathbf{w}}, \mathbf{w})$ in local queue Q.
- Output (init, sid).

Perturb: On input $(\mathsf{perturb}, sid)$, if fewer than B calls to perturb have already been made, party i does:

- Dequeue $([\![\mathbf{p}]\!]^i, \mathbf{w}, \bar{\mathbf{w}})$ from local queue Q.
- Output $(\mathsf{perturb}, sid, [\![\mathbf{p}]\!]^i, \mathbf{w}, \bar{\mathbf{w}})$.

Figure 23: Perturbation protocol

Practical Bootstrapping in Quasilinear Time

Jacob Alperin-Sheriff* Chris Peikert†

October 9, 2013

Abstract

Gentry's "bootstrapping" technique (STOC 2009) constructs a fully homomorphic encryption (FHE) scheme from a "somewhat homomorphic" one that is powerful enough to evaluate its own decryption function. To date, it remains the only known way of obtaining unbounded FHE. Unfortunately, bootstrapping is computationally very expensive, despite the great deal of effort that has been spent on improving its efficiency. The current state of the art, due to Gentry, Halevi, and Smart (PKC 2012), is able to bootstrap "packed" ciphertexts (which encrypt up to a linear number of bits) in time only *quasilinear* $\tilde{O}(\lambda) = \lambda \cdot \log^{O(1)} \lambda$ in the security parameter. While this performance is *asymptotically* optimal up to logarithmic factors, the practical import is less clear: the procedure composes multiple layers of expensive and complex operations, to the point where it appears very difficult to implement, and its concrete runtime appears worse than those of prior methods (all of which have quadratic or larger asymptotic runtimes).

In this work we give *simple*, *practical*, and entirely *algebraic* algorithms for bootstrapping in quasilinear time, for both "packed" and "non-packed" ciphertexts. Our methods are easy to implement (especially in the non-packed case), and we believe that they will be substantially more efficient in practice than all prior realizations of bootstrapping. One of our main techniques is a substantial enhancement of the "ring-switching" procedure of Gentry et al. (SCN 2012), which we extend to support switching between two rings where neither is a subring of the other. Using this procedure, we give a natural method for homomorphically evaluating a broad class of structured linear transformations, including one that lets us evaluate the decryption function efficiently.

*School of Computer Science, College of Computing, Georgia Institute of Technology. Email: jmas6@cc.gatech.edu

†School of Computer Science, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by the Alfred P. Sloan Foundation, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Sloan Foundation, DARPA or the U.S. Government.

1 Introduction

Bootstrapping, a central technique from the breakthrough work of Gentry [Gen09b, Gen09a] on fully homomorphic encryption (FHE), converts a sufficiently powerful "somewhat homomorphic" encryption (SHE) scheme into a fully homomorphic one. (An SHE scheme can support a bounded number of homomorphic operations on freshly generated ciphertexts, whereas an FHE scheme has no such bound.) In short, bootstrapping works by *homomorphically* evaluating the SHE scheme's decryption function on a ciphertext that cannot support any further homomorphic operations. This has the effect of "refreshing" the ciphertext, i.e., it produces a new one that encrypts the same message and can handle more homomorphic operations. Bootstrapping remains the only known way to achieve *unbounded* FHE, i.e., a scheme that can homomorphically evaluate any efficient function using keys and ciphertexts of a fixed size.[1]

In order to be "bootstrappable," an SHE scheme must be powerful enough to homomorphically evaluate its own decryption function, using whatever homomorphic operations it supports. For security reasons, the key and ciphertext sizes of all known SHE schemes grow with the *depth* and, to a lesser extent, the *size* of the functions that they can homomorphically evaluate. For instance, under plausible hardness conjectures, the key and ciphertext sizes of the most efficient SHE scheme to date [BGV12] grow quasilinearly in both the supported multiplicative depth d and the security parameter λ, i.e., as $\tilde{O}(d \cdot \lambda)$. Clearly, the runtime of bootstrapping must also grow with the sizes of the keys, ciphertexts, and decryption function. This runtime is perhaps the most important measure of efficiency for FHE, because bootstrapping is currently the biggest bottleneck by far in instantiations, both in theory and in practice.

The past few years have seen an intensive study of different forms of decryption procedures for SHE schemes, and their associated bootstrapping operations [Gen09b, Gen09a, vDGHV10, GH11b, BV11a, GH11a, BGV12, GHS12b]. The first few bootstrapping methods had moderate polynomial runtimes in the security parameter λ, e.g., $\tilde{O}(\lambda^4)$. Brakerski, Gentry, and Vaikuntanathan [BGV12] gave a major efficiency improvement, reducing the runtime to $\tilde{O}(\lambda^2)$. They also gave an amortized method that bootstraps $\tilde{\Omega}(\lambda)$ ciphertexts at once in $\tilde{O}(\lambda^2)$ time, i.e., quasilinear runtime per ciphertext. However, these results apply only to "non-packed" ciphertexts, i.e., ones that encrypt essentially just one bit each, which combined with the somewhat large runtimes makes these methods too inefficient to be used very much in practice. Most recently, Gentry, Halevi, and Smart [GHS12a] achieved bootstrapping for "packed" ciphertexts (i.e., ones that encrypt up to $\tilde{\Omega}(\lambda)$ bits each) in *quasilinear* $\tilde{O}(\lambda)$ runtime, which is asymptotically optimal in space and time, up to polylogarithmic factors. For this they relied on a general "compiler" from another work of theirs [GHS12b], which achieved SHE/FHE for sufficiently wide circuits with polylogarithmic multiplicative "overhead," i.e., cost relative to evaluating the circuit "in the clear."

Bootstrapping and FHE in quasi-optimal time and space is a very attractive and powerful theoretical result. However, the authors of [GHS12b, GHS12a] caution that their constructions may have limited potential for use in practice, for two main reasons: first, the runtimes, while asymptotically quasilinear, include very large polylogarithmic factors. For realistic values of the security parameter, these polylogarithmic terms exceed the rather small (but asymptotically worse) quasilinear overhead obtained in [BGV12]. The second reason is that their bootstrapping operation is algorithmically very complex and difficult to implement (see the next paragraphs for details). Indeed, while there are now a few working implementations of bootstrapping (e.g., [GH11b, CCK+13]) that follow the templates from [Gen09b, Gen09a, vDGHV10, BGV12], we are not aware of any attempt to implement any method having subquadratic runtime.

[1] This stands in contrast with *leveled* FHE schemes, which can homomorphically evaluate a function of any *a priori* bounded depth, but using keys and ciphertexts whose sizes depend on the bound. Leveled FHE can be constructed without resorting to bootstrapping [BGV12].

Is quasilinear efficient? The complexity and large practical overhead of the constructions in [GHS12b, GHS12a] arise from two kinds of operations. First, the main technique from [GHS12b] is a way of homomorphically evaluating any sufficiently shallow and wide arithmetic circuit on a "packed" ciphertext that encrypts a high-dimensional vector of plaintexts in multiple "slots." It works by first using ring automorphisms and key-switching operations [BV11a, BGV12] to obtain a small, fixed set of "primitive" homomorphic permutations on the slots. It then composes those permutations (along with other homomorphic operations) in a log-depth permutation network, to obtain any permutation. Finally, it homomorphically evaluates the desired circuit by combining appropriate permutations with relatively simple homomorphic slot-selection and ring operations.

In the context of bootstrapping, one of the key observations from [GHS12a] is that a main step of the decryption procedure can be evaluated using the above technique. Specifically, they need an operation that moves the coefficients of an encrypted plaintext polynomial, reduced modulo a cyclotomic polynomial $\Phi_m(X)$, into the slots of a packed ciphertext (and back again). Once the coefficients are in the slots, they can be rounded in a batched (SIMD) fashion, and then mapped back to coefficients of the plaintext. The operations that move the coefficients into slots and vice-versa can be expressed as $O(\log \lambda)$-depth arithmetic circuits of size $O(\lambda \log \lambda)$, roughly akin to the classic FFT butterfly network. Hence they can be evaluated homomorphically with polylogarithmic overhead, using [GHS12b]. However, as the authors of [GHS12a] point out, the decryption circuit is quite large and complex – especially the part that moves the slots back to the coefficients, because it involves reduction modulo $\Phi_m(X)$ for an m having several prime divisors. This modular reduction is the most expensive part of the decryption circuit, and avoiding it is one of the main open problems given in [GHS12a]. However, even a very efficient decryption circuit would still incur the large polylogarithmic overhead factors from the techniques of [GHS12b].

1.1 Our Contributions

We give a new bootstrapping algorithm that runs in *quasilinear* $\tilde{O}(\lambda)$ time per ciphertext with *small* polylogarithmic factors, and is algorithmically much simpler than previous methods. It is easy to implement, and we believe that it will be substantially more efficient in practice than all prior methods. We provide a unified bootstrapping procedure that works for both "non-packed" ciphertexts (which encrypt integers modulo some p, e.g., bits) and "packed" ciphertexts (which encrypt elements of a high-dimensional ring), and also interpolates between the two cases to handle an intermediate concept we call "semi-packed" ciphertexts.

Our procedure for non-packed ciphertexts is especially simple and efficient. In particular, it can work very naturally using only cyclotomic rings having power-of-two index, i.e., rings of the form $\mathbb{Z}[X]/(1 + X^{2^k})$, which admit very fast implementations. This improves upon the method of [BGV12], which achieves quasilinear *amortized* runtime when bootstrapping $\tilde{\Omega}(\lambda)$ non-packed ciphertexts at once. Also, while that method can also use power-of-two cyclotomics, it can only do so by emulating $\mathbb{Z}_2$ (bit) arithmetic within $\mathbb{Z}_p$ for some moderately large prime p, which translates additions in $\mathbb{Z}_2$ into much more costly multiplications in $\mathbb{Z}_p$. By contrast, our method works "natively" with any plaintext modulus.

For packed ciphertexts, our procedure draws upon high-level ideas from [GHS12b, GHS12a], but our approach is conceptually and technically very different. Most importantly, it completely avoids the two main inefficiencies from those works: first, unlike [GHS12b], we do not use permutation networks or any explicit permutations of the plaintext slots, nor do we rely on a general-purpose compiler for homomorphically evaluating arithmetic circuits. Instead, we give direct, practically efficient procedures for homomorphically mapping the coefficients of an encrypted plaintext element into slots and vice-versa. In particular, our procedure does not incur the large cost or algorithmic complexity of homomorphically reducing modulo $\Phi_m(X)$, which was the main bottleneck in the decryption circuit of [GHS12a].

At a higher level, our bootstrapping method has two other attractive and novel features: first, it is entirely "algebraic," by which we mean that the full procedure (including generation of all auxiliary data it uses) can be described as a short sequence of elementary operations from the "native instruction set" of the SHE scheme. By contrast, all previous methods at some point invoke rather generic arithmetic circuits, e.g., for modular addition of values represented as bit strings, or reduction modulo a cyclotomic polynomial $\Phi_m(X)$. Of course, arithmetic circuits can be evaluated using the SHE scheme's native operations, but we believe that the distinction between "algebraic" and "non-algebraic" is an important qualitative one, and it certainly affects the simplicity and concrete efficiency of the bootstrapping procedure.

The second nice feature of our method is that it completely decouples the algebraic structure of the SHE plaintext ring from that which is needed by the bootstrapping procedure. In previous methods that use amortization (or "batching") for efficiency (e.g., [SV11, BGV12, GHS12a]), the ring and plaintext modulus of the SHE scheme must be chosen so as to provide many plaintext slots. However, this structure may not always be a natural match for the SHE application's efficiency or functionality requirements. For example, the lattice-based pseudorandom function of [BPR12] works very well with a ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ where both q and n are powers of two, but for such parameters R_q has only *one* slot. Our method can bootstrap even for this kind of plaintext ring (and many others), while still using batching to achieve quasilinear runtime.

1.2 Techniques

At the heart of our bootstrapping procedure are two novel homomorphic operations for SHE schemes over cyclotomic rings: for non-packed (or semi-packed) ciphertexts, we give an operation that *isolates the message-carrying coefficient(s)* of a high-dimensional ring element; and for (semi-)packed ciphertexts, we give an operation that *maps coefficients to slots* and vice-versa.

Isolating coefficients. Our first homomorphic operation is most easily explained in the context of non-packed ciphertexts, which encrypt single elements of the quotient ring $\mathbb{Z}_p$ for some small modulus p, using ciphertexts over some cyclotomic quotient ring $R_q = R/qR$ of moderately large degree $d = \deg(R/\mathbb{Z}) = \tilde{O}(\lambda)$. We first observe that a ciphertext to be bootstrapped can be reinterpreted as an encryption of an R_q-element, one of whose $\mathbb{Z}_q$-coefficients (with respect to an appropriate basis of the ring) "noisily" encodes the message, and whose other coefficients are just meaningless noise terms. We give an simple and efficient homomorphic operation that preserves the meaningful coefficient, and maps all the others to zero. Having isolated the message-encoding coefficient, we can then homomorphically apply an efficient integer "rounding" function (see [GHS12a] and Appendix B) to recover the message from its noisy encoding, which completes the bootstrapping procedure. (Note that it is necessary to remove the meaningless noise coefficients first, otherwise they would interfere with the correct operation of the rounding function.)

Our coefficient-isolating procedure works essentially by applying the *trace function* $\mathrm{Tr}_{R/\mathbb{Z}} \colon R \to \mathbb{Z}$ to the plaintext. The trace is the "canonical" $\mathbb{Z}$-linear function from R to $\mathbb{Z}$, and it turns out that for the appropriate choice of $\mathbb{Z}$-basis of R used in decryption, the trace simply outputs (up to some scaling factor) the message-carrying coefficient we wish to isolate. One simple and very efficient way of applying the trace homomorphically is to use the "ring-switching" technique of [GHPS12], but unfortunately, this requires the ring-LWE problem [LPR10] to be hard over the target ring $\mathbb{Z}$, which is clearly not the case. Another way follows from the fact that $\mathrm{Tr}_{R/\mathbb{Z}}$ equals the sum of all d automorphisms of R; therefore, it can be computed by homomorphically applying each automorphism and summing the results. Unfortunately, this method takes at least *quadratic* $\Omega(\lambda^2)$ time, because applying each automorphism homomorphically takes $\Omega(\lambda)$ time, and there are $d = \Omega(\lambda)$ automorphisms.

So, instead of inefficiently computing the trace by summing all the automorphisms at once, we consider a *tower* of cyclotomic rings $\mathbb{Z} = R^{(0)} \subseteq R^{(1)} \subseteq \cdots \subseteq R^{(r)} = R$, usually written as $R^{(r)}/\cdots/R^{(1)}/R^{(0)}$. Then $\mathrm{Tr}_{R/\mathbb{Z}}$ is the composition of the individual trace functions $\mathrm{Tr}_{R^{(i)}/R^{(i-1)}}\colon R^{(i)} \to R^{(i-1)}$, and these traces are equal to the sums of all automorphisms of $R^{(i)}$ that fix $R^{(i-1)}$ pointwise, of which there are exactly $d_i = \deg(R^{(i)}/R^{(i-1)}) = \deg(R^{(i)}/\mathbb{Z})/\deg(R^{(i-1)}/\mathbb{Z})$. We can therefore compute each $\mathrm{Tr}_{R^{(i)}/R^{(i-1)}}$ in time linear in λ and in d_i; moreover, the number of trace functions to apply is at most logarithmic in $d = \deg(R/\mathbb{Z}) = \tilde{O}(\lambda)$, because each one reduces the degree by a factor of at least two. Therefore, by ensuring that the degrees of $R^{(r)}, R^{(r-1)}, \ldots, R^{(0)}$ decrease gradually enough, we can homomorphically apply the full $\mathrm{Tr}_{R/\mathbb{Z}}$ in quasilinear time. For example, a particularly convenient choice is to let $R^{(i)}$ be the 2^{i+1}st cyclotomic ring $\mathbb{Z}[X]/(1 + X^{2^i})$ of degree 2^i, so that every $d_i = 2$, and there are exactly $\log_2(d) = O(\log \lambda)$ trace functions to apply.

More generally, when bootstrapping a *semi-packed* ciphertext we start with a plaintext value in R_q that noisily encodes a message in S_p, for some subring $S \subseteq R$. (The case $S = \mathbb{Z}$ corresponds to a non-packed ciphertext.) We show that applying the trace function $\mathrm{Tr}_{R/S}$ to the R_q-plaintext yields a new plaintext *in* S_q that noisily encodes the message, thus isolating the meaningful part of the noisy encoding and vanishing the rest. We then homomorphically apply a rounding function to recover the S_p message from its noisy S_q encoding, which uses the technique described next.

Mapping coefficients to slots. Our second technique, and main technical innovation, is in bootstrapping (semi-)packed ciphertexts. We enhance the recent "ring-switching" procedure of [GHPS12], and use it to efficiently move "noisy" plaintext coefficients (with respect to an appropriate decryption basis) into slots for batch-rounding, and finally move the rounded slot values back to coefficients. We note that all previous methods for loading plaintext data into slots used the *same* ring for the source and destination, and so required the plaintext to come from a ring designed to have many slots. In this work, we use ring-switching to go from the SHE plaintext ring to a *different* ring having many slots, which is used only temporarily for batch-rounding. This is what allows the SHE plaintext ring to be decoupled from the rings used in bootstrapping, as mentioned above.

To summarize our technique, we first recall the ring-switching procedure of [GHPS12]. It was originally devised to provide moderate efficiency gains for SHE/FHE schemes, by allowing them to switch ciphertexts from high-degree cyclotomic rings to *subrings* of smaller degree (once enough homomorphic operations have been performed to make this secure). We generalize the procedure, showing how to switch between two rings where neither ring need be a subring of the other. The procedure has a very simple implementation, and as long as the two rings have a large *common subring*, it is also very efficient (i.e., quasilinear in the dimension). Moreover, it supports, as a side effect, the homomorphic evaluation of any function that is *linear over the common subring*. However, the larger the common subring is, the more restrictive this condition on the function becomes.

We show how our enhanced ring-switching can move the plaintext coefficients into the slots of the target ring (and back), which can be seen as just evaluating a certain $\mathbb{Z}$-linear function. Here we are faced with the main technical challenge: for efficiency, the common subring of the source and destination rings must be large, but then the supported class of linear functions is very restrictive, and certainly does not include the $\mathbb{Z}$-linear one we want to evaluate. We solve this problem by switching through a short sequence of "hybrid" rings, where adjacent rings have a large common subring, but the initial and final rings have only the integers $\mathbb{Z}$ in common. Moreover, we show that for an appropriately chosen sequence of hybrid rings, the $\mathbb{Z}$-linear function we want to evaluate *is* realizable by a sequence of allowed linear functions between adjacent hybrid rings. Very critically, this decomposition requires the SHE scheme to use a *highly structured*

basis of the ring for decryption. The usual representation of a cyclotomic ring as $\mathbb{Z}[X]/\Phi_m(X)$ typically does not correspond to such a basis, so we instead rely on the *tensorial decomposition* of the ring and its corresponding bases, as recently explored in [LPR13]. At heart, this is what allows us to avoid the expensive homomorphic reduction modulo $\Phi_m(X)$, which is one of the main bottlenecks in previous work [GHS12a].[2]

Stepping back a bit, the technique of switching through hybrid rings and bases is reminiscent of standard "sparse decompositions" for linear transformations like the FFT, in that both decompose a complicated high-dimensional transform into a short sequence of simpler, structured transforms. (Here, the simple transforms are computed merely as a side-effect of passing through the hybrid rings.) Because of these similarities, we believe that the enhanced ring-switching procedure will be applicable in other domain-specific applications of homomorphic encryption, e.g., signal-processing transforms or statistical analysis.

Organization. Section 2.1 recalls the extensive algebraic background required for our constructions, and Section 2.2 recalls a standard ring-based SHE scheme and some of its natural homomorphic operations. Section 3 defines the general bootstrapping procedure. Sections 4 and 5 respectively fill in the details of the two novel homomorphic operations used in the bootstrapping procedure. Appendix A documents a folklore transformation between two essentially equivalent ways of encoding messages in SHE schemes. Appendix B describes an integer rounding procedure that simplifies the one given in [GHS12a], and Appendix C gives some concrete choices of rings that our method can use in practice.

Acknowledgments. We thank Oded Regev for helpful discussions during the early stages of this research, and the anonymous CRYPTO'13 reviewers for their thoughtful comments.

2 Preliminaries

For a positive integer k, we let $[k] = \{0, \ldots, k-1\}$. For an integer modulus q, we let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo q. For integers q, q', we define the integer "rounding" function $\lfloor \cdot \rceil_{q'} \colon \mathbb{Z}_q \to \mathbb{Z}_{q'}$ as $\lfloor x \rceil_{q'} = \lfloor (q'/q) \cdot x \rceil \bmod q'$.

2.1 Algebraic Background

Throughout this work, by "ring" we mean a commutative ring with identity. For two rings $R \subseteq R'$, an R-basis of R' is a set $B \subset R'$ such that every $r \in R'$ can be written uniquely as an R-linear combination of elements of B. For two rings R, S with a common subring E, an E-linear function $L \colon R \to S$ is one for which $L(r + r') = L(r) + L(r')$ for all $r, r' \in R$, and $L(e \cdot r) = e \cdot L(r)$ for all $e \in E, r \in R$. It is immediate that such a function is defined uniquely by its values on any E-basis of R.

2.1.1 Cyclotomic Rings

For a positive integer m called the *index*, let $\mathcal{O}_m = \mathbb{Z}[\zeta_m]$ denote the mth *cyclotomic ring*, where ζ_m is an abstract element of order m over $\mathbb{Q}$. (In particular, we do not view ζ_m as any particular complex root of unity.) The minimal polynomial of ζ_m over $\mathbb{Q}$ is the mth *cyclotomic polynomial* $\Phi_m(X) = \prod_{i \in \mathbb{Z}_m^*} (X - \omega_m^i) \in \mathbb{Z}[X]$, where $\omega_m = \exp(2\pi\sqrt{-1}/m) \in \mathbb{C}$ is the principal mth complex root of unity, and the roots $\omega_m^i \in \mathbb{C}$ range

[2] The use of more structured representations of cyclotomic rings in [LPR13] was initially motivated by the desire for simpler and more efficient algorithms for cryptographic operations. Interestingly, these representations yield moderate efficiency improvements for computations "in the clear," but dramatic benefits for their homomorphic counterparts!

over all the *primitive* complex mth roots of unity. Therefore, $\mathcal{O}_m$ is a ring extension of degree $n = \varphi(m)$ over $\mathbb{Z}$. (In particular, $\mathcal{O}_1 = \mathcal{O}_2 = \mathbb{Z}$.) Clearly, $\mathcal{O}_m$ is isomorphic to the polynomial ring $\mathbb{Z}[X]/\Phi_m(X)$ by identifying ζ_m with X, and has the "power basis" $\{1, \zeta_m, \ldots, \zeta_m^{n-1}\}$ as a $\mathbb{Z}$-basis. However, for non-prime-power m the power basis can be somewhat cumbersome and inefficient to work with. In Section 2.1.4 we consider other, more structured bases that are essential to our techniques.

If $m|m'$, we can view the mth cyclotomic ring $\mathcal{O}_m$ as a subring of $\mathcal{O}_{m'} = \mathbb{Z}[\zeta_{m'}]$, via the ring embedding (i.e., injective ring homomorphism) that maps ζ_m to $\zeta_{m'}^{m'/m}$. The ring extension $\mathcal{O}_{m'}/\mathcal{O}_m$ has degree $d = \varphi(m')/\varphi(m)$, and also d automorphisms τ_i (i.e., automorphisms of $\mathcal{O}_{m'}$ that fix $\mathcal{O}_m$ pointwise), which are defined by $\tau_i(\zeta_{m'}) = \zeta_{m'}^i$ for each $i \in \mathbb{Z}_{m'}^*$ such that $i = 1 \pmod m$. The *trace* function $\mathrm{Tr} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} : \mathcal{O}_{m'} \to \mathcal{O}_m$ can be defined as the sum of these automorphisms:

$$\mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m}(a) = \sum_i \tau_i(a) \in \mathcal{O}_m.$$

Notice that Tr is $\mathcal{O}_m$-linear by definition. If $\mathcal{O}_{m''}/\mathcal{O}_{m'}/\mathcal{O}_m$ is a tower of ring extensions, then the trace satisfies the composition property $\mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_m} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} \circ \mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_{m'}}$.

An important element in the mth cyclotomic ring is

$$g := \prod_{\text{odd prime } p|m} (1 - \zeta_p) \in \mathcal{O}_m. \tag{2.1}$$

Also define $\hat{m} = m/2$ if m is even, otherwise $\hat{m} = m$, for any cyclotomic index m. It is known that $g|\hat{m}$ (see, e.g., [LPR13, Section 2.5.4]). The following lemma shows how the elements g in different cyclotomic rings, and the ideals they generate, are related by the trace function.

Lemma 2.1. *Let $m|m'$ be positive integers and let $g \in R = \mathcal{O}_m, g' \in R' = \mathcal{O}_{m'}$ and $\hat{m}, \hat{m}'$ be as defined above. Then $\mathrm{Tr}_{R'/R}(g'R') = (\hat{m}'/\hat{m}) \cdot gR$, and in particular, $\mathrm{Tr}_{R'/R}(g') = (\hat{m}'/\hat{m}) \cdot g$.*

Later on we use the *scaled* trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$, which by the above lemma maps the ideal $g'R$ to gR, and g' to g.

Proof. Let $\mathrm{Tr} = \mathrm{Tr}_{R'/R}$. To prove the first claim, we briefly recall certain properties of $R^\vee$, the fractional ideal "dual" to R; see [LPR13, Section 2.5.4] for further details. First, $R^\vee = (g/\hat{m})R$, and similarly $(R')^\vee = (g'/\hat{m}')R'$. It also follows directly from the definition of the dual ideal that $\mathrm{Tr}((R')^\vee) = R^\vee$; see for example [GHPS12, Equation 2.2]. Therefore, $\mathrm{Tr}(g'R') = (\hat{m}'/\hat{m}) \cdot gR$.

For the second claim, we first show the effect of the trace on g' when $m' = m \cdot p$ for some prime p. If p divides m, then $\hat{m}'/\hat{m} = m'/m = p$, the degree of R'/R is $\varphi(m')/\varphi(m) = p$, and $g' = g \in R$, so $\mathrm{Tr}(g') = \mathrm{Tr}(g) = p \cdot g$. Now suppose p does not divide m. If $p = 2$, then m is even and m' is odd, so $\hat{m}'/\hat{m} = (m'/2)/m = 1$, the degree of R'/R is 1, and $g' = g \in R$, so $\mathrm{Tr}(g') = g$. Otherwise p is odd, so $\hat{m}'/\hat{m} = m'/m = p$ and $g' = (1 - \zeta_p)g$. Therefore $\mathrm{Tr}(g') = \mathrm{Tr}(1 - \zeta_p) \cdot g = p \cdot g$, where the final equality follows from $\mathrm{Tr}(1) = p - 1$ and $\mathrm{Tr}(\zeta_p) = \zeta_p^1 + \zeta_p^2 + \cdots \zeta_p^{p-1} = -1$.

The general case follows from the composition property of the trace, by iteratively applying the above case to any cyclotomic tower $R^{(r)}/R^{(r-1)}/\cdots/R^{(0)}$, where $R^{(r)} = R'$ and $R^{(0)} = R$, and the ratio of the indices of $R^{(i)}, R^{(i-1)}$ is prime for every $i = 1, \ldots, r$. □

2.1.2 Tensorial Decomposition of Cyclotomics

An important fact from algebraic number theory, used centrally in this work (and in [LPR13]), is the *tensorial decomposition* of cyclotomic rings (and their bases) in terms of subrings. Let $\mathcal{O}_{m_1}, \mathcal{O}_{m_2}$ be cyclotomic rings. Then their largest common subring is $\mathcal{O}_{m_1} \cap \mathcal{O}_{m_2} = \mathcal{O}_g$ where $g = \gcd(m_1, m_2)$, and their smallest common extension ring, called the *compositum*, is $\mathcal{O}_{m_1} + \mathcal{O}_{m_2} = \mathcal{O}_l$ where $l = \mathrm{lcm}(m_1, m_2)$. When considered as extensions of $\mathcal{O}_g$, the ring $\mathcal{O}_l$ is isomorphic to the *ring tensor product* of $\mathcal{O}_{m_1}$ and $\mathcal{O}_{m_2}$, written as (sometimes suppressing $\mathcal{O}_g$ when it is clear from context)

$$\mathcal{O}_l/\mathcal{O}_g \cong (\mathcal{O}_{m_1}/\mathcal{O}_g) \otimes (\mathcal{O}_{m_2}/\mathcal{O}_g).$$

On the right, the ring tensor product is defined as the set of all $\mathcal{O}_g$-linear combinations of *pure tensors* $a_1 \otimes a_2$, with ring operations defined by $\mathcal{O}_g$-bilinearity:

$$\begin{aligned}(a_1 \otimes a_2) + (b_1 \otimes a_2) &= (a_1 + b_1) \otimes a_2,\\ (a_1 \otimes a_2) + (a_1 \otimes b_2) &= a_1 \otimes (a_2 + b_2),\\ c(a_1 \otimes a_2) &= (ca_1) \otimes a_2 = a_1 \otimes (ca_2)\end{aligned}$$

for any $c \in \mathcal{O}_g$, and the mixed-product property $(a_1 \otimes a_2) \cdot (b_1 \otimes b_2) = (a_1 b_1) \otimes (a_2 b_2)$. The isomorphism with $\mathcal{O}_l/\mathcal{O}_g$ then simply identifies $a_1 \otimes a_2$ with $a_1 \cdot a_2 \in \mathcal{O}_l$. Note that any $a_1 \in \mathcal{O}_{m_1}$ corresponds to the pure tensor $a_1 \otimes 1$, and similarly for any $a_2 \in \mathcal{O}_{m_2}$.

The following simple lemma will be central to our techniques.

Lemma 2.2. *Let m_1, m_2 be positive integers and $g = \gcd(m_1, m_2)$, $l = \mathrm{lcm}(m_1, m_2)$. Then for any $\mathcal{O}_g$-linear function $\bar{L}\colon \mathcal{O}_{m_1} \to \mathcal{O}_{m_2}$, there is an (efficiently computable) $\mathcal{O}_{m_2}$-linear function $L\colon \mathcal{O}_l \to \mathcal{O}_{m_2}$ that coincides with $\bar{L}$ on the subring $\mathcal{O}_{m_1} \subseteq \mathcal{O}_l$.*

Proof. Write $\mathcal{O}_l \cong \mathcal{O}_{m_1} \otimes \mathcal{O}_{m_2}$, where the common base ring $\mathcal{O}_g$ is implicit. Let $L\colon (\mathcal{O}_{m_1} \otimes \mathcal{O}_{m_2}) \to \mathcal{O}_{m_2}$ be the $\mathcal{O}_g$-linear function uniquely defined by $L(a_1 \otimes a_2) = \bar{L}(a_1) \cdot a_2 \in \mathcal{O}_{m_2}$ for all pure tensors $a_1 \otimes a_2$. Then because $(a_1 \otimes a_2) \cdot b_2 = a_1 \otimes (a_2 b_2)$ for any $b_2 \in \mathcal{O}_{m_2}$ by the mixed-product property, L is also $\mathcal{O}_{m_2}$-linear. Finally, for any $a_1 \in \mathcal{O}_{m_1}$ we have $L(a_1 \otimes 1) = \bar{L}(a_1)$ by construction. □

2.1.3 Ideal Factorization and Plaintext Slots

Here we recall the unique factorization of prime integers into prime ideals in cyclotomic rings, and, following [SV11], how the Chinese remainder theorem can yield several plaintext "slots" that embed $\mathbb{Z}_q$ as a subring, even for composite q. Similar facts for composite moduli are presented in [GHS12a], but in terms of p-adic approximations and Hensel lifting. Here we give an ideal-theoretic interpretation using the Chinese remainder theorem, which we believe is more elementary, and is a direct extension of the case of prime moduli.

Let $p \in \mathbb{Z}$ be a prime integer. In the mth cyclotomic ring $R = \mathcal{O}_m = \mathbb{Z}[\zeta_m]$ (which has degree $n = \varphi(m)$ over $\mathbb{Z}$), the ideal pR factors into prime ideals as follows. First write $m = \bar{m} \cdot p^k$ where $p \nmid \bar{m}$. Let $e = \varphi(p^k)$, and let d be the multiplicative order of p modulo in $\mathbb{Z}^*_{\bar{m}}$, and note that d divides $\varphi(\bar{m}) = n/e$. The ideal pR then factors into the product of eth powers of $\varphi(\bar{m})/d = n/(de)$ distinct prime ideals $\mathfrak{p}_i$, i.e.,

$$pR = \prod \mathfrak{p}_i^e.$$

Each prime ideal $\mathfrak{p}_i$ has norm $|R/\mathfrak{p}_i| = p^d$, so each quotient ring $R/\mathfrak{p}_i$ is isomorphic to the finite field $\mathbb{F}_{p^d}$. In particular, it embeds $\mathbb{Z}_p$ as a subfield. (Although we will not need this, the prime ideals are concretely given by $\mathfrak{p}_i = pR + F_i(\zeta_m)R$, where $\Phi_{\bar{m}}(X) = \prod_i F_i(X) \pmod p$ is the mod-p factorization of the $\bar{m}$th cyclotomic polynomial into $\varphi(\bar{m})/d$ distinct irreducible polynomials of degree d.)

We now see how to obtain quotient rings of R that embed the ring $\mathbb{Z}_q$, where $q = p^r$ for some integer $r \geq 1$. (The case of arbitrary integer modulus q follows immediately from the Chinese remainder theorem.) Here we have the factorization $qR = \prod_i \mathfrak{p}_i^{re}$, and it turns out that each quotient ring $R/\mathfrak{p}_i^{re}$ embeds $\mathbb{Z}_q$ as a subring. One easy way to see this is to notice that q is the smallest power of p in $\mathfrak{p}_i^{re}$, so the integers $\{0, \ldots, q-1\}$ representing $\mathbb{Z}_q$ are distinct modulo $\mathfrak{p}_i^{re}$.

By the Chinese Remainder Theorem (CRT), for $q = p^r$ the natural ring homomorphism from R_q to the product ring $\bigoplus_i (R/\mathfrak{p}_i^{re})$ is an isomorphism. When the natural plaintext space of a cryptosystem is R_q, we refer to the $\varphi(\bar{m})/d$ quotient rings $R/\mathfrak{p}_i^{re}$ as the plaintext "$\mathbb{Z}_q$-slots" (or just "slots"), and use them to store vectors of $\mathbb{Z}_q$-elements via the CRT isomorphism. With this encoding, ring operations in R_q induce "batch" (or "SIMD") component-wise operations on the corresponding vectors of $\mathbb{Z}_q$ elements. We note that the CRT isomorphism is easy to compute in both directions. In particular, to map from a vector of $\mathbb{Z}_q$-elements to R_q just requires knowing a fixed mod-q CRT set $C = \{c_i\} \subset R$ for which $c_i = 1 \pmod{\mathfrak{p}_i^{re}}$ and $c_i = 0 \pmod{\mathfrak{p}_j^{re}}$ for all $j \neq i$. Such a set can be precomputed using, e.g., a generalization of the extended Euclidean algorithm.

Splitting in cyclotomic extension rings. Now consider a cyclotomic extension R'/R where $R' = \mathcal{O}_{m'} = \mathbb{Z}[\zeta_{m'}]$ for some m' divisible by m. Then for each prime ideal $\mathfrak{p}_i \subset R$ dividing pR, the ideal $\mathfrak{p}_i R'$ factors into equal powers of the same number of prime ideals $\mathfrak{p}'_{i,i'} \subset R'$, where all the $\mathfrak{p}'_{i,i'}$ are distinct. The ideal $\mathfrak{p}'_{i,i'}$ is said to "lie over" $\mathfrak{p}_i$ (and $\mathfrak{p}_i$ in turns lies over p). Since $\mathfrak{p}'_{i,i'}$ are also the prime ideals appearing in the factorization pR', we can determine their number and multiplicity exactly as above. Letting $\bar{m}'$, e' and d' be defined as above for R', we known that $pR' = \prod_{i,i'} (\mathfrak{p}'_{i,i'})^{e'}$, where there are a total of $\varphi(\bar{m}')/d'$ distinct prime ideals $\mathfrak{p}'_{i,i'}$. Therefore, each $\mathfrak{p}_i$ splits into exactly $(\varphi(\bar{m}') \cdot d)/(\varphi(\bar{m}) \cdot d')$ ideals each; this number is sometimes called the "relative splitting number" of p in R'/R.

2.1.4 Product Bases

Our bootstrapping technique relies crucially on certain highly structured bases and CRT sets, which we call "product bases (sets)," that arise from towers of cyclotomic rings. Let $\mathcal{O}_{m''}/\mathcal{O}_{m'}/\mathcal{O}_m$ be such a tower, let $B'' = \{b''_{j''}\} \subset \mathcal{O}_{m''}$ be any $\mathcal{O}_{m'}$-basis of $\mathcal{O}_{m''}$, and let $B' = \{b'_{j'}\} \subset \mathcal{O}_{m'}$ be any $\mathcal{O}_m$-basis of $\mathcal{O}_{m'}$. Then it follows immediately that the product set $B'' \cdot B' := \{b''_{j''} \cdot b'_{j'}\} \subset \mathcal{O}_{m''}$ is an $\mathcal{O}_m$-basis of $\mathcal{O}_{m''}$.[3] Of course, for a tower of several cyclotomic extensions and relative bases, we can obtain product bases that factor with a corresponding degree of granularity.

Factorization of the powerful and decoding bases. An important structured $\mathbb{Z}$-basis of $\mathcal{O}_m$, called the "powerful" basis in [LPR13], was defined in that work as the product of all the power $\mathbb{Z}$-bases $\{\zeta^0, \zeta^1, \ldots, \zeta^{\varphi(p^e)-1}\}$ of $\mathcal{O}_{p^e}$ (where $\zeta = \zeta_{p^e}$), taken over all the maximal prime-power divisors p^e of m. In turn, it is straightforward to verify that the power $\mathbb{Z}$-basis of $\mathcal{O}_{p^e}$ can be obtained from the tower $\mathcal{O}_{p^e}/\mathcal{O}_{p^{e-1}}/\cdots/\mathbb{Z}$, as the product of all the power $\mathcal{O}_{p^{i-1}}$-bases $\{\zeta_{p^i}^0, \ldots, \zeta_{p^i}^{d_i-1}\}$ of $\mathcal{O}_{p^i}$ for $i = 1, \ldots, e$, where $d_i = \varphi(p^i)/\varphi(p^{i-1}) \in \{p-1, p\}$ is the degree of $\mathcal{O}_{p^i}/\mathcal{O}_{p^{i-1}}$. Therefore, the powerful basis has a

[3] Formally, this basis is a *Kronecker* product of the bases B'' and B', which is typically written using the $\otimes$ operator. We instead use $\cdot$ to avoid confusion with pure tensors in a ring tensor product, which the elements of $B'' \cdot B'$ may not necessarily be.

"finest possible" product structure. (This is not the case for other commonly used bases of $\mathcal{O}_m$, such as the power $\mathbb{Z}$-basis, unless m is a prime power.)

Similarly, [LPR13] defines the "decoding" $\mathbb{Z}$-basis D of a certain fractional ideal $\mathcal{O}_m^\vee = (g/\hat{m})\mathcal{O}_m$, which is the "dual ideal" of $\mathcal{O}_m$, to be the dual basis of the conjugate powerful basis. Unlike the powerful basis, the decoding basis has optimal noise tolerance (see [LPR13, Section 6.2]) and is therefore a best choice to use in decryption, when using the dual ideal $\mathcal{O}_m^\vee$ appropriately in a cryptosystem. For simplicity, our formulation of the cryptosystem (see Section 2.2) avoids using $\mathcal{O}_m^\vee$ by "scaling up" to $(\hat{m}/g)\mathcal{O}_m^\vee = \mathcal{O}_m$, and so we are interested in factorizations of the scaled-up $\mathbb{Z}$-basis $(\hat{m}/g)D$ of $\mathcal{O}_m$. As shown in [LPR13, Lemma 6.3], this basis is very closely related to the powerful basis, and has a nearly identical product structure arising from the towers $\mathcal{O}_{p^e}/\mathcal{O}_{p^{e-1}}/\cdots/\mathbb{Z}$ for the maximal prime-power divisors p^e of m. The only difference is in the choice of the lowest-level $\mathbb{Z}$-bases of each $\mathcal{O}_p/\mathbb{Z}$, which are taken to be $\{\zeta_p^j + \zeta_p^{j+1} + \cdots + \zeta_p^{p-2}\}_{j\in\{0,\ldots,p-2\}}$ instead of the power basis. In summary, the preferred $\mathbb{Z}$-basis of $\mathcal{O}_m$ used for decryption also has a finest-possible product structure.

Factorization of CRT sets. Using the splitting behavior of primes and prime ideals, we can also define CRT sets having a finest-possible product structure. First consider any cyclotomic extension $\mathcal{O}_{m'}/\mathcal{O}_m$, and suppose that prime integer p splits in $\mathcal{O}_m$ into distinct prime ideals $\mathfrak{p}_i$. In turn, each $\mathfrak{p}_i$ splits in $\mathcal{O}_{m'}$ into the same number k of prime ideals $\mathfrak{p}'_{i,i'}$, which are all distinct. For simplicity, assume for now that p does not divide m or m', so none of the ideals occur with multiplicity.

A mod-p CRT set $C = \{c_i\}$ for $\mathcal{O}_m$ satisfies $c_i = 1 \pmod{\mathfrak{p}_i}$ and $c_i = 0 \pmod{\mathfrak{p}_j}$ for $j \neq i$; therefore, $c_i = 1 \pmod{\mathfrak{p}'_{i,i'}}$ and $c_i = 0 \pmod{\mathfrak{p}'_{j,i'}}$ for all i' and all $j \neq i$. We can choose a set $S = \{s_{i'}\} \subset \mathcal{O}_{m'}$ of size k such that $C' = S \cdot C$ is a mod-p CRT set for $\mathcal{O}_{m'}$, as follows: partition the ideals $\mathfrak{p}'_{i,i'}$ arbitrarily according to i', and define $s_{i'} \in \mathcal{O}_{m'}$ to be congruent to 1 modulo all those ideals $\mathfrak{p}'_{i,i'}$ in the i'th component of the partition, and 0 modulo all the other ideals $\mathfrak{p}'_{j,j'}$. Then it is immediate that each product $c_i \cdot s_{i'}$ is 1 modulo $\mathfrak{p}'_{i,i'}$, and 0 modulo all other $\mathfrak{p}'_{j,j'}$. Therefore, $C' = S \cdot C$ is a mod-p CRT set for $\mathcal{O}_{m'}$. The generalization of this process to the case where p factors into *powers* of the ideals, and to moduli $q = p^r$, is immediate.

For an arbitrary cyclotomic index m, consider any cyclotomic tower $\mathcal{O}_m/\cdots/\mathbb{Z}$. Then a mod-$q$ CRT set with corresponding product structure can be obtained by iteratively applying the above procedure at each level of the tower. A finest-possible product structure is obtained by using tower of maximal length (i.e., one in which the ratio of indices at adjacent levels is always prime).

2.2 Ring-Based Homomorphic Cryptosystem

Here we recall a somewhat-homomorphic encryption scheme whose security is based on the ring-LWE problem [LPR10] in arbitrary cyclotomic rings. For our purposes we focus mainly on its decryption function, though below we also recall its support for "ring switching" [GHPS12]. For further details on its security guarantees, various homomorphic properties, and efficient implementation, see [LPR10, BV11b, BGV12, GHS12c, GHPS12, LPR13].

Let $R = \mathcal{O}_m \subseteq R' = \mathcal{O}_{m'}$ be respectively the mth and m'th cyclotomic rings, where $m|m'$. The plaintext ring is the quotient ring R_p for some integer p; ciphertexts are made up of elements of R'_q for some integer q, which for simplicity we assume is divisible by p; and the secret key is some $s \in R'$. The case $m = 1$ corresponds to "non-packed" ciphertexts, which encrypt elements of $\mathbb{Z}_p$ (e.g., single bits), whereas $m = m'$ corresponds to "packed" ciphertexts, and $1 < m < m'$ corresponds to what we call "semi-packed" ciphertexts. Note that without loss of generality we can treat any ciphertext as packed, since R'_p embeds R_p.

But the smaller m is, the simpler and more practically efficient our bootstrapping procedure can be. Since our focus is on refreshing ciphertexts that have large noise rate, we can think of m' as being somewhat small (e.g., in the several hundreds) via ring-switching [GHPS12], and q also as being somewhat small (e.g., in the several thousands) via modulus-switching. Our main focus in this work is on a plaintext modulus p that is a power of two, though for generality we present all our techniques in terms of arbitrary p.

A ciphertext encrypting a message $\mu \in R_p$ under secret key $s' \in R'$ is some pair $c' = (c'_0, c'_1) \in R'_q \times R'_q$ satisfying the relation

$$c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'} \tag{2.2}$$

for some error (or "noise") term $e' \in R'$ such that $e' \cdot g' \in g'R'$ is sufficiently "short," where $g' \in R'$ is as defined in Equation (2.1).[4] Informally, the "noise rate" of the ciphertext is the ratio of the "size" of e' (or more precisely, the magnitude of its coefficients in a suitable basis) to q/p.

We note that Equation (2.2) corresponds to what is sometimes called the "most significant bit" (msb) message encoding, whereas somewhat-homomorphic schemes are often defined using "least significant bit" (lsb) encoding, in which p and q are coprime and $c'_0 + c'_1 s' = e' \pmod{qR'}$ for some error term $e' \in \mu + pR'$. For our purposes the msb encoding is more natural, and in any case the two encodings are essentially equivalent: when p and q are coprime, we can trivially switch between the two encodings simply by multiplying by p or p^{-1} modulo q (see Appendix A). When p divides q, we can use homomorphic operations for the msb encoding due to Brakerski [Bra12]; alternatively, we can switch to and from a different modulus q' that is coprime with p, allowing us to switch between lsb and msb encodings as just described. In practice, it may be preferable to use homomorphic operations for the lsb encoding, because they admit optimizations (e.g., the "double-CRT representation" [GHS12c]) that may not be possible for the msb operations (at least when p divides q).

2.2.1 Decryption

At a high level, the decryption algorithm works in two steps: the "linear" step simply computes $v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \in R'_q$, and the "non-linear" step outputs $\lfloor v' \rceil_p \in R_p$ using a certain "ring rounding function" $\lfloor \cdot \rceil_p \colon R'_q \to R_p$. As long as the error term e' is within the tolerance of the rounding function, the output will be $\mu \in R_p$. This is all entirely analogous to decryption in LWE-based systems, but here the rounding is n-dimensional, rather than just from $\mathbb{Z}_q$ to $\mathbb{Z}_p$.

Concretely, the ring rounding function $\lfloor \cdot \rceil_p \colon R'_q \to R_p$ is defined in terms of the integer rounding function $\lfloor \cdot \rceil_p \colon \mathbb{Z}_q \to \mathbb{Z}_p$ and a certain "decryption" $\mathbb{Z}$-basis $B' = \{b_j\}$ of R', as follows.[5] Represent the input $v' \in R'_q$ in the decryption basis as $v' = \sum_j v'_j \cdot b'_j$ for some coefficients $v'_j \in \mathbb{Z}_q$, then independently round the coefficients, yielding an element $\sum \lfloor v'_j \rceil_p \cdot b'_j \in R'_p$ that corresponds to the message $\mu \in R_p$ (under the standard embedding of R_p into R'_p).

[4] Quantitatively, "short" is defined with respect to the *canonical embedding* of R', whose precise definition is not needed in this work. The above system is equivalent to the one from [LPR13] in which the message, error term, and ciphertext components are all taken over the "dual" fractional ideal $(R')^\vee = (g'/\hat{m}')R'$ in the m'th cyclotomic number field, and the error term has an essentially spherical distribution over $(R')^\vee$. In that system, decryption is best accomplished using a certain $\mathbb{Z}$-basis of $(R')^\vee$, called the *decoding basis*, which optimally decodes spherical errors. The above formulation is more convenient for our purposes, and simply corresponds with multiplying everything in the system of [LPR13] by an $\hat{m}'/g'$ factor. This makes $e' \cdot g' \in g'R' = \hat{m}'(R')^\vee)$ short and essentially spherical in our formulation. See [LPR10, LPR13] for further details.

[5] In our formulation, the basis B' is $(\hat{m}'/g')$ times the decoding basis of $(R')^\vee$. See Section 2.1.4 and Footnote 4.

2.2.2 Changing the Plaintext Modulus

We use two operations on ciphertexts that alter the plaintext modulus p and encrypted message $\mu \in R_p$. The first operation changes p to any multiple $p' = dp$, and produces an encryption of some $\mu' \in R'_{p'}$ such that $\mu' = \mu \pmod{pR'}$. To do this, it simply "lifts" the input ciphertext $c' = (c'_0, c'_1) \in (R'_q)^2$ to an arbitrary $c'' = (c''_0, c''_1) \in (R'_{q'})^2$ such that $c''_j = c'_j \pmod{qR'}$, where $q' = dq$. This works because

$$c''_0 + c''_1 \cdot s' \in c'_0 + c'_1 \cdot s' + qR' = \left(\frac{q}{p} \cdot \mu + e'\right) + qR' = \frac{q'}{p'}(\mu + pR') + e' \pmod{q'R'}.$$

Notice that this leaves the noise rate unchanged, because the noise term is still e', and $q'/p' = q/p$.

The second operation applies to an encryption of a message $\mu \in R_p$ that is known to be divisible by some divisor d of p, and produces an encryption of $\mu/d \in R_{p/d}$. The operation actually leaves the ciphertext c' unchanged; it just declares the associated plaintext modulus to be p/d (which affects how decryption is performed). This works because

$$c'_0 + c'_1 \cdot s' = \frac{q}{p}\mu + e' = \frac{q}{p/d} \cdot (\mu/d) + e' \pmod{qR'}.$$

Notice that the noise rate of the ciphertext has been divided by d, because the noise term is still e' but $q/p' = d(q/p)$.

2.2.3 Ring Switching

We rely heavily on the cryptosystem's support for switching ciphertexts to a cyclotomic subring S' of R', which as a side-effect homomorphically evaluates any desired S'-linear function on the plaintext. Notice that the linear function L is applied to the plaintext as embedded in R'_p; this obviously applies the induced function on the true plaintext space R_p.

Proposition 2.3 ([GHPS12], full version). *Let $S' \subseteq R'$ be cyclotomic rings. Then the above-described cryptosystem supports the following homomorphic operation: given any S'-linear function $L\colon R'_p \to S'_p$ and a ciphertext over R'_q encrypting (with sufficiently small error term) a message $\mu \in R'_p$, the output is a ciphertext over S'_q encrypting $L(\mu) \in S'_p$.*

The security of the procedure described in Proposition 2.3 is based on the hardness of the ring-LWE problem in S', so the dimension of S' must be sufficiently large. The procedure itself is quite simple and efficient: it first switches to a secret key that lies in the subring S', then it multiplies the resulting ciphertext by an appropriate fixed element of R' (which is determined solely by the function L). Finally, it applies to the ciphertext the trace function $\mathrm{Tr}_{R'/S'}\colon R' \to S'$. All of these operations are quasi-linear time in the dimension of $R'/\mathbb{Z}$, and very efficient in practice. In particular, the trace is a trivial linear-time operation when elements are represented in any of the bases we use. The ring-switching procedure increases the effective error rate of the ciphertext by a factor of about the square root of the dimension of R', which is comparable to that of a single homomorphic multiplication. See [GHPS12] for further details.

3 Overview of Bootstrapping Procedure

Here we give a high-level description of our bootstrapping procedure. We present a unified procedure for non-packed, packed, and semi-packed ciphertexts, but note that for non-packed ciphertexts, Steps 3a and 3c (and possibly 1c) are null operations, while for packed ciphertexts, Steps 1b, 1c, and 2 are null operations.

Recalling the cryptosystem from Section 2.2, the plaintext ring is R_p and the ciphertext ring is R'_q, where $R = \mathcal{O}_m \subseteq R' = \mathcal{O}_{m'}$ are cyclotomic rings (so $m|m'$), and q is a power of p. The procedure also uses a larger cyclotomic ring $R'' = \mathcal{O}_{m''} \supseteq R'$ (so $m'|m''$) to work with ciphertexts that encrypt elements of the original ciphertext ring R'_q. To obtain quasilinear runtimes and exponential hardness (under standard hardness assumptions), our procedure imposes some mild conditions on the indices m, m', and m'':

- The dimension $\varphi(m'')$ of R'' must be quasilinear, so we can represent elements of R'' efficiently.
- For Steps 2 and 3, all the prime divisors of m and m' must be small (i.e., polylogarithmic).
- For Step 3, m and m''/m must be coprime, which implies that m and m'/m must be coprime also. Note that the former condition is always satisfied for non-packed ciphertexts (where $m = 1$). For packed ciphertexts (where $m = m'$), the latter condition is always satisfied, which makes it easy to choose a valid m''. For semi-packed ciphertexts (where $1 < m < m'$), we can always satisfy the latter condition either by increasing m (at a small expense in practical efficiency in Step 3; see Section 5.1.3), or by effectively decreasing m slightly (at a possible improvement in practical efficiency; see Section 3.2).

For example, when $m = 1$, both m' and m'' can be powers of two.

The input to the procedure is a ciphertext $c' = (c'_0, c'_1) \in (R'_q)^2$ that encrypts some plaintext $\mu \in R_p$ under a secret key $s' \in R'$, i.e., it satisfies the relation

$$v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'}$$

for some small enough error term $e' \in R'$. The procedure computes a new encryption of $\lfloor v' \rceil_p = \mu$ (under some secret key, not necessarily s') that has substantially smaller noise rate than the input ciphertext. It proceeds as follows (explanatory remarks appear in italics):

1. Convert c' to a "noiseless" ciphertext c'' over a large ring R''_Q that encrypts a plaintext $(g'/g)u' \in R'_{q'}$, where $g' \in R', g \in R$ and $\hat{m}, \hat{m}' \in \mathbb{Z}$ are as defined in (and following) Equation (2.1), $q' = (\hat{m}'/\hat{m})q$, and $u' = v' \pmod{qR'}$. This proceeds in the following sub-steps (see Section 3.1 for further details).

   Note that $g'/g \in R'$ by definition, and that it divides $\hat{m}'/\hat{m}$.

   (a) Reinterpret c' as a noiseless encryption of $v' = \frac{q}{p} \cdot \mu + e' \in R'_q$ as a *plaintext*, noting that both the plaintext and ciphertext rings are now taken to be R'_q.
   This is purely a conceptual change in perspective, and does not involve any computation.

   (b) Using the procedure described in Section 2.2.2, change the plaintext (and ciphertext) modulus to $q' = (\hat{m}'/\hat{m})q$, yielding a noiseless encryption of some $u' \in R'_{q'}$ such that $u' = v' \pmod{qR'}$.
   Note that this step is a null operation if the original ciphertext was packed, i.e., if $m = m'$.
   We need to increase the plaintext modulus because homomorphically computing $\mathrm{Tr}_{R'/R}$ in Step 2 below introduces an $\hat{m}'/\hat{m}$ factor into the plaintext, which we will undo by scaling the plaintext modulus back down to q. (See Section 3.2 for an alternative choice of q'.)

   (c) Multiply the ciphertext from the previous step by $g'/g \in R'$, yielding a noiseless encryption of plaintext $(g'/g)u' \in R'_{q'}$.
   The factor $(g'/g) \in R'$ is needed when we homomorphically compute $\mathrm{Tr}_{R'/R}$ in Step 2 below. Note that $g'/g = 1$ if and only if every odd prime divisor of m' also divides m, e.g., if $m = m'$.

(d) Convert to a noiseless ciphertext c'' that still encrypts $(g'/g)u' \in R'_{q'}$, but using a large enough ciphertext ring R''_Q for some $R'' = \mathcal{O}_{m''} \supseteq R'$ and modulus $Q \gg q'$.

A larger ciphertext ring R''_Q is needed for security in the upcoming homomorphic operations, to compensate for the low noise rates that will need to be used. These operations will expand the initial noise rate by a quasipolynomial $\lambda^{O(\log \lambda)}$ factor in total, so the dimension of R'' and the bit length of Q can be $\tilde{O}(\lambda)$ and $\tilde{O}(1)$, respectively.

The remaining steps are described here only in terms of their effect on the *plaintext* value and ring. Using ring- and modulus-switching, the ciphertext ring R'' and modulus Q may be made smaller as is convenient, subject to the security and functionality requirements. (Also, the ciphertext ring implicitly changes during Steps 3a and 3c.)

2. Homomorphically apply the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ to the encryption of $(g'/g)u' \in R'_{q'}$, to obtain an encryption of plaintext

$$u = \frac{\hat{m}}{\hat{m}'} \cdot \mathrm{Tr}_{R'/R}\Big(\frac{g'}{g} \cdot u'\Big) = \frac{q}{p} \cdot \mu + e \in R_q$$

for some suitably small error term $e \in R$. See Section 4 for further details.

This step changes the plaintext ring from $R'_{q'}$ to R_q, and homomorphically isolates the noisy R_q-encoding of μ. It is a null operation if the original ciphertext was packed, i.e., if $m = m'$.

3. Homomorphically apply the ring rounding function $\lfloor \cdot \rceil_p \colon R_q \to R_p$, yielding an output ciphertext that encrypts $\lfloor u \rceil_p = \mu \in R_p$. This proceeds in three sub-steps, all of which are applied homomorphically (see Section 5 for details):

(a) Map the coefficients u_j of $u \in R_q$ (with respect to the decryption basis B of R) to the $\mathbb{Z}_q$-slots of a ring S_q, where S is a suitably chosen cyclotomic.

This step changes the plaintext ring from R_q to S_q. It is a null operation if the original ciphertext was non-packed (i.e., if $m = 1$), because we can let $S = R = \mathbb{Z}$.

(b) Batch-apply the integer rounding function $\lfloor \cdot \rceil \colon \mathbb{Z}_q \to \mathbb{Z}_p$ to the $\mathbb{Z}_q$-slots of S_q, yielding a ciphertext that encrypts the values $\mu_j = \lfloor u_j \rceil_p \in \mathbb{Z}_p$ in its $\mathbb{Z}_p$-slots.

This step changes the plaintext ring from S_q to S_p. It constitutes the only non-linear operation on the plaintext, with multiplicative depth $\lceil \lg p \rceil \cdot (\log_p(q) - 1) \approx \log(q)$, and as such is the most expensive in terms of runtime, noise expansion, etc.

(c) Reverse the map from the step 3a, sending the values μ_j from the $\mathbb{Z}_p$-slots of S_p to coefficients with respect to the decryption basis B of R_p, yielding an encryption of $\mu = \sum_j \mu_j b_j \in R_p$.

This step changes the plaintext ring from S_p to R_p. Just like step 3a, it is a null operation for non-packed ciphertexts.

3.1 Obtaining a Noiseless Ciphertext

Step 1 of our bootstrapping procedure is given as input a ciphertext $c' = (c'_0, c'_1)$ over R'_q that encrypts (typically with a high noise rate) a message $\mu \in R_p$ under key $s' \in R'$, i.e., $v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \in R'_q$ for some error term e'. We first change our perspective and view c' as a "noiseless" encryption (still under s')

of the plaintext value $v' \in R'_q$, taking both the plaintext and ciphertext rings to be R'_q. This view is indeed formally correct, because

$$c'_0 + c'_1 \cdot s' = \frac{q}{q} \cdot v' + 0 \pmod{qR'}.$$

Next, in preparation for the upcoming homomorphic operations we increase the plaintext (and ciphertext) modulus to q', and multiply the resulting ciphertext by g'/g. These operations clearly preserve noiselessness. Finally, we convert the ciphertext ring to R''_Q for a sufficiently large cyclotomic $R'' \supseteq R'$ and modulus $Q \gg q$ that is divisible by q. This is done by simply embedding R' into R'' and introducing extra precision, i.e., scaling the ciphertext up by a Q/q factor. It is easy to verify that these operations also preserve noiselessness.

3.2 Variants and Optimizations

Our basic procedure admits a few minor variants and practical optimizations, which we discuss here.

Smaller temporary modulus q'. In Step 1b we increase the plaintext modulus from q to $q' = rq$ where $r = \hat{m}'/\hat{m}$, and at the end of Step 2 we reduce the modulus back to q because the plaintext is divisible by r. The net effect of this, versus using a modulus q throughout, is that the modulus Q is larger by an r factor, as are the error rates used for key-switching in Step 2. This does not affect the asymptotic cost of bootstrapping, but it may have a small impact in practice. Instead, we can increase the modulus to only $q' = (r/d)q$, where d is the largest divisor of r coprime with q. Then in Step 2 we can remove an (r/d) factor from the plaintext by scaling the modulus back down to q, and keep track of the remaining d factor and remove it upon decryption. (We could also remove the d factor by multiplying the ciphertext by $d^{-1} \bmod q$, but this would increase the noise rate by up to a $q/2$ factor, which is typically much larger than the $\hat{m}'/\hat{m}$ factor we were trying to avoid in the first place.)

Using a smaller index m in Steps 2 and 3. Steps 3a and 3c can be much more costly in practice than Step 2, because they require working with rings that have at least $\varphi(m)$ $\mathbb{Z}_q$-slots. As the number of needed slots increases, the indices of such rings tend to grow quickly, and involve more prime divisors of larger size (though asymptotically the indices remain quasilinear); see Appendix C for some examples. So, in practice it may be faster to invoke Step 3 *a few times* to evaluate the rounding function over a *smaller* ring $\tilde{R} = \mathcal{O}_{\tilde{m}} \subset R$, for some proper divisor $\tilde{m}$ of m. Our procedure can be adapted to work in this way, even if the original plaintext μ is an arbitrary element of the plaintext space R_p.

The main facts we use are that the decryption basis B of R factors as $B = B' \cdot \tilde{B}$, where $\tilde{B}$ is the decryption basis of $\tilde{R}$, and in particular B' is an optimally short $\tilde{R}$-basis of R. (See Section 2.1.4.) Moreover, applying the ring rounding function on any $u \in R_q$ is equivalent to independently applying the ring rounding function on each of u's $\tilde{R}_q$-coefficients with respect to B'. Lastly, the $\tilde{R}_q$-coefficients of u can be individually extracted using the trace function $\mathrm{Tr}_{R/\tilde{R}}$ on certain fixed (short) multiples of u. (This all just generalizes the case $\tilde{R} = \mathbb{Z}$ in the natural way.) Using these facts, in Step 2 we can homomorphically apply $\mathrm{Tr}_{R/\tilde{R}}$ several times to obtain encryptions of the $\tilde{R}_q$-coefficients of the noisy encoding $u \approx (q/p) \cdot \mu$, then use Step 3 to homomorphically round those coefficients to get the $\tilde{R}_p$-coefficients of $\mu \in R_p$, and finally reassemble the pieces by homomorphically multiplying by the short basis elements in B', and summing the results.

Note that the above method requires evaluating $\mathrm{Tr}_{R/\tilde{R}}$ a total of $\varphi(m)/\varphi(\tilde{m})$ times in Step 2, and the same goes for the $\tilde{R}_q$ rounding function in Step 3. Because each evaluation takes quasilinear time no matter what $\tilde{m}$ is, the asymptotic performance can only worsen as $\tilde{m}$ decreases. However, in practice there may be benefits in choosing $\tilde{m}$ to be slightly smaller than m.

4 Homomorphic Trace

Here we show how to perform Step 2 of our bootstrapping procedure, which homomorphically evaluates the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ on an encryption of $(g'/g)u' \in R'_{q'}$, where recall that: $g' \in R', g \in R$ are as defined in Equation (2.1), and (g'/g) divides $(\hat{m}'/\hat{m})$; the plaintext modulus is $q' = (\hat{m}'/\hat{m})q$; and

$$u' = v' = \frac{q}{p} \cdot \mu + e' \pmod{qR'},$$

where $e' \cdot g' \in g'R'$ is sufficiently short. Our goal is to show that:

1. the scaled trace of the plaintext $(g'/g)u'$ is some $u = \frac{q}{p} \cdot \mu + e \in R_q$, where $e \cdot g \in gR$ is short, and
2. we can efficiently homomorphically apply the scaled trace on a ciphertext c'' over some larger ring $R'' = \mathcal{O}_{m''} \supseteq R'$.

4.1 Trace of the Plaintext

We first show the effect of the scaled trace on the plaintext $(g'/g)u' \in R'_{q'}$. By the above description of $u' \in R'_{q'}$ and the fact that $(g'/g)q$ divides $q' = (\hat{m}'/\hat{m})q$, we have

$$(g'/g)u' = (g'/g)v' = (g'/g)\left(\frac{q}{p} \cdot \mu + e'\right) \pmod{(g'/g)qR'}.$$

Therefore, letting $\mathrm{Tr} = \mathrm{Tr}_{R'/R}$, by R-linearity of the trace and Lemma 2.1, we have

$$\begin{aligned}\mathrm{Tr}((g'/g)u') &= \mathrm{Tr}(g'/g) \cdot \frac{q}{p} \cdot \mu + \mathrm{Tr}(e' \cdot g')/g \\ &= \frac{\hat{m}'}{\hat{m}}\left(\frac{q}{p} \cdot \mu + e\right) \pmod{q'R},\end{aligned}$$

where $e = (\hat{m}/\hat{m}')\,\mathrm{Tr}(e' \cdot g')/g \in R$. Therefore, after scaling down the plaintext modulus q' by an $\hat{m}'/\hat{m}$ factor (see Section 2.2.2), the plaintext is $\frac{q}{p} \cdot \mu + e \in R_q$.

Moreover, $e \cdot g = (\hat{m}/\hat{m}')\,\mathrm{Tr}(e' \cdot g') \in gR$ is short because $e' \cdot g' \in g'R'$ is short; see, e.g., [GHPS12, Corollary 2.2]. In fact, by basic properties of the decoding/decryption basis (as defined in [LPR13]) under the trace, the coefficient vector of e with respect to the decryption basis of R is merely a subvector of the coefficient vector of e' with respect to the decryption basis of R'. Therefore, e is within the error tolerance of the rounding function on R_q, assuming e' is within the error tolerance of the rounding function on R'_q.

4.2 Applying the Trace

Now we show how to efficiently homomorphically apply the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ to an encryption of any plaintext in $R'_{q'}$ that is divisible by (g'/g). Note that this condition ensures that the output of the trace is a multiple of $\hat{m}/\hat{m}'$ in $R_{q'}$ (see Lemma 2.1), making the scaling a well-defined operation that results in an element of R_q.

First recall that $\mathrm{Tr}_{R'/R}$ is the sum of all $\varphi(m')/\varphi(m)$ automorphisms of R'/R, i.e., automorphisms of R' that fix R pointwise. Therefore, one way of homomorphically computing the scaled trace is to homomorphically apply the proper automorphisms, sum the results, and scale down the plaintext and its modulus. While this "sum-automorphisms" procedure yields the correct result, computing the trace in this way does not run in quasilinear time, unless the number $\varphi(m')/\varphi(m)$ of automorphisms is only polylogarithmic.

Instead, we consider a sufficiently fine-grained tower of cyclotomic rings

$$R^{(r)}/\cdots/R^{(1)}/R^{(0)},$$

where $R' = R^{(r)}$, $R = R^{(0)}$, and each $R^{(i)} = \mathcal{O}_{m_i}$, where m_i is divisible by m_{i-1} for $i > 0$; for the finest granularity we would choose the tower so that every m_i/m_{i-1} is prime. Notice that the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ is the composition of the scaled trace functions $(\hat{m}_{i-1}/\hat{m}_i)\,\mathrm{Tr}_{R^{(i)}/R^{(i-1)}}$, and that g'/g is the product of all $g^{(i)}/g^{(i-1)}$ for $i = 1, \ldots, r$, where $g^{(i)} \in R^{(i)}$ is as defined in Equation (2.1). So, another way of homomorphically applying the full scaled trace is to apply the corresponding scaled trace in sequence for each level of the tower, "climbing down" from $R' = R^{(r)}$ to $R = R^{(0)}$. In particular, if we use the above sum-automorphisms procedure with a tower of finest granularity, then there are at most $\log_2(m'/m) = O(\log \lambda)$ levels, and since we have assumed that every prime divisor of m'/m is bounded by polylogarithmic in the security parameter λ, the full procedure will run in quasilinear $\tilde{O}(\lambda)$ time.

For technical reasons related to the analysis of noise terms under automorphisms, we actually use the sum-automorphisms procedure only on levels $R^{(i)}/R^{(i-1)} = \mathcal{O}_{m_i}/\mathcal{O}_{m_{i-1}}$ of the tower where every odd prime dividing m_i also divides m_{i-1}. Otherwise, we instead apply the scaled trace via an alternative procedure using ring-switching, which has essentially the same runtime (see Section 4.2.2 below for details). In fact, the alternative procedure can actually be used for *any* level of the tower, but it has the slight disadvantage of requiring the index of the ciphertext ring to be divisible by at least one prime that does not divide m_i; this is why we prefer not to use it when, e.g., m_i is a power of two.

4.2.1 Details of the Sum-Automorphisms Procedure

Here we specify the procedure for homomorphically applying the scaled trace by summing automorphisms, as sketched above. Let $R'/R = \mathcal{O}_{m'}/\mathcal{O}_m$ be a cyclotomic extension, where here m, m' are just dummy indices, not necessarily the ones from above. As already mentioned, we require that every odd prime dividing m' also divides m. The procedure takes as input a ciphertext c'' over some $R'' \supseteq R'$ that encrypts a plaintext $w' \in R'_{q'}$ under secret key $s'' \in R''$, where $q' = (\hat{m}'/\hat{m})q$ and w' is divisible by (g'/g). It proceeds as follows:

1. Compute ciphertexts $\tau_i(c'')$ over R'' for a certain set of automorphisms τ_i of R''/R that induce the automorphisms of R'/R. These ciphertexts will respectively encrypt $\tau_i(w') \in R'_{q'}$ under secret key $\tau_i(s'')$. Then key-switch [BV11a, BGV12] these to ciphertexts $c^{(i)}$ encrypting $\tau_i(w')$ under a common secret key $\tilde{s}$. See below for further details.

2. Sum the ciphertexts $c^{(i)}$ (component-wise) to get a new ciphertext $\tilde{c}$ that encrypts (under secret key $\tilde{s}$) the plaintext $\mathrm{Tr}_{R'/R}(w') = \sum_i \tau_i(w') \in R_{q'}$, which is divisible by $\hat{m}'/\hat{m}$.

3. Using the procedure from Section 2.2.2, reduce the plaintext modulus to q, resulting in a ciphertext that encrypts the scaled trace $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}(w') \in R_q$ under $\tilde{s}$.

The correctness of Steps 2 and 3 is immediate, so we just need to give the details of Step 1. We need to choose automorphisms τ_i of R''/R that induce the automorphisms of R'/R. Recall that the latter are defined by $\tau_j(\zeta_{m'}) = \zeta_{m'}^j$ for all $j \in \mathbb{Z}^*_{m'}$ such that $j = 1 \pmod m$. For each such j, we choose an $i \in \mathbb{Z}^*_{m''}$ such that $i = j \pmod{m'}$ and such that i is 1 modulo every prime p that divides m'' but not m'; this is possible by the Chinese Remainder Theorem. Then $\tau_i(\zeta_{m''}) = \zeta_{m''}^i$ is an automorphism of R''/R that induces τ_j, because $i = 1 \pmod m$ and

$$\tau_i(\zeta_{m'}) = \zeta_{m''}^{(m''/m')i} = \zeta_{m'}^j.$$

Also, by our assumption on m, m', each i we use is 1 modulo every prime that divides m'', because every such prime either divides m, or does not divide m', or is 2.

To complete the details of Step 1, we need to show why the ciphertext $\tau_i(c'')$ encrypts $\tau_i(w') \in R'_{q'}$ under secret key $\tau_i(s'')$. This follows from the decryption relation for c'', and the fact that τ_i is a ring homomorphism that induces an automorphism of R' and fixes $\mathbb{Z} \subseteq R$ pointwise:

$$\tau_i(c''_0) + \tau_i(c''_1) \cdot \tau_i(s'') = \frac{q}{p} \cdot \tau_i(\mu) + \tau_i(e''),$$

where the error term $e'' \in R''$ of c'' is such that $e'' \cdot g''$ is short (under the canonical embedding of R'').

The only subtlety is that we need $\tau_i(e'') \cdot g''$ to be short. We show below that $g'' = \tau_i(g'')$, from which it follows that $\tau_i(e'') \cdot g'' = \tau_i(e'' \cdot g'')$, which is short because the automorphisms of R'' simply permute the coordinates of the canonical embedding, and hence preserve norms (see, e.g., [LPR10, Lemma 5.6]). To see that $g'' = \tau_i(g'')$, recall that $i \in \mathbb{Z}^*_{m''}$ is 1 modulo every prime p that divides m''. Therefore, τ_i fixes every ζ_p and hence also fixes g''.[6]

Lastly, we briefly analyze the efficiency of the procedure. Applying automorphisms to the ciphertext ring elements is a trivial linear-time operation in the dimension, when the element is represented in any of the structured bases we consider (and also in the so-called "Chinese remainder" basis). Similarly, key-switching is quasilinear time in the bit length of the ciphertext, which itself is quasilinear in our context.

4.2.2 Applying the Trace via Ring-Switching

Here we describe the alternative procedure for applying the scaled trace, which uses the ring-switching technique from [GHPS12] (see Proposition 2.3). Let $R'/R = \mathcal{O}_{m'}/\mathcal{O}_m$ be an arbitrary cyclotomic extension, where m, m' are again dummy variables. For this procedure, we require that the ciphertext ring $R'' = \mathcal{O}_{m''} \supseteq R'$ be such that m''/m' is coprime with m', but otherwise we can choose m'' however we like. As before, the input is a ciphertext c'' over R'' that encrypts a plaintext $w' \in R'_{q'}$, where w' is divisible by (g'/g).

The main idea is that since m' and m''/m' are coprime, we can write $R'' \cong R' \otimes U$ where $U = \mathcal{O}_{m''/m'}$ and the tensor product is over the largest common base ring $\mathbb{Z}$. Then the R-linear function $\mathrm{Tr}_{R'/R}$ is induced by the $(R \otimes U)$-linear function $L\colon (R' \otimes U) \to (R \otimes U)$ defined by $L(a' \otimes u) = \mathrm{Tr}_{R'/R}(a') \otimes u$ for all $a' \in R', u \in U$. So, using the ring-switching procedure from Proposition 2.3, we can homomorphically evaluate L on ciphertext c'', yielding an encryption of $\mathrm{Tr}_{R'/R}(w')$, and then scale down the plaintext and its modulus as usual. One nice fact we highlight is that using ring-switching to evaluate the function $\mathrm{Tr}_{R'/R}$ does not incur *any* multiplicative increase in the noise rate, only a small additive one from the key-switching step. This is because the factor associated with the function $\mathrm{Tr}_{R'/R}$ that is applied to the ciphertext in the ring-switching procedure is simply 1.

One very important point is that ring-switching requires ring-LWE to be hard over the target ring $\mathcal{O}_{m'' \cdot m/m'} \cong R \otimes U$, so its dimension must be sufficiently large, but at the same time we cannot make the dimension of $R'' = \mathcal{O}_{m''}$ too large, for efficiency reasons. Therefore, we only use the procedure when m'/m is small, and for sufficiently large m''. Note that if the m'' associated with a given input ciphertext is too small, we can trivially increase it by embedding into a larger cyclotomic ring.

[6] If, contrary to our assumption, m' was divisible by one or more primes that did not divide m, then the error term $\tau_i(e'' \cdot g'')$ appearing in the ciphertext would be accompanied by a factor of $g''/\tau_i(g'')$. The expansion associated with this term can be bounded and is not excessive, but it depends on the number and sizes of the primes dividing m' and not m. By contrast, the alternative procedure described in Section 4.2.2 incurs no multiplicative increase in the noise rate.

5 Homomorphic Ring Rounding

In this section we describe how to efficiently homomorphically evaluate the "ring rounding function" $\lfloor \cdot \rceil_p \colon R_q \to R_p$, where $R = \mathcal{O}_m$ is the mth cyclotomic ring. Conceptually, we follow the high-level strategy from [GHS12a], but instantiate it with very different technical components. Recall from Section 2.2.1 that the rounding function expresses its input u in the "decryption" $\mathbb{Z}$-basis $B = \{b_j\}$ of R, as $u = \sum_j u_j \cdot b_j$ for $u_j \in \mathbb{Z}_q$, and outputs $\lfloor u \rceil_p := \sum_j \lfloor u_j \rceil_p \cdot b_j \in R_p$. Unlike with integer rounding from $\mathbb{Z}_q$ to $\mathbb{Z}_p$, it is not clear whether this rounding function has a low-depth arithmetic formula using just the ring operations of R. One difficulty is that there are an *exponentially* large number of values in R_q that map to a given value in R_p, which might be seen as evidence that a corresponding arithmetic formula must have large depth. Fortunately, we show how to circumvent this issue by using an additional homomorphic operation, namely, an enhancement of ring-switching. In short, we reduce the homomorphic evaluation of the ring rounding function (from R_q to R_p) very simply and efficiently to that of several parallel (batched) evaluations of the integer rounding function (from $\mathbb{Z}_q$ to $\mathbb{Z}_p$).

5.1 Overview

Suppose we choose some cyclotomic ring $S = \mathcal{O}_\ell$ having a mod-q CRT set $C = \{c_j\} \subset S$ of cardinality exactly $|B|$. That is, we have a ring embedding from the product ring $\mathbb{Z}_q^{|B|}$ into S_q, given by $\mathbf{u} \mapsto \sum_j u_j \cdot c_j$. Note that the choice of the ring S is at our convenience, and need not have any relationship to the plaintext ring R_q. We express the rounding function $R_q \to R_p$ as a sequence of three steps:

1. Map $u = \sum_j u_j \cdot b_j \in R_q$ to $\sum_j u_j \cdot c_j \in S_q$, i.e., send the $\mathbb{Z}_q$-coefficients of u (with respect to the decryption basis B) to the $\mathbb{Z}_q$-slots of S_q.
2. Batch-apply the integer rounding function from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ to the slot values u_j, to get $\sum_j \lfloor u_j \rceil_p \cdot c_j \in S_p$.
3. Invert the map from the first step to obtain $\lfloor u \rceil_p = \sum_j \lfloor u_j \rceil_p \cdot b_j \in R_p$.

Using batch/SIMD operations [SV11], the second step is easily achieved using the fact that S_q embeds the product of several copies of the ring $\mathbb{Z}_q$, via the CRT elements c_j. That is, we can simultaneously round all the coefficients u_j to $\mathbb{Z}_p$, using just one evaluation of an arithmetic procedure over S corresponding to one for the integer rounding function from $\mathbb{Z}_q$ to $\mathbb{Z}_p$.

We now describe one way of expressing the first and third steps above, in terms of operations that can be evaluated homomorphically. The first simple observation is that the function mapping $u = \sum_j u_j \cdot b_j$ to $\sum_j u_j \cdot c_j$ is induced by a $\mathbb{Z}$-linear function $\bar{L} \colon R \to S$. Specifically, $\bar{L}$ simply maps each $\mathbb{Z}$-basis element b_j to c_j. Now suppose that we choose S so that its largest common subring with R is $\mathbb{Z}$, i.e., the indices m, ℓ are coprime. Then letting $T = R + S = \mathcal{O}_{m\ell} \cong R \otimes S$ be the compositum ring, Lemma 2.2 yields an S-linear function $L \colon T \to S$ that coincides with $\bar{L}$ on $R \subseteq T$, and in particular on u. The ring-switching procedure from Proposition 2.3 can homomorphically evaluate any S-linear function from T to S, and in particular, the function L. Therefore, by simply embedding R into T, we can homomorphically evaluate $\bar{L}(x) = L(x)$ by applying the ring-switching procedure with L.

Unfortunately, there is a major problem with the efficiency of the above approach: the *dimension* (over $\mathbb{Z}$) of the compositum ring T is the product of those of R and S, which are each at least linear in the security parameter. Therefore, representing and operating on arbitrary elements in T requires at least quadratic time.

5.1.1 Efficiently Mapping from B to C

In hindsight, the quadratic runtime of the above approach should not be a surprise, because we treated $\bar{L}\colon R \to S$ as an arbitrary $\mathbb{Z}$-linear transformation, and B, C as arbitrary sets. To do better, $\bar{L}$, B, and C must have some structure we can exploit. Fortunately, they can—*if* we choose them carefully. We now describe a way of expressing the first and third steps above in terms of simple operations that can be evaluated homomorphically in quasilinear time.

The main idea is as follows, and is summarized in Figure 1. Instead of mapping directly from R to S, we will express $\bar{L}$ as a *sequence* of linear transformations $\bar{L}_1, \ldots, \bar{L}_r$ through several "hybrid" cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$. For sets B and C with an appropriate product structure, these transformations will respectively map $A_0 = B \subset H^{(0)}$ to some structured subset $A_1 \subset H^{(1)}$, then A_1 to some structured subset $A_2 \subset H^{(2)}$, and so on, finally mapping A_{r-1} to $A_r = C \subset H^{(r)}$. In contrast to the inefficient method described above, the hybrid rings will be chosen so that each compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ of adjacent rings has dimension just *slightly larger* (by only a polylogarithmic factor) than that of R. This is achieved by choosing the indices of $H^{(i-1)}, H^{(i)}$ to have large greatest common divisor, and hence small least common multiple. For example, the indices can share almost all the same prime divisors (with multiplicity), and have just one different prime divisor each. Of course, other tradeoffs between the number of hybrid rings and the dimensions of the compositums are also possible.

The flip side of this approach is that using ring-switching, we can homomorphically evaluate only $E^{(i)}$-*linear* functions $\bar{L}_i\colon H^{(i-1)} \to H^{(i)}$, where $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ is the largest common subring of adjacent hybrid rings. Since each $E^{(i)}$ is large by design (to keep the compositum $T^{(i)}$ small), this requirement is quite strict, yet we still need to construct linear functions $\bar{L}_i$ that sequentially map $B = A_0$ to $C = A_r$. To achieve this, we construct all the sets A_i to have appropriate product structure. Specifically, we ensure that for each $i = 1, \ldots, r$, we have factorizations

$$A_{i-1} = A_{i-1}^{\text{out}} \cdot Z_i, \quad A_i = A_i^{\text{in}} \cdot Z_i \tag{5.1}$$

for some set $Z_i \subset E^{(i)}$, where both A_{i-1}^{out} and A_i^{in} are linearly independent over $E^{(i)}$. (Note that for $1 \le i < r$, each A_i needs to factor in two ways over two subrings $E^{(i-1)}$ and $E^{(i)}$, which is why we need two sets A_i^{in} and A_i^{out}.) Then, we simply define $\bar{L}_i$ to be an arbitrary $E^{(i)}$-linear function that bijectively maps A_{i-1}^{out} to A_i^{in}. (Note that A_{i-1}^{out} and A_i^{in} have the same cardinality, because A_{i-1} and A_i do.) It immediately follows that $\bar{L}_i$ bijectively maps A_{i-1} to A_i, because

$$\bar{L}_i(A_{i-1}) = \bar{L}_i(A_{i-1}^{\text{out}} \cdot Z_i) = \bar{L}_i(A_{i-1}^{\text{out}}) \cdot Z_i = A_i^{\text{in}} \cdot Z_i$$

by $E^{(i)}$-linearity and the fact that $Z_i \subset E^{(i)}$.

Summarizing the above discusion, we have the following theorem.

Theorem 5.1. *Suppose there exists a sequence of cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$ and sets $A_i \subset H^{(i)}$ such that for all $i = 1, \ldots, r$, we have $A_{i-1} = A_{i-1}^{out} \cdot Z_i$ and $A_i = A_i^{in} \cdot Z_i$ for some sets $Z_i \subset E^{(i)} = H^{(i-1)} \cap H^{(i)}$ and A_{i-1}^{out}, A_i^{in} that are each $E^{(i)}$-linearly independent and of equal cardinality. Then there is a sequence of $E^{(i)}$-linear maps $\bar{L}_i\colon H^{(i-1)} \to H^{(i)}$, for $i = 1, \ldots, r$, whose composition $\bar{L}_r \circ \cdots \circ \bar{L}_1$ bijectively maps A_0 to A_r.*

5.1.2 Applying the Map Homomorphically

So far we have described how our desired map between *plaintext* rings R and S can be expressed as a sequence of linear maps through hybrid plaintext rings. In the context of bootstrapping, for security these

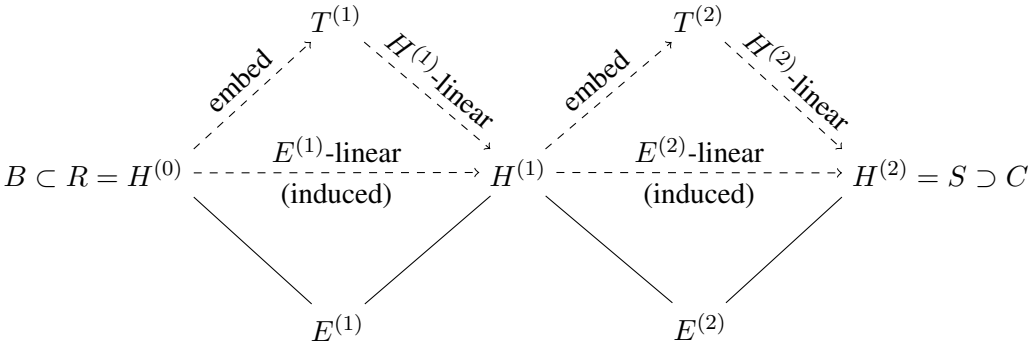


Figure 1: An example mapping from $B \subset R$ to $C \subset S$, via a sequence of hybrid rings. Each $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ is a largest common subring, and each $T^{(i)} = H^{(i-1)} + H^{(i)}$ is a compositum, of adjacent hybrid rings. For any $E^{(i)}$-linear function from $H^{(i-1)}$ to $H^{(i)}$, there is a corresponding $H^{(i)}$-linear function from $T^{(i)}$ to $H^{(i)}$ that coincides with it on $H^{(i-1)}$ (see Lemma 2.2).

plaintext rings typically need to be embedded in some larger ciphertext rings, because the dimensions of R, S are not large enough to securely support the very small noise used in bootstrapping. For example, following Step 2 of our bootstrapping procedure (Section 3), we have a ciphertext over the ring R'' where $R'' = \mathcal{O}_{m''} \supseteq R$ for some m'' of our choice that is divisible by m. We need to choose the sequence of hybrid *ciphertext* rings so that they admit linear functions (over the respective largest common subrings) that induce the desired ones on the underlying *plaintext* rings. This turns out to be very easy to do, as we now explain.

Let H, H' be adjacent hybrid plaintext rings having largest common subring $E = H \cap H'$ and compositum $T = H + H'$. Then we want the corresponding ciphertext rings to be $\tilde{H} \cong H \otimes U$, $\tilde{H}' \cong H' \otimes U'$ for some cyclotomic rings U, U', and the largest common subring and compositum of $\tilde{H}, \tilde{H}'$ to be $\tilde{E} \cong E \otimes (U \cap U')$ and $\tilde{T} \cong T \otimes (U + U')$, respectively (where all the tensor products are over the common base ring $\mathbb{Z}$). Then any E-linear function $L\colon H \to H'$ is induced by any $\tilde{E}$-linear function $\tilde{L}\colon \tilde{H} \to \tilde{H}'$ satisfying $\tilde{L}(h \otimes 1) = L(h) \otimes 1$, which is the function we actually apply when switching between ciphertext rings.

To satisfy the above conditions, it is sufficient (and in fact necessary) to choose the respective indices u, u' of U, U' so that $\mathrm{lcm}(u, u')$ is coprime with $\mathrm{lcm}(h, h')$, where h, h' are the respective indices of H, H'. Then the ciphertext rings $\tilde{H}, \tilde{H}'$ have indices hu and $h'u'$, and their compositum has index $\mathrm{lcm}(h, h') \cdot \mathrm{lcm}(u, u')$, which must be quasilinear for asymptotic efficiency. In typical instantiations, in order to get enough additional slots in each successive ring, h'/h will be moderately large and $\mathrm{lcm}(h, h') \approx h'$. So to ensure that all the ciphertext rings are about the same size, we can choose $u/u' \approx h'/h$ and $\mathrm{lcm}(u, u') \approx u$.

5.1.3 Mapping Selected Coefficients

In some settings we may only need to map certain coefficients into slots, i.e., map a particular portion of B to a CRT set of appropriate size. For example, when bootstrapping a semi-packed ciphertext over $R' = \mathcal{O}_{m'}$ with plaintext over $\tilde{R} = \mathcal{O}_{\tilde{m}}$, we may need to artificially expand our view of the plaintext ring to some $R = \mathcal{O}_m$, so that m is coprime with m'/m (see the constraints listed at the start of Section 3). In such a case, the decryption basis B of R factors as $B = B' \cdot \tilde{B}$, where $\tilde{B}$ is the decryption basis of $\tilde{R}$ and $B' \subset R$ is a particular $\tilde{R}$-basis of R. Since the true message is really only over $\tilde{R}$, it can be shown that the only coefficients we need to recover the message are associated with the subset $b' \cdot \tilde{B} \subseteq B$ for a particular fixed

$b' \in B'$. Therefore, when designing the hybrid rings and CRT sets we only need $|\tilde{B}|$ slots in total. When initially switching from R through the hybrid rings, we do so in a way that maps b' to one entry of a CRT set and all the other elements of B' to zero, then continue by mapping all of $\tilde{B}$ to a CRT set as usual. Note that we still need to go through just as many hybrid rings to map from R to S, but the size of S can be significantly smaller because it needs fewer CRT slots.

5.2 Construction

By Theorem 5.1 and the ring-switching procedure, in order to map $B \subset R$ to a CRT set C of some ring S of our choice in a way that can be efficiently evaluated homomorphically, we just need to construct hybrid cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$ and sets $A_i \subset H^{(i)}$ (where $A_0 = B$ and $A_r = C$) to satisfy the following two properties for each $i = 1, \ldots, r$:

1. Each compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ is not too large, i.e., its dimension is quasilinear.
2. The sets A_{i-1}, A_i factor as described in Equation (5.1).

The remainder of this subsection is dedicated to providing such a construction.

5.2.1 Decomposition of R and Basis $B \subset R$

For our given ring $R = \mathcal{O}_m$ and its $\mathbb{Z}$-basis B used in decryption, we consider a tower of cyclotomic rings

$$R^{(r)}/R^{(r-1)}/\cdots/R^{(1)}/R^{(0)},$$

where $R^{(r)} = R$ and $R^{(0)} = \mathcal{O}_1 = \mathbb{Z}$, and each $R^{(i)}$ has index m_i, which is divisible by m_{i-1} for $i > 0$. For example, in a finest-grained decomposition, r is the number of prime divisors (with multiplicity) of m, and the ratios m_i/m_{i-1} are all these prime divisors in some arbitrary order. A coarser-grained decomposition may be used as well, but will tend to make the compositum rings $T^{(i)}$ larger.

We need $\mathbb{Z}$-bases B_i of the rings $R^{(i)}$ that have a product structure induced by the tower. Specifically, for each $i = 1, \ldots, r$ we need to have the factorization

$$B_i = B'_i \cdot B_{i-1} \subset R^{(i)} \tag{5.2}$$

for some set $B'_i \subset R^{(i)}$ that is linearly independent over $R^{(i-1)}$. We also need the basis $B^{(r)}$ of $R = R^{(r)}$ to be the one used for decryption. As shown in Section 2.1.4, the scaled-up "decoding" basis of R has a finest-possible factorization, so we can use it as B for any choice of the tower.

We mention that the power basis $\{1, \zeta_m, \zeta_m^2, \ldots, \zeta_m^{\varphi(m)-1}\}$ of R, which is implicitly the one used when representing R as the polynomial ring $\mathbb{Z}[X]/\Phi_m(X)$, *does not* have the required product structure when m is divisible by two or more odd primes, but that it does coincide with the scaled-up decoding basis when m is a power of 2. (See [LPR13] for details.)

5.2.2 Ring S and CRT Set $C \subset S$.

We next design $S = \mathcal{O}_\ell$ so that it also yields a tower of cyclotomic rings $S^{(r)}/S^{(r-1)}/\cdots/S^{(1)}/S^{(0)}$, where $S^{(r)} = S$ and $S^{(0)} = \mathbb{Z}$, and each $S^{(i)}$ has index ℓ_i. As described in Sections 2.1.3 and 2.1.4, there are structured mod-q CRT sets $\tilde{C}_i$ of $S^{(i)}$ that factor as

$$\tilde{C}_i = \tilde{C}'_i \cdot \tilde{C}_{i-1},$$

where $\tilde{C}'_i \subset S^{(i)}$ is an $S^{(i-1)}$-linearly independent set whose cardinality is the "relative splitting number" of p in $S^{(i)}/S^{(i-1)}$, i.e., the number of distinct prime ideals in $S^{(i)}$ lying over any prime ideal divisor of p in $S^{(i-1)}$.

We need to choose the ring S and its tower so that for all $i = 1, \ldots, r$,

- the respective indices m_{r-i+1}, ℓ_i of $R^{(r-i+1)}, S^{(i)}$ are coprime (certainly it suffices for m and ℓ to be coprime, but this is not always necessary);
- the dimension $\varphi(m_{r-i+1} \cdot \ell_i)$ is not too large (specifically, it is quasi-linear in the security parameter);
- the relative splitting number $|\tilde{C}'_i| \geq |B'_{r-i+1}|$.

We can then easily define structured CRT sets $C_i \subset \tilde{C}_i \subset S^{(i)}$ of the appropriate cardinality, and in particular $C = C_r$, as follows. Define $C_0 = \{1\} \subset \mathbb{Z} = S^{(0)}$. Then for each $i = 1, \ldots, r$, let $C'_i \subseteq \tilde{C}'_i$ be an arbitrary subset having cardinality exactly $|B'_{r-i+1}|$, and define

$$C_i = C'_i \cdot C_{i-1} \subset \tilde{C}_i. \tag{5.3}$$

5.2.3 Hybrid Rings $H^{(i)}$ and Sets $A_i \subset H^{(i)}$

Informally, with each successive hybrid ring we remove another level from the R-tower and add on another level to the S-tower, and similarly with the corresponding components of the structured sets B and C. Formally, for $i = 0, 1, \ldots, r$ we define

$$\begin{aligned} H^{(i)} &= \mathcal{O}_{m_{r-i}\,\ell_i} \cong R^{(r-i)} \otimes S^{(i)}, \\ A_i &= B_{r-i} \cdot C_i \subset H^{(i)}, \end{aligned} \tag{5.4}$$

where the tensor product in Equation (5.4) applies to the rings as extensions of $\mathbb{Z}$, and the isomorphism holds because $\gcd(m_{r-i}, \ell_i) \leq \gcd(m_{r-i+1}, \ell_i) = 1$ by design. Note that $H^{(0)} = \mathcal{O}_{m_r} = R$, $H^{(r)} = \mathcal{O}_{\ell_r} = S$, and $A_0 = B_r = B$, $A_r = C_r = C$, as required.

For each $i = 1, \ldots, r$, because m_{r-i+1} and ℓ_i are coprime, it is straightforward to verify that the largest common subring $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ and compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ are

$$\begin{aligned} E^{(i)} &= \mathcal{O}_{m_{r-i}\,\ell_{i-1}} \cong R^{(r-i)} \otimes S^{(i-1)} \\ T^{(i)} &= \mathcal{O}_{m_{r-i+1}\,\ell_i} \cong R^{(r-i+1)} \otimes S^{(i)}, \end{aligned}$$

where the tensor products above are over the common base ring $\mathbb{Z}$. Note that the dimension of $T^{(i)}/\mathbb{Z}$ is $\varphi(m_{r-i+1} \cdot \ell_i)$, which is quasi-linear in the security parameter by construction.

Lemma 5.2. *The sets A_{i-1}, A_i factor as in Equation* (5.1)*, i.e., $A_{i-1} = A^{out}_{i-1} \cdot Z_i$ and $A_i = A^{in}_i \cdot Z_i$ for some sets $Z_i \subset E^{(i)}$ and A^{out}_{i-1}, A^{in}_i that are each $E^{(i)}$-linearly independent and of equal cardinality.*

Proof. Define $Z_i = B_{r-i} \cdot C_{i-1} \subset E^{(i)}$. Recall from Equation (5.2) that $B_{r-i+1} = B'_{r-i+1} \cdot B_{r-i}$, where $B'_{r-i+1} \subset R^{(r-i+1)}$ is linearly independent over $R^{(r-i)} \subset H^{(i-1)}$, and hence also over $E^{(i)} \cong R^{(r-i)} \otimes S^{(i-1)}$ (because it corresponds to the set of pure tensors $B'_{r-i+1} \otimes \{1\} \subset R^{(r-i+1)} \otimes S^{(i-1)}$). Then

$$A_{i-1} = (B'_{r-i+1} \cdot B_{r-i}) \cdot C_{i-1} = B'_{r-i+1} \cdot Z_i$$

is the desired factorization. Similarly, recall from Definition (5.3) that $C_i = C'_i \cdot C_{i-1}$, where $C'_i \subseteq \tilde{C}'_i \subset S^{(i)}$ is linearly independent over $S^{(i-1)}$, and hence also over $E^{(i)}$. Then we have the desired factorization

$$A_i = B_{r-i} \cdot (C'_i \cdot C_{i-1}) = C'_i \cdot Z_i.$$

Finally, we have $|A^{\text{out}}_{i-1}| = |B'_{r-i+1}| = |C'_i| = |A^{\text{in}}_i|$ by design of C'_i. □

References

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ICTS*, pages 309–325. 2012.

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. 2012.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. 2011.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CCK+13] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 315–335. 2013.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GH11a] C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS*, pages 107–109. 2011.

[GH11b] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148. 2011.

[GHPS12] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN*, pages 19–37. 2012. Full version at `http://eprint.iacr.org/2012/240`.

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography*, pages 1–16. 2012.

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482. 2012.

[GHS12c] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, pages 850–867. 2012.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 2013. To appear. Preliminary version in Eurocrypt 2010.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013.

[SV11] N. Smart and F. Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011. `http://eprint.iacr.org/`.

[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43. 2010.

A Transformation Between LSB and MSB Encodings

Here we describe a folklore transformation between the "least significant bit" and "most significant bit" message encodings for (ring-)LWE-based cryptosystems.

Let plaintext modulus p and ciphertext modulus q be coprime, fix integers c_p, c_q such that $c_p p + c_q q = 1$, and observe that $c_p = p^{-1} \pmod q$ and $c_q = q^{-1} \pmod p$.

- An lsb encoding of a value $\mu \in \mathbb{Z}_p$ is any $v \in \mathbb{Z}_q$ such that $v = e \pmod q$ for some integer $e \in [-q/2, q/2)$ where $e = \mu \pmod p$.
- An msb encoding of μ is any $w \in \mathbb{Z}_q$ such that $\lfloor w \rceil_p := \lfloor w \cdot (p/q) \rceil = \mu \pmod p$.

If $v \in \mathbb{Z}_q$ is an lsb encoding of $\mu \in \mathbb{Z}_p$, then we claim that $p^{-1} \cdot v \in \mathbb{Z}_q$ is an msb encoding of $-q^{-1} \cdot \mu \in \mathbb{Z}_p$. Indeed, since $v = e \pmod q$ for some $e \in (\mu + p\mathbb{Z}) \cap [-q/2, q/2)$, we have

$$\lfloor p^{-1} \cdot v \rceil_p = \left\lfloor \tfrac{1 - c_q q}{p} \cdot e \cdot \tfrac{p}{q} \right\rceil = \left\lfloor (\tfrac{1}{q} - c_q) \cdot e \right\rceil = -c_q \cdot e = -q^{-1} \cdot \mu \pmod p.$$

In the other direction, if $w \in \mathbb{Z}_q$ is an msb encoding of $\mu \in \mathbb{Z}_p$, then we claim that $p \cdot w$ is an lsb encoding of $-q \cdot \mu \in \mathbb{Z}_p$. Indeed, by assumption we have

$$\lfloor w \rceil_p = \lfloor w \cdot (p/q) \rceil = w \cdot (p/q) - f = \mu \pmod p$$

for some $f \in \frac{1}{q}\mathbb{Z} \cap [-1/2, 1/2)$. Multiplying by q and letting $e = q \cdot f \in \mathbb{Z} \cap [-q/2, q/2)$, we have

$$p \cdot w - e = q \cdot \mu \pmod{pq}.$$

Reducing this modulo q, we get $p \cdot w = e \pmod q$, and reducing it modulo p, we have $e = -q \cdot \mu \pmod p$.

The above facts make it possible to convert between lsb and msb representations of (ring-)LWE ciphertexts, simply by multiplying the ciphertext by p or p^{-1} modulo q. This works because decryption recovers a $\mathbb{Z}_q$-encoding of the message simply as a linear function of the ciphertext, so the p or p^{-1} factor simply "passes through" the ciphertext to the encoding. (In the ring setting, the encoding of plaintext ring elements is coefficient-wise in a certain basis, so the same reasoning applies.) If $q = -1 \pmod p$, then the above transformations preserve the message exactly. In other cases, we can just keep track of the factors of $-q$ or $-q^{-1}$ introduced by the conversions (which may be affected by other homomorphic operations), and remove them upon decryption.

B Integer Rounding Procedure

Here we recall (a close variant of) the efficient arithmetic procedure from [GHS12a] for computing the "most significant bit" function $\mathrm{msb}_q \colon \mathbb{Z}_q \to \mathbb{Z}_2$ for $q = 2^\ell \geq 4$, defined as $\mathrm{msb}_q(x) = \lfloor x/(q/2) \rfloor$. Note that the integer rounding function $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \mathbb{Z}_2$ is simply $\lfloor x \rceil_2 = \mathrm{msb}_q(x + q/4)$. The multiplicative depth and cost (in number of operations) of the msb_q procedure are not precisely analyzed in [GHS12a], and the procedure as written turns out to be suboptimal in depth and number of operations by $\log_2(q)$ factors, because it (homomorphically) raises ciphertexts to large powers in an inner loop. So for completeness, here we

present a simplified and optimized version of the procedure, and an analysis of its depth and cost. It uses the standard ring operations of $\mathbb{Z}_{2^j}$, as well as division by 2 of values that are guaranteed to be even. All of these operations can be evaluated homomorphically for the cryptosystem described in Section 2.2, as explained in Section 2.2.2. The procedure also easily generalizes to any prime base.

Algorithm 1 Arithmetic procedure for computing $\mathrm{msb}_q \colon \mathbb{Z}_q \to \mathbb{Z}_2$ [GHS12a]

Input: Element $x \in \mathbb{Z}_q$, where $q = 2^\ell$ for some positive integer ℓ
Output: $\mathrm{msb}_q(x) \in \mathbb{Z}_2$

1: $w_0 \leftarrow x$ // $w_0 \in \mathbb{Z}_q$
2: **for** $i \leftarrow 1, \ldots, \ell - 1$ **do**
3: $\quad y \leftarrow x$ // $y \in \mathbb{Z}_q$, $y = x \pmod{2^{i+1}}$
4: $\quad$ **for** $j \leftarrow 0, \ldots, i-1$ **do**
5: $\quad\quad w_j \leftarrow w_j^2$ // now $w_j = \mathrm{lsb}(\lfloor x/2^j \rfloor) \pmod{2^{i-j+1}}$
6: $\quad\quad y \leftarrow (y - w_j)/2 \bmod (q/2^{j+1})$ // now $y \in \mathbb{Z}_{q/2^{j+1}}$, $y = \lfloor x/2^{j+1} \rfloor \pmod{2^{i-j}}$
7: $\quad w_i \leftarrow y$ // $w_i \in \mathbb{Z}_{q/2^i}$, $w_i = \lfloor x/2^i \rfloor \pmod 2$
8: **return** $w_{\ell-1} \in \mathbb{Z}_2$

Correctness follows from [GHS12a, Lemma 2]. The main idea is that when initially assigned, each w_j has the same least-significant bit as $\lfloor x/2^j \rfloor$, i.e., $w_j = \lfloor x/2^j \rfloor \pmod 2$ (but its other bits may not agree with x's). Each time w_j is squared in Step 5, its least-significant bit remains the same, but an additional more-significant bit is set to zero. That is, after t squarings, $w_j = \mathrm{lsb}(\lfloor x/2^j \rfloor) \pmod{2^{t+1}}$. Therefore, in iteration i, the inner loop "shifts away" the i least-significant bits of x, leaving the $(i+1)$st bit intact in the least significant position (but possibly changing the others), at which point we can assign w_i and maintain the invariant.

We now briefly analyze the *homomorphic* evaluation of the procedure, in terms of its induced noise growth and runtime cost. The most important observation is that although it is written using a doubly nested loop, the procedure actually has multiplicative depth exactly $\ell - 1 = \log_2(q/2)$. This is because in the inner loop, each w_j for $j = 0, \ldots, i-1$ can be squared in parallel (Step 5). Each squaring of the plaintext value $w_j \in \mathbb{Z}_{q/2^j}$ induces the usual small polynomial expansion $(q/2^j) \cdot n^c$ (where $c \approx 1$) in the noise rate of the associated ciphertext. The iterated subtractions and divisions by 2 (Step 6) cause no growth at all in the noise rate: each subtraction sums (at worst) the noise rates of the associated ciphertexts, and division by 2 halves the noise rate.

In the ith iteration, the procedure performs i homomorphic multiplications and i subtractions (and also i divisions by 2, but these are trivial as homomorphic operations). Therefore, the procedure uses a total of $\ell(\ell-1)/2$ homomorphic multiplications and subtractions each.

C Concrete Choices of Rings

Here, for $p = 2$ and several values of the original cyclotomic index m, we give some workable values for the target cyclotomic index ℓ, along with the indices of the intermediate "hybrid" rings, the dimensions of the compositum rings, etc. Note that when $m_{r-i+1} = 2$, then the ring R_{r-i+1} has dimension 1, and so we can move directly from $m_{r-i+1} = 2$ to $m_{r-i+1} = 1$. In the tables below, and following the notation in Section 5:

- m_{r-i+1} is the index of the ring R_{r-i+1} at step i;

- ℓ_i is the index of the ring S_i at step i;
- $\varphi(m_{r-i+1} \cdot \ell_i)$ is the dimension of the compositum ring at step i;
- $|B'_{r-i+1}|$ is the dimension of the intermediate ring extension $R^{(r-i+1)}/R^{(r-i)}$;
- $|\tilde{C}'_i|$ is the "relative splitting number" of $p = 2$ in the extension $S^{(i)}/S^{(i-1)}$.
- r denotes the number of hybrid rings R_{r-i+1}, $i \in [r]$

All the indices are *lower bounds* needed to support the *functionality* of the ring-rounding technique on the plaintext space (Section 5). Larger ciphertext indices may be required to ensure adequate security for all the homomorphic operations; see Section 5.1.2.

Table 1: Concrete choices for $m_r = 1024, \varphi(m_r) = 512$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	1024	17	2	2	8192
2	512	$221 = 17 \cdot 13$	4	4	49152
3	128	$1547 = 221 \cdot 7$	4	6	73728
4	32	$7735 = 1547 \cdot 5$	4	4	73728
5	8	$23205 = 7735 \cdot 3$	2	2	36864
6	4	$69615 = 23205 \cdot 3$	2	3	55296
7	1	69615			55296

Table 2: Concrete choices for $m_r = 512, \varphi(m_r) = 256$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	512	17	2	2	4096
2	256	$221 = 17 \cdot 13$	4	4	24576
3	64	$1547 = 221 \cdot 7$	4	6	36864
4	16	$7735 = 1547 \cdot 5$	4	4	36864
5	4	$23205 = 7735 \cdot 3$	2	2	18432
6	1	23205			18432

Table 3: Concrete choices for $m_r = 256, \varphi(m_r) = 128$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	256	17	2	2	2048
2	128	$221 = 17 \cdot 13$	4	4	12288
3	32	$1105 = 221 \cdot 5$	4	4	12288
4	8	$3315 = 1105 \cdot 3$	2	2	6144
5	4	$9945 = 3315 \cdot 3$	2	3	9216
6	1	9945			9216

Table 4: Concrete choices for $m_r = 128, \varphi(m_r) = 64$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	128	17	2	2	2048
2	64	$119 = 17 \cdot 7$	2	2	3072
3	32	$595 = 119 \cdot 5$	4	4	6144
4	8	$1785 = 595 \cdot 3$	2	2	3072
5	4	$5355 = 1785 \cdot 3$	2	3	4608
6	1	5355			4608

Hardness of SIS and LWE with Small Parameters

Daniele Micciancio* Chris Peikert†

February 13, 2013

Abstract

The Short Integer Solution (SIS) and Learning With Errors (LWE) problems are the foundations for countless applications in lattice-based cryptography, and are provably as hard as approximate lattice problems in the worst case. A important question from both a practical and theoretical perspective is how small their parameters can be made, while preserving their hardness.

We prove two main results on SIS and LWE with small parameters. For SIS, we show that the problem retains its hardness for moduli $q \geq \beta \cdot n^\delta$ for any constant $\delta > 0$, where β is the bound on the Euclidean norm of the solution. This improves upon prior results which required $q \geq \beta \cdot \sqrt{n \log n}$, and is essentially optimal since the problem is trivially easy for $q \leq \beta$. For LWE, we show that it remains hard even when the errors are small (e.g., uniformly random from $\{0, 1\}$), provided that the number of samples is small enough (e.g., linear in the dimension n of the LWE secret). Prior results required the errors to have magnitude at least $\sqrt{n}$ and to come from a Gaussian-like distribution.

1 Introduction

In modern lattice-based cryptography, two average-case computational problems serve as the foundation of almost all cryptographic schemes: Short Integer Solution (SIS), and Learning With Errors (LWE). The SIS problem dates back to Ajtai's pioneering work [1], and is defined as follows. Let n and q be integers, where n is the primary security parameter and usually $q = \mathrm{poly}(n)$, and let $\beta > 0$. Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some $m = \mathrm{poly}(n)$, the goal is to find a nonzero integer vector $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$ (where $\|\cdot\|$ denotes Euclidean norm). Observe that β should be set large enough to ensure that a solution exists (e.g., $\beta > \sqrt{n \log q}$ suffices), but that $\beta \geq q$ makes the problem trivially easy to solve. Ajtai showed that for appropriate parameters, SIS enjoys a remarkable worst-case/average-case hardness property: solving it *on the average* (with any noticeable probability) is at least as hard as approximating several lattice problems on n-dimensional lattices *in the worst case*, to within $\mathrm{poly}(n)$ factors.

*University of California, San Diego. 9500 Gilman Dr., Mail Code 0404, La Jolla, CA 92093, USA. Email: daniele@cs.ucsd.edu. This material is based on research sponsored by DARPA under agreement number FA8750-11-C-0096 and NSF under grant CNS-1117936. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, NSF or the U.S. Government.

†School of Computer Science, Georgia Institute of Technology. This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by DARPA under agreement number FA8750-11-C-0096, and by the Alfred P. Sloan Foundation.

The LWE problem was introduced in the celebrated work of Regev [24], and has the same parameters n and q, along with a "noise rate" $\alpha \in (0,1)$. The problem (in its search form) is to find a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, given a "noisy" random linear system $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$, where $\mathbf{A}$ is uniformly random and the entries of $\mathbf{e}$ are i.i.d. from a Gaussian-like distribution with standard deviation roughly αq. Regev showed that as long as $\alpha q \geq 2\sqrt{n}$, solving LWE on the average (with noticeable probability) is at least as hard as approximating lattice problems in the worst case to within $\tilde{O}(n/\alpha)$ factors using a *quantum* algorithm. Subsequently, Peikert [21] gave a *classical* reduction for a subset of the lattice problems and the same approximation factors, but under the additional condition that $q \geq 2^{n/2}$ (or $q \geq 2\sqrt{n}/\alpha$ based on some non-standard lattice problems).

A significant line of research has been devoted to improving the tightness of worst-case/average-case connections for lattice problems. For SIS, a series of works [1, 7, 14, 19, 12] gave progressively better parameters that guarantee hardness, and smaller approximation factors for the underlying lattice problems. The state of the art (from [12], building upon techniques introduced in [19]) shows that for $q \geq \beta \cdot \omega(\sqrt{n \log n})$, finding a SIS solution with norm bounded by β is as hard as approximating worst-case lattice problems to within $\tilde{O}(\beta\sqrt{n})$ factors. (The parameter m does not play any significant role in the hardness results, and can be any polynomial in n.) For LWE, Regev's initial result remains the tightest, and the requirement that $q \geq \sqrt{n}/\alpha$ (i.e., that the errors have magnitude at least $\sqrt{n}$) is in some sense optimal: a clever algorithm due to Arora and Ge [2] solves LWE in time $2^{\tilde{O}(\alpha q)^2}$, so a proof of hardness for substantially smaller errors would imply a subexponential time (quantum) algorithm for approximate lattice problems, which would be a major breakthrough. Interestingly, the current modulus bound for LWE is in some sense better than the one for SIS by a $\tilde{\Omega}(\sqrt{n})$ factor: there are applications of LWE for $1/\alpha = \tilde{O}(1)$ and hence $q = \tilde{O}(\sqrt{n})$, whereas SIS is only useful for $\beta \geq \sqrt{n}$, and therefore requires $q \geq n$ according to the state-of-the-art reductions.

Further investigating the smallest parameters for which SIS and LWE remain provably hard is important from both a practical and theoretical perspective. On the practical side, improvements would lead to smaller cryptographic keys without compromising the theoretical security guarantees, or may provide greater confidence in more practical parameter settings that so far lack provable hardness. Also, proving the hardness of LWE for non-Gaussian error distributions (e.g., uniform over a small set) would make applications easier to implement. Theoretically, improvements may eventually shed light on related problems like Learning Parity with Noise (LPN), which can be seen as a special case of LWE for modulus $q = 2$, and which is widely used in coding-based cryptography, but which has no known proof of hardness.

1.1 Our Results

We prove two complementary results on the hardness of SIS and LWE with small parameters. For SIS, we show that the problem retains its hardness for moduli q nearly equal to the solution bound β. For LWE, we show that it remains hard even when the errors are small (e.g., uniformly random from $\{0,1\}$), provided that the number m of noisy equations is small enough. This qualification is necessary in light of the Arora-Ge attack [2], which for large enough m can solve LWE with binary errors in polynomial time. Details follow.

SIS with small modulus. Our first theorem says that SIS retains its hardness with a modulus as small as $q \geq \beta \cdot n^\delta$, for any $\delta > 0$. Recall that the best previous reduction [12] required $q \geq \beta \cdot \omega(\sqrt{n \log n})$, and that SIS becomes trivially easy for $q \leq \beta$, so the q obtained by our proof is essentially optimal. It also essentially closes the gap between LWE and SIS, in terms of how small a useful modulus can be. More precisely, the following is a special case of our main SIS hardness theorem; see Section 3 for full details.

Theorem 1.1 (Corollary of Theorem 3.8). *Let n and $m = \mathrm{poly}(n)$ be integers, let $\beta \geq \beta_\infty \geq 1$ be reals, let $Z = \{\mathbf{z} \in \mathbb{Z}^m : \|\mathbf{z}\|_2 \leq \beta$ and $\|\mathbf{z}\|_\infty \leq \beta_\infty\}$, and let $q \geq \beta \cdot n^\delta$ for some constant $\delta > 0$. Then solving (on the average, with non-negligible probability)* SIS *with parameters n, m, q and solution set $Z \setminus \{\mathbf{0}\}$ is at least as hard as approximating lattice problems in the worst case on n-dimensional lattices to within $\gamma = \max\{1, \beta \cdot \beta_\infty/q\} \cdot \tilde{O}(\beta\sqrt{n})$ factors.*

Of course, the ℓ_∞ bound on the SIS solutions can be easily removed simply setting $\beta_\infty = \beta$, so that $\|\mathbf{z}\|_\infty \leq \|\mathbf{z}\|_2 \leq \beta$ automatically holds true. We include an explicit ℓ_∞ bound $\beta_\infty \leq \beta$ in order to obtain more precise hardness results, based on potentially smaller worst-case approximation factors γ. We point out that the bound β_∞ and the associated extra term $\max\{1, \beta \cdot \beta_\infty/q\}$ in the worst-case approximation factor is not present in previous results. Notice that this term can be as small as 1 (if we take $q \geq \beta \cdot \beta_\infty$, and in particular if $\beta_\infty \leq n^\delta$), and as large as β/n^δ (if $\beta_\infty = \beta$). This may be seen as the first theoretical evidence that, at least when using a small modulus q, restricting the ℓ_∞ norm of the solutions may make the SIS problem qualitatively harder than just restricting the ℓ_2 norm. There is already significant empirical evidence for this belief: the most practically efficient attacks on SIS, which use lattice basis reduction (e.g., [11, 8]), only find solutions with bounded ℓ_2 norm, whereas combinatorial attacks such as [5, 25] (see also [20]) or theoretical lattice attacks [9] that can guarantee an ℓ_∞ bound are much more costly in practice, and also require exponential space. Finally, we mention that setting $\beta_\infty \ll \beta$ is very natural in the usual formulations of one-way and collision-resistant hash functions based on SIS, where collisions correspond (for example) to vectors in $\{-1, 0, 1\}^m$, and therefore have ℓ_∞ bound $\beta_\infty = 1$, but ℓ_2 bound $\beta = \sqrt{m}$. Similar gaps between β_∞ and β can easily be enforced in other applications, e.g., digital signatures [12].

LWE with small errors. In the case of LWE, we prove a general theorem offering a trade-off among several different parameters, including the size of the errors, the dimension and number of samples in the LWE problem, and the dimension of the underlying worst-case lattice problems. Here we mention just one instantiation for the case of prime modulus and uniformly distributed *binary* (i.e., 0-1) errors, and refer the reader to Section 4 and Theorem 4.6 for the more general statement and a discussion of the parameters.

Theorem 1.2 (Corollary of Theorem 4.6). *Let n and $m = n \cdot (1 + \Omega(1/\log n))$ be integers, and $q \geq n^{O(1)}$ a sufficiently large polynomially bounded (prime) modulus. Then solving* LWE *with parameters n, m, q and independent uniformly random binary errors (i.e., in $\{0, 1\}$) is at least as hard as approximating lattice problems in the worst case on $\Theta(n/\log n)$-dimensional lattices within a factor $\gamma = \tilde{O}(\sqrt{n} \cdot q)$.*

We remark that our results (see Theorem 4.6) apply to many other settings, including error vectors $\mathbf{e} \in X$ chosen from any (sufficiently large) subset $X \subseteq \{0, 1\}^m$ of binary strings, as well as error vectors with larger entries. Interestingly, our hardness result for LWE with very small errors relies on the worst-case hardness of lattice problems in dimension $n' = O(n/\log n)$, which is smaller than (but still quasi-linear in) the dimension n of the LWE problem; however, this is needed only when considering very small error vectors. Theorem 4.6 also shows that if $\mathbf{e}$ is chosen uniformly at random with entries bounded by n^ϵ (which is still much smaller than $\sqrt{n}$), then the dimension of the underlying worst-case lattice problems (and the number $m - n$ of extra samples, beyond the LWE dimension n) can be linear in n.

The restriction that the number of LWE samples $m = O(n)$ be linear in the dimension of the secret can also be relaxed slightly. But some restriction is necessary, because LWE with small errors can be solved in polynomial time when given an arbitrarily large polynomial number of samples. We focus on linear $m = O(n)$ because this is enough for most (but not all) applications in lattice cryptography, including identity-based encryption and fully homomorphic encryption, when the parameters are set appropriately. (The one exception that we know of is the security proof for pseudorandom functions [3].)

1.2 Techniques and Comparison to Related Work

Our results for SIS and LWE are technically disjoint, and all they have in common is the goal of proving hardness results for smaller values of the parameters. So, we describe our technical contributions in the analysis of these two problems separately.

SIS with small modulus. For SIS, as a warm-up, we first give a proof for a special case of the problem where the input is restricted to vectors of a special form (e.g., binary vectors). For this restricted version of SIS, we are able to give a self-reduction (from SIS to SIS) which reduces the size of the modulus. So, we can rely on previous worst-case to average-case reductions for SIS as "black boxes," resulting in an extremely simple proof. However, this simple self-reduction has some drawbacks. Beside the undesirable restriction on the SIS inputs, our the reduction is rather loose with respect to the underlying worst-case lattice approximation problem: in order to establish the hardness of SIS with small moduli q (and restricted inputs), one needs to assume the worst-case hardness of lattice problems for rather large polynomial approximation factors. (By contrast, previous hardness results for larger moduli [19, 12] only assumed hardness for quasi-linear approximation factors.) We address both drawbacks by giving a direct reduction from worst-case lattice problems to SIS with small modulus. This is our main SIS result, and it combines ideas from previous work [19, 12] with two new technical ingredients:

- All previous SIS hardness proofs [1, 7, 14, 19, 12] solved worst-case lattice problems by iteratively finding (sets of linearly independent) lattice vectors of shorter and shorter length. Our first new technical ingredient (inspired by the pioneering work of Regev [24] on LWE) is the use a different intermediate problem: instead of finding progressively shorter lattice vectors, we consider the problem of sampling lattice vectors according to Gaussian-like distributions of progressively smaller widths. To the best of our knowledge, this is the first use of Gaussian lattice sampling as an intermediate worst-case problem in the study of SIS, and it appears necessary to lower the SIS modulus below n. We mention that Gaussian lattice sampling has been used before to reduce the modulus in hardness reductions for SIS [12], but still within the framework of iteratively finding short vectors (which in [12] are used to generate fresh Gaussian samples for the reduction), which results in larger moduli $q > n$.
- The use of Gaussian lattice sampling as an intermediate problem within the SIS hardness proof yields linear combinations of several discrete Gaussian samples with adversarially chosen coefficients. Our second technical ingredient, used to analyze these linear combinations, is a new convolution theorem for discrete Gaussians (Theorem 3.3), which strengthens similar ones previously proved in [22, 6]. Here again, the strength of our new convolution theorem appears necessary to obtain hardness results for SIS with modulus smaller than n.

Our new convolution theorem may be of independent interest, and might find applications in the analysis of other lattice algorithms.

LWE with small errors. We now move to our results on LWE. For this problem, the best provably hard parameters to date were those obtained in the original paper of Regev [24], which employed Gaussian errors, and required them to have (expected) magnitude at least $\sqrt{n}$. These results were believed to be optimal due to a clever algorithm of Arora and Ge [2], which solves LWE in subexponential time when the errors are asymptotically smaller than $\sqrt{n}$. The possibility of circumventing this barrier by limiting the number of LWE samples was first suggested by Micciancio and Mol [17], who gave "sample preserving" search-to-decision reductions for LWE, and asked if LWE with small uniform errors could be proved hard when the number

of available samples is sufficiently small. Our results provide a first answer to this question, and employ concepts and techniques from the work of Peikert and Waters [23] (see also [4]) on *lossy* (trapdoor) functions. In brief, a lossy function family is an indistinguishable pair of function families $\mathcal{F}, \mathcal{L}$ such that functions in $\mathcal{F}$ are injective and those in $\mathcal{L}$ are lossy, in the sense that they map their common domain to much smaller sets, and therefore lose information about the input. As shown in [23], from the indistinguishability of $\mathcal{F}$ and $\mathcal{L}$, it follows that the families $\mathcal{F}$ and $\mathcal{L}$ are both one-way.

In Section 2 we present a generalized framework for the study of lossy function families, which does not require the functions to have trapdoors, and applies to arbitrary (not necessarily uniform) input distributions. While the techniques we use are all standard, and our definitions are minor generalizations of the ones given in [23], we believe that our framework provides a conceptual simplification of previous work, relating the relatively new notion of lossy functions to the classic security definitions of second-preimage resistance and uninvertibility.

The lossy function framework is used to prove the hardness of LWE with small uniform errors and (necessarily) a small number of samples. Specifically, we use the standard LWE problem (with large Gaussian errors) to set up a lossy function family $\mathcal{F}, \mathcal{L}$. (Similar families with trapdoors were constructed in [23, 4], but not for the parameterizations required to obtain interesting hardness results for LWE.) The indistinguishability of $\mathcal{F}$ and $\mathcal{L}$ follows directly from the hardness of the underlying LWE problem. The new hardness result for LWE (with small errors) is equivalent to the one-wayness of $\mathcal{F}$, and is proved by a relatively standard analysis of the second-preimage resistance and uninvertibility of certain subset-sum functions associated to $\mathcal{L}$.

Comparison to related work. In an independent work that was submitted concurrently with ours, Döttling and Müller-Quade [10] also used a lossyness argument to prove new hardness results for LWE. (Their work does not address the SIS problem.) At a syntactic level, they use LWE (i.e., generating matrix) notation and a new concept they call "lossy codes," while here we use SIS (i.e., parity-check matrix) notation and rely on the standard notions of uninvertible and second-preimage resistant functions. By the dual equivalence of SIS and LWE [15, 17] (see Proposition 2.9), this can be considered a purely syntactic difference, and the high-level lossyness strategy (including the lossy function family construction) used in [10] and in our work are essentially the same. However, the low-level analysis techniques and final results are quite different. The main result proved in [10] is essentially the following.

Theorem 1.3 ([10]). *Let $n, q, m = n^{O(1)}$ and $r \geq n^{1/2+\epsilon} \cdot m$ be integers, for an arbitrary small constant $\epsilon > 0$. Then the* LWE *problem with parameters n, m, q and independent uniformly distributed errors in $\{-r, \ldots, r\}^m$ is at least as hard as (quantumly) solving worst-case problems on $(n/2)$-dimensional lattices to within a factor $\gamma = n^{1+\epsilon} \cdot mq/r$.*

The contribution of [10] over previous work is to prove the hardness of LWE for *uniformly distributed* errors, as opposed to errors that follow a Gaussian distribution. Notice that the magnitude of the errors used in [10] is always at least $\sqrt{n} \cdot m$, which is substantially larger (by a factor of m) than in previous results. So, [10] makes no progress towards reducing the magnitude of the errors, which is the main goal of this paper. By contrast, our work shows the hardness of LWE for errors smaller than $\sqrt{n}$ (indeed, as small as $\{0, 1\}$), provided the number of samples is sufficiently small.

Like our work, [10] requires the number of LWE samples m to be fixed in advance (because the error magnitude r depends on m), but it allows m to be an arbitrary polynomial in n. This is possible because for the large errors $r \gg \sqrt{n}$ considered in [10], the attack of [2] runs in at least exponential time. So, in principle, it may even be possible (and is an interesting open problem) to prove the hardness of LWE with

(large) uniform errors as in [10], but for an unbounded number of samples. In our work, hardness of LWE for errors smaller than $\sqrt{n}$ is proved for a much smaller number of samples m, and this is necessary in order to avoid the subexponential time attack of [2].

While the focus of our work in on LWE with small errors, we remark that our main LWE hardness result (Theorem 4.6) can also be instantiated using large polynomial errors $r = n^{O(1)}$ to obtain any (linear) number of samples $m = \Theta(n)$. In this setting, [10] provides a much better dependency between the magnitude of the errors and the number of samples (which in [10] can be an arbitrary polynomial). This is due to substantial differences in the low-level techniques employed in [10] and in our work to analyze the statistical properties of the lossy function family. For these same reasons, even for large errors, our results seem incomparable to those of [10] because we allow for a much wider class of error distributions.

2 Preliminaries

We use uppercase roman letters F, X for sets, lowercase roman for set elements $x \in X$, bold $\mathbf{x} \in X^n$ for vectors, and calligraphic letters $\mathcal{F}, \mathcal{X}, \ldots$ for probability distributions. The support of a probability distribution $\mathcal{X}$ is denoted $[\mathcal{X}]$. The uniform distribution over a finite set X is denoted $\mathcal{U}(X)$.

Two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ are (t, ϵ)-indistinguishable if for all (probabilistic) algorithms $\mathcal{D}$ running in time at most t,

$$|\Pr[x \leftarrow \mathcal{X} : \mathcal{D}(x) \text{ accepts}] - \Pr[y \leftarrow \mathcal{Y} : \mathcal{D}(y) \text{ accepts}]| \leq \epsilon.$$

2.1 One-Way Functions

A function family is a probability distribution $\mathcal{F}$ over a set of functions $F \subseteq (X \to Y)$ with common domain X and range Y. Formally, function families are defined as distributions over bit strings (function descriptions) together with an evaluation algorithm, mapping each bitstring to a corresponding function, with possibly multiple descriptions associated to the same function. In this paper, for notational simplicity, we identify functions and their description, and unless stated otherwise, all statements about function families should be interpreted as referring to the corresponding probability distributions over function descriptions. For example, if we say that two function families $\mathcal{F}$ and $\mathcal{G}$ are indistinguishable, we mean that no efficient algorithm can distinguish between function descriptions selected according to either $\mathcal{F}$ or $\mathcal{G}$, where $\mathcal{F}$ and $\mathcal{G}$ are probability distributions over bitstrings that are interpreted as functions using the same evaluation algorithm.

A function family $\mathcal{F}$ is (t, ϵ) *collision resistant* if for all (probabilistic) algorithms $\mathcal{A}$ running in time at most t,

$$\Pr[f \leftarrow \mathcal{F}, (x, x') \leftarrow \mathcal{A}(f) : f(x) = f(x') \wedge x \neq x'] \leq \epsilon.$$

Let $\mathcal{X}$ be a probability distribution over the domain X of a function family $\mathcal{F}$. We recall the following standard security notions:

- $(\mathcal{F}, \mathcal{X})$ is (t, ϵ)-*one-way* if for all probabilistic algorithms $\mathcal{A}$ running in time at most t,

$$\Pr[f \leftarrow \mathcal{F}, x \leftarrow \mathcal{X} : \mathcal{A}(f, f(x)) \in f^{-1}(f(x))] \leq \epsilon.$$

- $(\mathcal{F}, \mathcal{X})$ is (t, ϵ)-*uninvertible* if for all probabilistic algorithms $\mathcal{A}$ running in time at most t,

$$\Pr[f \leftarrow \mathcal{F}, x \leftarrow \mathcal{X} : \mathcal{A}(f, f(x)) = x] \leq \epsilon.$$

- $(\mathcal{F}, \mathcal{X})$ is (t, ϵ)-*second preimage resistant* if for all probabilistic algorithms $\mathcal{A}$ running in time at most t,

$$\Pr[f \leftarrow \mathcal{F}, x \leftarrow \mathcal{X}, x' \leftarrow \mathcal{A}(f, x) : f(x) = f(x') \wedge x \neq x'] \leq \epsilon.$$

- $(\mathcal{F}, \mathcal{X})$ is (t, ϵ)-*pseudorandom* if the distributions $\{f \leftarrow \mathcal{F}, x \leftarrow \mathcal{X} : (f, f(x))\}$ and $\{f \leftarrow \mathcal{F}, y \leftarrow \mathcal{U}(Y) : (f, y)\}$ are (t, ϵ)-indistinguishable.

The above probabilities (or the absolute difference between probabilities, for indistinguishability) are called the *advantages* in breaking the corresponding security notions. It easily follows from the definition that if a function family is one-way with respect to any input distribution $\mathcal{X}$, then it is also uninvertible with respect to the same input distribution $\mathcal{X}$. Also, if a function family is collision resistant, then it is also second preimage resistant with respect to any efficiently samplable input distribution.

All security definitions are immediately adapted to the asymptotic setting, where we implicitly consider sequences of finite function families indexed by a security parameter. In this setting, for any security definition (one-wayness, collision resistance, etc.) we omit t, and simply say that a function is secure if for any t that is polynomial in the security parameter, it is (t, ϵ)-secure for some ϵ that is negligible in the security parameter. We say that a function family is *statistically* secure if it is (t, ϵ)-secure for some negligible ϵ and *arbitrary* t, i.e., it is secure even with respect to computationally unbounded adversaries.

The composition of function families is defined in the natural way. Namely, for any two function families with $[\mathcal{F}] \subseteq X \to Y$ and $[\mathcal{G}] \subseteq Y \to Z$, the composition $\mathcal{G} \circ \mathcal{F}$ is the function family that selects $f \leftarrow \mathcal{F}$ and $g \leftarrow \mathcal{G}$ independently at random, and outputs the function $(g \circ f) \colon X \to Z$.

2.2 Lossy Function Families

Lossy functions, introduced in [23], are usually defined in the context of trapdoor function families, where the functions are efficiently invertible with the help of some trapdoor information, and therefore injective (at least with high probability over the choice of the key). We give a more general definition of lossy function families that applies to non-injective functions and arbitrary input distributions, though we will be mostly interested in input distributions that are uniform over some set.

Definition 2.1. *Let $\mathcal{L}, \mathcal{F}$ be two probability distributions (with possibly different supports) over the same set of (efficiently computable) functions $F \subseteq X \to Y$, and let $\mathcal{X}$ be an efficiently sampleable distribution over the domain X. We say that $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a* lossy function family *if the following properties are satisfied:*

- *the distributions $\mathcal{L}$ and $\mathcal{F}$ are indistinguishable,*
- *$(\mathcal{L}, \mathcal{X})$ is uninvertible, and*
- *$(\mathcal{F}, \mathcal{X})$ is second preimage resistant.*

The uninvertibility and second preimage resistance properties can be either computational or statistical. (The definition from [23] requires both to be statistical.) We remark that uninvertible functions and second preimage resistant functions are not necessarily one-way. For example, the constant function $f(x) = 0$ is (statistically) uninvertible when $|X|$ is super-polynomial in the security parameter, and the identity function $f(x) = x$ is (statistically) second preimage resistant (in fact, even collision resistant), but neither is one-way. Still, if a function family is simultaneously uninvertible and second preimage resistant, then one-wayness easily follows.

Lemma 2.2. *Let $\mathcal{F}$ be a family of functions computable in time t'. If $(\mathcal{F}, \mathcal{X})$ is both (t, ϵ)-uninvertible and $(t + t', \epsilon')$-second preimage resistant, then it is also $(t, \epsilon + \epsilon')$-one-way.*

Proof. Let $\mathcal{A}$ be an algorithm running in time at most t and attacking the one-wayness property of $(\mathcal{F}, \mathcal{X})$. Let $f \leftarrow \mathcal{F}$ and $x \leftarrow \mathcal{X}$ be chosen at random, and compute $y \leftarrow \mathcal{A}(f, f(x))$. We want to bound the probability that $f(x) = f(y)$. We consider two cases:

- If $x = y$, then $\mathcal{A}$ breaks the uninvertibility property of $(\mathcal{F}, \mathcal{X})$.
- If $x \neq y$, then $\mathcal{A}'(f, x) = \mathcal{A}(f, f(x))$ breaks the second preimage property of $(\mathcal{F}, \mathcal{X})$.

By assumption, the probability of these two events are at most ϵ and ϵ' respectively. By the union bound, $\mathcal{A}$ breaks the one-wayness property with advantage at most $\epsilon + \epsilon'$. □

It easily follows by a simple indistinguishability argument that if $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a lossy function family, then both $(\mathcal{L}, \mathcal{X})$ and $(\mathcal{F}, \mathcal{X})$ are one-way.

Lemma 2.3. *Let $\mathcal{F}$ and $\mathcal{F}'$ be any two indistinguishable, efficiently computable function families, and let $\mathcal{X}$ be an efficiently sampleable input distribution. Then if $(\mathcal{F}, \mathcal{X})$ is uninvertible (respectively, second-preimage resistant), then $(\mathcal{F}', \mathcal{X})$ is also uninvertible (resp., second-preimage resistant). In particular, if $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a lossy function family, then $(\mathcal{L}, \mathcal{X})$ and $(\mathcal{F}, \mathcal{X})$ are both one-way.*

Proof. Assume that $(\mathcal{F}, \mathcal{X})$ is uninvertible and that there exists an efficient algorithm $\mathcal{A}$ breaking the uninvertibility property of $(\mathcal{F}', \mathcal{X})$. Then $\mathcal{F}$ and $\mathcal{F}'$ can be efficiently distinguished by the following algorithm $\mathcal{D}(f)$: choose $x \leftarrow \mathcal{X}$, compute $x' \leftarrow \mathcal{A}(f, f(x))$, and accept if $\mathcal{A}$ succeeded, i.e., if $x = x'$.

Next, assume that $(\mathcal{F}, \mathcal{X})$ is second preimage resistant, and that there exists an efficient algorithm $\mathcal{A}$ breaking the second preimage resistance property of $(\mathcal{F}', \mathcal{X})$. Then $\mathcal{F}$ and $\mathcal{F}'$ can be efficiently distinguished by the following algorithm $\mathcal{D}(f)$: choose $x \leftarrow \mathcal{X}$, compute $x' \leftarrow \mathcal{A}(f, x)$, and accept if $\mathcal{A}$ succeeded, i.e., if $x \neq x'$ and $f(x) = f(x')$.

It follows that if $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a lossy function family, then $(\mathcal{L}, \mathcal{X})$ and $(\mathcal{F}, \mathcal{X})$ are both uninvertible and second preimage resistant. Therefore, by Lemma 2.2, they are also one-way. □

The standard definition of (injective) lossy trapdoor functions [23], is usually stated by requiring the ratio $|f(X)|/|X|$ to be small. Our general definition can easily be related to the standard definition by specializing it to uniform input distributions. The next lemma gives an equivalent characterization of uninvertible functions when the input distribution is uniform.

Lemma 2.4. *Let $\mathcal{L}$ be a family of functions on a common domain X, and let $\mathcal{X} = \mathcal{U}(X)$ the uniform input distribution over X. Then $(\mathcal{L}, \mathcal{X})$ is ϵ-uninvertible (even statistically, with respect to computationally unbounded adversaries) for $\epsilon = \mathbb{E}_{f \leftarrow \mathcal{L}}[|f(X)|]/|X|$.*

Proof. Fix a function f, and choose a random input $x \leftarrow \mathcal{X}$. The best (computationally unbounded) attack on the uninvertibility of $(\mathcal{L}, \mathcal{X})$, given input f and $y = f(x)$, outputs an $x' \in X$ such that $f(x') = y$ and the probability of x' under $\mathcal{X}$ is maximized. Since $\mathcal{X}$ is the uniform distribution over X, the conditional distribution of x given y is uniform over $f^{-1}(y)$, and the attack succeeds with probability $1/|f^{-1}(y)|$. Each y is output by f with probability $|f^{-1}(y)|/|X|$. So, the success probability of the attack is

$$\sum_{y \in f(X)} \frac{|f^{-1}(y)|}{|X|} \cdot \frac{1}{|f^{-1}(y)|} = \frac{|f(X)|}{|X|}.$$

Taking the expectation over the choice of f, we get that the attacker succeeds with probability ϵ. □

We conclude this section with the observation that uninvertibility behaves as expected with respect to function composition.

Lemma 2.5. *If $(\mathcal{F}, \mathcal{X})$ is uninvertible and $\mathcal{G}$ is any family of efficiently computable functions, then $(\mathcal{G} \circ \mathcal{F}, \mathcal{X})$ is also uninvertible.*

Proof. Any inverter $\mathcal{A}$ for $\mathcal{G} \circ \mathcal{F}$ can be easily transformed into an inverter $\mathcal{A}'(f, y)$ for $(\mathcal{F}, \mathcal{X})$ that chooses $g \leftarrow \mathcal{G}$ at random, and outputs the result of running $\mathcal{A}(g \circ f, g(y))$ □

A similar statement holds also for one-wayness, under the additional assumption that $\mathcal{G}$ is second preimage resistant, but it is not needed here.

2.3 Lattices and Gaussians

An n-dimensional *lattice* of *rank* k is the set Λ of integer combinations of k linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$, i.e. $\Lambda = \left\{ \sum_{i=1}^{k} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \text{ for } i = 1, \ldots, k \right\}$. The matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ is called a *basis* for the lattice Λ. The *dual* of a (not necessarily full-rank) lattice Λ is the set $\Lambda^* = \{\mathbf{x} \in \mathrm{span}(\Lambda) : \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$. In what follows, unless otherwise specified we work with *full-rank* lattices, where $k = n$.

The ith *successive minimum* $\lambda_i(\Lambda)$ is the smallest radius r such that Λ contains i linearly independent vectors of (Euclidean) length at most r. A fundamental computational problem in the study of lattice cryptography is the approximate Shortest Independent Vectors Problem SIVP_γ, which, on input a full-rank n-dimensional lattice Λ (typically represented by a basis), asks to find n linearly independent lattice vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \Lambda$ all of length at most $\gamma \cdot \lambda_n(\Lambda)$, where $\gamma \geq 1$ is an approximation factor and is usually a function of the lattice dimension n. Another problem is the (decision version of the) approximate Shortest Vector Problem GapSVP_γ, which, on input an n-dimensional lattice Λ, asks to output "yes" if $\lambda_1(\Lambda) \leq 1$ and "no" if $\lambda_1(\Lambda) > \gamma$. (If neither is the case, any answer is acceptable.)

For a matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ of linearly independent vectors, the *Gram-Schmidt* orthogonalization $\widetilde{\mathbf{B}}$ is the matrix of vectors $\widetilde{\mathbf{b}}_i$ where $\widetilde{\mathbf{b}}_1 = \mathbf{b}_1$, and for each $i = 2, \ldots, k$, the vector $\widetilde{\mathbf{b}}_i$ is the projection of $\mathbf{b}_i$ orthogonal to $\mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. The *Gram-Schmidt* minimum of a lattice Λ is $\tilde{bl}(\Lambda) = \min_{\mathbf{B}} \|\widetilde{\mathbf{B}}\|$, where $\|\widetilde{\mathbf{B}}\| = \max_i \|\widetilde{\mathbf{b}}_i\|$ and the minimum is taken over all bases $\mathbf{B}$ of Λ. Given any basis $\mathbf{D}$ of a lattice Λ and any set $\mathbf{S}$ of linearly independent vectors in Λ, it is possible to efficiently construct a basis $\mathbf{B}$ of Λ such that $\|\widetilde{\mathbf{B}}\| \leq \|\widetilde{\mathbf{S}}\|$ (see [16]).

The *Gaussian function* $\rho_s \colon \mathbb{R}^m \to \mathbb{R}$ with parameter s is defined as $\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2 / s^2)$. When s is omitted, it is assumed to be 1. The *discrete Gaussian distribution* $D_{\Lambda + \mathbf{c}, s}$ with parameter s over a lattice coset $\Lambda + \mathbf{c}$ is the distribution that samples each element $\mathbf{x} \in \Lambda + \mathbf{c}$ with probability $\rho_s(\mathbf{x}) / \rho_s(\Lambda + \mathbf{c})$, where $\rho_s(\Lambda + \mathbf{c}) = \sum_{\mathbf{y} \in \Lambda + \mathbf{c}} \rho_s(\mathbf{y})$ is a normalization factor.

For any $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(\Lambda)$ [19] is the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$. When ϵ is omitted, it is some unspecified negligible function $\epsilon = n^{-\omega(1)}$ of the lattice dimension or security parameter n, which may vary from place to place.

We observe that the smoothing parameter satisfies the following decomposition lemma. The general case for the sum of several lattices (whose linear spans have trivial pairwise intersections) follows immediately by induction.

Lemma 2.6. *Let lattice $\Lambda = \Lambda_1 + \Lambda_2$ be the (internal direct) sum of two lattices such that $\mathrm{span}(\Lambda_1) \cap \mathrm{span}(\Lambda_2) = \{\mathbf{0}\}$, and let $\widetilde{\Lambda}_2$ be the projection of Λ_2 orthogonal to $\mathrm{span}(\Lambda_1)$. Then for any $\epsilon_1, \epsilon_2, \epsilon > 0$ such*

that $1+\epsilon = (1+\epsilon_1)(1+\epsilon_2)$*, we have*

$$\eta_\epsilon(\widetilde{\Lambda}_2) \le \eta_\epsilon(\Lambda) \le \eta_\epsilon(\Lambda_1 + \widetilde{\Lambda}_2) \le \max\{\eta_{\epsilon_1}(\Lambda_1), \eta_{\epsilon_2}(\widetilde{\Lambda}_2)\}.$$

Proof. Let Λ^*, Λ_1^* and $\widetilde{\Lambda}_2^*$ be the dual lattices of Λ, Λ_1 and $\widetilde{\Lambda}_2$, respectively. For the first inequality, notice that $\widetilde{\Lambda}_2^*$ is a sublattice of Λ^*. Therefore, $\rho_{1/s}(\widetilde{\Lambda}_2^* \setminus \{\mathbf{0}\}) \le \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\})$ for any $s > 0$, and thus $\eta_\epsilon(\widetilde{\Lambda}_2) \le \eta_\epsilon(\Lambda)$.

Next we prove that $\eta_\epsilon(\Lambda) \le \eta_\epsilon(\Lambda_1 + \widetilde{\Lambda}_2)$. It is routine to verify that we can express the dual lattice Λ^* as the sum $\Lambda^* = \widetilde{\Lambda}_1^* + \widetilde{\Lambda}_2^*$, where $\widetilde{\Lambda}_1$ is the projection of Λ_1 orthogonal to $\mathrm{span}(\Lambda_2)$, and $\widetilde{\Lambda}_1^*$ is its dual. Moreover, the projection of $\widetilde{\Lambda}_1^*$ orthogonal to $\mathrm{span}(\widetilde{\Lambda}_2^*)$ is exactly Λ_1^*. For any $\tilde{\mathbf{x}}_1 \in \widetilde{\Lambda}_1^*$, let $\mathbf{x}_1 \in \Lambda_1^*$ denote its projection orthogonal to $\mathrm{span}(\widetilde{\Lambda}_2^*)$. Then for any $s > 0$ we have

$$\begin{aligned}
\rho_{1/s}(\Lambda^*) &= \sum_{\widetilde{\mathbf{x}}_1 \in \widetilde{\Lambda}_1^*} \sum_{\widetilde{\mathbf{x}}_2 \in \widetilde{\Lambda}_2^*} \rho_{1/s}(\widetilde{\mathbf{x}}_1 + \widetilde{\mathbf{x}}_2) \\
&= \sum_{\widetilde{\mathbf{x}}_1 \in \widetilde{\Lambda}_1^*} \sum_{\widetilde{\mathbf{x}}_2 \in \widetilde{\Lambda}_2^*} \rho_{1/s}(\mathbf{x}_1) \cdot \rho_{1/s}((\widetilde{\mathbf{x}}_1 - \mathbf{x}_1) + \widetilde{\mathbf{x}}_2) \\
&= \sum_{\widetilde{\mathbf{x}}_1 \in \widetilde{\Lambda}_1^*} \rho_{1/s}(\mathbf{x}_1) \cdot \rho_{1/s}((\widetilde{\mathbf{x}}_1 - \mathbf{x}_1) + \widetilde{\Lambda}_2^*) \\
&\le \rho_{1/s}(\Lambda_1^*) \cdot \rho_{1/s}(\widetilde{\Lambda}_2^*) = \rho_{1/s}(\Lambda_1^* + \widetilde{\Lambda}_2^*) = \rho_{1/s}((\Lambda_1 + \widetilde{\Lambda}_2)^*),
\end{aligned}$$

where the inequality follows from the bound $\rho_{1/s}(\Lambda + \mathbf{c}) \le \rho_{1/s}(\Lambda)$ from [19, Lemma 2.9], and the last two equalities follow from the orthogonality of Λ_1^* and $\widetilde{\Lambda}_2^*$. This proves that $\eta_\epsilon(\Lambda) \le \eta_\epsilon(\Lambda_1 + \widetilde{\Lambda}_2)$.

Finally, for $s_1 = \eta_{\epsilon_1}(\Lambda_1)$, $s_2 = \eta_{\epsilon_2}(\widetilde{\Lambda}_2)$ and $s = \max\{s_1, s_2\}$, we have

$$\rho_{1/s}((\Lambda_1 + \widetilde{\Lambda}_2)^*) = \rho_{1/s}(\Lambda_1^*) \cdot \rho_{1/s}(\widetilde{\Lambda}_2^*) \le \rho_{1/s_1}(\Lambda_1^*) \cdot \rho_{1/s_2}(\widetilde{\Lambda}_2^*) = (1+\epsilon_1)(1+\epsilon_2) = 1+\epsilon.$$

Therefore, $\eta_\epsilon(\Lambda_1 + \widetilde{\Lambda}_2^*) \le s$. □

Using the decomposition lemma, one easily obtains known bounds on the smoothing parameter. For example, for any lattice basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, applying Lemma 2.6 repeatedly to the decomposition into the rank-1 lattices defined by each of the basis vectors yields $\eta(\mathbf{B} \cdot \mathbb{Z}^n) \le \max_i \eta(\widetilde{\mathbf{b}}_i \cdot \mathbb{Z}) = \|\widetilde{\mathbf{B}}\| \cdot \omega_n$, where $\omega_n = \eta(\mathbb{Z}) = \omega(\sqrt{\log n})$ is the smoothing parameter of the integer lattice $\mathbb{Z}$. Choosing a basis $\mathbf{B}$ achieving $\tilde{bl}(\Lambda) = \min_{\mathbf{B}} \|\widetilde{\mathbf{B}}\|$ (where the minimum is taken over all bases $\mathbf{B}$ of Λ), we get the bound $\eta(\Lambda) \le \tilde{bl}(\Lambda) \cdot \omega_n$ from [12, Theorem 3.1]. Similarly, choosing a set $\mathbf{S} \subset \Lambda$ of linearly independent vectors of length $\|\mathbf{S}\| \le \lambda_n(\Lambda)$, we get the bound $\eta(\Lambda) \le \eta(\mathbf{S} \cdot \mathbb{Z}^n) \le \|\widetilde{\mathbf{S}}\| \cdot \omega_n \le \|\mathbf{S}\| \cdot \omega_n = \lambda_n(\Lambda) \cdot \omega_n$ from [19, Lemma 3.3]. In this paper we use a further generalization of these bounds, still easily obtained from the decomposition lemma.

Corollary 2.7. *The smoothing parameter of the tensor product of any two lattices* Λ_1, Λ_2 *satisfies* $\eta(\Lambda_1 \otimes \Lambda_2) \le \tilde{bl}(\Lambda_1) \cdot \eta(\Lambda_2)$.

Proof. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ be a basis of Λ_1 achieving $\max_i \|\widetilde{\mathbf{b}}_i\| = \tilde{bl}(\Lambda_1)$, and consider the natural decomposition of $\Lambda_1 \otimes \Lambda_2$ into the sum

$$(\mathbf{b}_1 \otimes \Lambda_2) + \cdots + (\mathbf{b}_k \otimes \Lambda_2).$$

Notice that the projection of each sublattice $\mathbf{b}_i \otimes \Lambda_2$ orthogonal to the previous sublattices $\mathbf{b}_j \otimes \Lambda_2$ (for $j < i$) is precisely $\widetilde{\mathbf{b}}_i \otimes \Lambda_2$, and has smoothing parameter $\eta(\widetilde{\mathbf{b}}_i \otimes \Lambda_2) = \|\widetilde{\mathbf{b}}_i\| \cdot \eta(\Lambda_2)$. Therefore, by repeated application of Lemma 2.6, we have $\eta(\Lambda_1 \otimes \Lambda_2) \le \max_i \|\widetilde{\mathbf{b}}_i\| \cdot \eta(\Lambda_2) = \tilde{bl}(\Lambda_1) \cdot \eta(\Lambda_2)$. □

The following proposition relates the problem of sampling lattice vectors according to a Gaussian distribution to the SIVP.

Proposition 2.8 ([24], Lemma 3.17). *There is a polynomial time algorithm that, given a basis for an n-dimensional lattice Λ and polynomially many samples from $D_{\Lambda,\sigma}$ for some $\sigma \geq 2\eta(\Lambda)$, solves SIVP_γ on input lattice Λ (in the worst case over Λ, and with overwhelming probability over the choice of the lattice samples) for approximation factor $\gamma = \sigma\sqrt{n} \cdot \omega_n$.*

2.4 The SIS and LWE Functions

In this paper we are interested in two special families of functions, which are the fundamental building blocks of lattice cryptography. Both families are parametrized by three integers m, n and q, and a set $X \subseteq \mathbb{Z}^m$ of short vectors. Usually n serves as a security parameter and m and q are functions of n.

The *Short Integer Solution* function family $\mathrm{SIS}(m, n, q, X)$ is the set of all functions $f_{\mathbf{A}}$ indexed by $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with domain $X \subseteq \mathbb{Z}^m$ and range $Y = \mathbb{Z}_q^n$, defined as $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$. The *Learning With Errors* function family $\mathrm{LWE}(m, n, q, X)$ is the set of all functions $g_{\mathbf{A}}$ indexed by $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with domain $\mathbb{Z}_q^n \times X$ and range $Y = \mathbb{Z}_q^m$, defined as $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}^T\mathbf{s} + \mathbf{x} \bmod q$. Both function families are endowed with the uniform distribution over $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. We omit the set X from the notation $\mathrm{SIS}(m, n, q)$ and $\mathrm{LWE}(m, n, q)$ when clear from the context, or unimportant.

In the context of collision resistance, we sometimes write $\mathrm{SIS}(m, n, q, \beta)$ for some real $\beta > 0$, without an explicit domain X. Here the collision-finding problem is, given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, to find distinct $\mathbf{x}, \mathbf{x}' \in \mathbb{Z}^m$ such that $\|\mathbf{x} - \mathbf{x}'\| \leq \beta$ and $f_{\mathbf{A}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x}')$. It is easy to see that this is equivalent to finding a nonzero $\mathbf{z} \in \mathbb{Z}^m$ of length at most $\|\mathbf{z}\| \leq \beta$ such that $f_{\mathbf{A}}(\mathbf{z}) = \mathbf{0}$.

For other security properties (e.g., one-wayness, uninvertibility, etc.), the most commonly used classes of domains and input distributions $\mathcal{X}$ for SIS are the uniform distribution $\mathcal{U}(X)$ over the set $X = \{0, \ldots, s-1\}^m$ or $X = \{-s, \ldots, 0, \ldots, s\}^m$, and the discrete Gaussian distribution $D_{\mathbb{Z},s}^m$. Usually, this distribution is restricted to the set of short vectors $X = \{\mathbf{x} \in \mathbb{Z}^m \colon \|\mathbf{x}\| \leq s\sqrt{m}\}$, which carries all but a $2^{-\Omega(m)}$ fraction of the probability mass of $D_{\mathbb{Z},s}^m$.

For the LWE function family, the input is usually chosen according to distribution $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{X}$, where $\mathcal{X}$ is one of the SIS input distributions. This makes the SIS and LWE function families essentially equivalent, as shown in the following proposition.

Proposition 2.9 ([15, 17]). *For any n, $m \geq n + \omega(\log n)$, q, and distribution $\mathcal{X}$ over $\mathbb{Z}^m$, the $\mathrm{LWE}(m, n, q)$ function family is one-way (resp. pseudorandom, or uninvertible) with respect to input distribution $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{X}$ if and only if the $\mathrm{SIS}(m, m-n, q)$ function family is one-way (resp. pseudorandom, or uninvertible) with respect to the input distribution $\mathcal{X}$.*

In applications, the SIS function family is typically used with larger input domains X for which the functions are surjective but not injective, while the LWE function family is used with smaller domains X for which the functions are injective, but not surjective. The results in this paper are more naturally stated using the SIS function family, so we will use the SIS formulation to establish our main results, and then reformulate them in terms of the LWE function family by invoking Proposition 2.9. We also use Proposition 2.9 to reformulate known hardness results (from worst-case complexity assumptions) for LWE in terms of SIS.

Assuming the quantum worst-case hardness of standard lattice problems, Regev [24] showed that the $\mathrm{LWE}(m, n, q)$ function family is hard to invert with respect to the discrete Gaussian error distribution $D_{\mathbb{Z},\sigma}^m$ for any $\sigma > 2\sqrt{n}$. (See also [21] for a classical reduction that requires q to be exponentially large in n.

Because we are concerned with small parameters in this work, we focus mainly on the implications of the quantum reduction.)

Proposition 2.10 ([24], Theorem 3.1). *For any $m = n^{O(1)}$, integer q and real $\alpha \in (0,1)$ such that $\alpha q > 2\sqrt{n}$, there is a polynomial time quantum reduction from sampling $D_{\Lambda,\sigma}$ (for any n-dimensional lattice Λ and $\sigma > (\sqrt{2n}/\alpha)\eta(\Lambda)$) to inverting the $\mathrm{LWE}(m,n,q)$ function family on input $\mathcal{Y} = D_{\mathbb{Z}^m,\alpha q}$.*

Combining Propositions 2.8, 2.9 and 2.10, we get the following corollary.

Corollary 2.11. *For any positive m, n such that $\omega(\log n) \le m - n \le n^{O(1)}$ and $2\sqrt{n} < \sigma < q$, the $\mathrm{SIS}(m, m-n, q)$ function family is uninvertible with respect to input distribution $D^m_{\mathbb{Z},\sigma}$, under the assumption that no (quantum) algorithm can efficiently sample from a distribution statistically close to $D_{\Lambda,\sqrt{2n}q/\sigma}$.*

In particular, assuming the worst-case (quantum) hardness of $\mathrm{SIVP}_{n\omega_n q/\sigma}$ over n-dimensional lattices, the $\mathrm{SIS}(m, m-n, q)$ function family is uninvertible with respect to input distribution $D^m_{\mathbb{Z},\sigma}$.

We use the fact that LWE/SIS is not only hard to invert, but also pseudorandom. This is proved using search-to-decision reductions for those problems. The most general such reductions known to date are given in the following two theorems.

Theorem 2.12 ([17]). *For any positive m, n such that $\omega(\log n) \le m - n \le n^{O(1)}$, any positive $\sigma \le n^{O(1)}$, and any q with no divisors in the interval $((\sigma/\omega_n)^{m/k}, \sigma \cdot \omega_n)$, if $\mathrm{SIS}(m, m-n, q, D^m_{\mathbb{Z},\sigma})$ is uninvertible, then it is also pseudorandom.*

Notice that when $\sigma > \omega_n^{(m+k)/(m-k)}$, the interval $((\sigma/\omega_n)^{m/k}, \sigma \cdot \omega_n)$ is empty, and Theorem 2.12 holds without any restriction on the factorization of the modulus q.

Theorem 2.13 ([18]). *Let q have prime factorization $q = p_1^{e_1} \cdots p_k^{e_k}$ for pairwise distinct $\mathrm{poly}(n)$-bounded primes p_i with each $e_i \ge 1$, and let $0 < \alpha \le 1/\omega_n$. If $\mathrm{LWE}(m, n, q, D^m_{\mathbb{Z},\alpha q})$ is hard to invert for* all *$m(n) = n^{O(1)}$, then $\mathrm{LWE}(m', n, q, D^m_{\mathbb{Z},\alpha' q})$ is pseudorandom for any $m' = n^{O(1)}$ and*

$$\alpha' \ge \max\{\alpha, \omega_n^{1+1/\ell} \cdot \alpha^{1/\ell}, \omega_n/p_1^{e_1}, \ldots, \omega_n/p_k^{e_k}\},$$

where ℓ is an upper bound on number of prime factors $p_i < \omega_n/\alpha'$.

In this work we focus on the use of Theorem 2.12, because it guarantees pseudorandomness for the *same* value of m as for the assumed one-wayness. This feature is important for applying our results from Section 4, which guarantee one-wayness for *particular* values of m (but not necessarily all $m = n^{O(1)}$).

Corollary 2.14. *For any positive m, n, σ, q such that $\omega(\log n) \le m-n \le n^{O(1)}$ and $2\sqrt{n} < \sigma < q < n^{O(1)}$, if q has no divisors in the range $((\sigma/\omega_n)^{1+n/k}, \sigma \cdot \omega_n)$, then the $\mathrm{SIS}(m, m-n, q)$ function family is pseudorandom with respect to input distribution $D^m_{\mathbb{Z},\sigma}$, under the assumption that no (quantum) algorithm can efficiently sample (up to negligible statistical errors) $D_{\Lambda,\sqrt{2n}q/\sigma}$.*

In particular, assuming the worst-case (quantum) hardness of $\mathrm{SIVP}_{n\omega_n q/\sigma}$ on n-dimensional lattices, the $\mathrm{SIS}(m, m-n, q)$ function family is pseudorandom with respect to input distribution $D^m_{\mathbb{Z},\sigma}$.

3 Hardness of SIS with Small Modulus

We first prove a simple "success amplification" lemma for collision-finding in SIS, which says that any inverse-polynomial advantage can be amplified to essentially 1, at only the expense of a larger runtime and value of m (which will have no ill effects on our final results). Therefore, for the remainder of this section we implicitly restrict our attention to collision-finding algorithms that have overwhelming advantage.

Lemma 3.1. *For arbitrary n, q, m and $X \subseteq \mathbb{Z}^m$, suppose there exists a probabilistic algorithm $\mathcal{A}$ that has advantage $\epsilon > 0$ in collision-finding for $\mathrm{SIS}(m, n, q, X)$. Then there exists a probabilistic algorithm $\mathcal{B}$ that has advantage $1-(1-\epsilon)^t \geq 1-\exp(-\epsilon t) = 1-\exp(-n)$ in collision-finding for $\mathrm{SIS}(M = t \cdot m, n, q, X')$, where $t = n/\epsilon$ and $X' = \bigcup_{i=1}^{t}(\{0^m\}^{i-1} \times X \times \{0^m\}^{t-i})$. The runtime of $\mathcal{B}$ is essentially t times that of $\mathcal{A}$.*

Proof. The algorithm $\mathcal{B}$ simply partitions its input $\mathbf{A} \in \mathbb{Z}_q^{n \times M}$ into blocks $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and invokes $\mathcal{A}$ (with fresh random coins) on each of them, until $\mathcal{A}$ returns a valid collision $\mathbf{x}, \mathbf{x}' \in X$ for some $\mathbf{A}_i$. Then $\mathcal{B}$ returns

$$(0^{m(i-1)}, \mathbf{x}, 0^{m(t-i)}), (0^{m(i-1)}, \mathbf{x}', 0^{m(t-i)}) \in X'$$

as a collision for $\mathbf{A}$. Clearly, $\mathcal{B}$ succeeds if any call to $\mathcal{A}$ succeeds. Since all t calls to $\mathcal{A}$ are on independent inputs $\mathbf{A}_i$ and use independent coins, some call will succeed, except with $(1-\epsilon)^t$ probability. □

3.1 SIS-to-SIS Reduction

Our first proof that the $\mathrm{SIS}(m, n, q, \beta)$ function family is collision resistant for moduli q as small as $n^{1/2+\delta}$ proceeds by a reduction between SIS problems with different parameters. Previous hardness results based on worst-case lattice assumptions require the modulus q to be at least $\beta \cdot \omega(\sqrt{n \log n})$ [12, Theorem 9.2], and $\beta \geq \sqrt{n \log q}$ is needed to guarantee that a nontrivial solution exists. For such parameters, SIS is collision resistant assuming the hardness of approximating worst-case lattice problems to within $\approx \beta\sqrt{n}$ factors.

The intuition behind our proof for smaller moduli is easily explained. We reduce SIS with modulus q^c and solution bound β^c (for any constant integer $c \geq 1$) to SIS with modulus q and bound β. Then as long as $(q/\beta)^c \geq \omega(\sqrt{n \log n})$, the former problem enjoys worst-case hardness, hence so does the latter. Thus we can take $q = \beta \cdot n^\delta$ for any constant $\delta > 0$, and $c > 1/(2\delta)$. Notice, however, that the underlying approximation factor for worst-case lattice problems is $\approx \beta^c\sqrt{n} \geq n^{1/2+1/(4\delta)}$, which, while still polynomial, degrades severely as δ approaches 0. In the next subsection we give a direct reduction from worst-case lattice problems to SIS with a small modulus, which does not have this drawback.

The above discussion is formalized in the following proposition. For technical reasons, we prove that $\mathrm{SIS}(m, n, q, X)$ is collision resistant assuming that the domain X has the property that all SIS solutions $\mathbf{z} \in (X - X) \setminus \{\mathbf{0}\}$ satisfy $\gcd(\mathbf{z}, q) = 1$. This restriction is satisfied in many (but not all) common settings, e.g., when $q > \beta$ is prime, or when $X \subseteq \{0,1\}^m$ is a set of binary vectors.

Proposition 3.2. *Let n, q, m, β and $X \subseteq \mathbb{Z}^m$ be such that $\gcd(\mathbf{x} - \mathbf{x}', q) = 1$ and $\|\mathbf{x} - \mathbf{x}'\| \leq \beta$ for any distinct $\mathbf{x}, \mathbf{x}' \in X$. For any positive integer c, there is a deterministic reduction from collision-finding for $\mathrm{SIS}(m^c, n, q^c, \beta^c)$ to collision-finding for $\mathrm{SIS}(m, n, q, X)$ (in both cases, with overwhelming advantage). The reduction runs in time polynomial in its input size, and makes fewer than m^c calls to its oracle.*

Proof. Let $\mathcal{A}$ be an efficient algorithm that finds a collision for $\mathrm{SIS}(m, n, q, X)$ with overwhelming advantage. We use it to find a nonzero solution for $\mathrm{SIS}(m^c, n, q^c, \beta^c)$. Let $\mathbf{A} \in \mathbb{Z}_{q^c}^{n \times m^c}$ be an input SIS instance. Partition the columns of $\mathbf{A}$ into m^{c-1} blocks $\mathbf{A}_i \in \mathbb{Z}_{q^c}^{n \times m}$, and for each one, invoke $\mathcal{A}$ to find a collision modulo q, i.e., a pair of distinct vectors $\mathbf{x}_i, \mathbf{x}_i' \in X$ such that $\mathbf{A}_i\mathbf{z}_i = \mathbf{0} \bmod q$, where $\mathbf{z}_i = \mathbf{x}_i - \mathbf{x}_i'$ and $\|\mathbf{z}_i\| \leq \beta$.

For each i, since $\gcd(\mathbf{z}_i, q) = 1$ and $\mathbf{A}_i\mathbf{z}_i = \mathbf{0} \bmod q$, the vector $\mathbf{a}'_i = (\mathbf{A}_i\mathbf{z}_i)/q \in \mathbb{Z}_{q^{c-1}}^n$ is uniformly random, even after conditioning on $\mathbf{z}_i$ and $\mathbf{A}_i \bmod q$. So, the matrix $\mathbf{A}' \in \mathbb{Z}_{q^{c-1}}^{n \times m^{c-1}}$ made up of all these columns is uniformly random. By induction on c, using $\mathcal{A}$ we can find a nonzero solution $\mathbf{z}' \in \mathbb{Z}^{m^{c-1}}$ such that $\mathbf{A}'\mathbf{z}' = \mathbf{0} \bmod q^{c-1}$ and $\|\mathbf{z}'\| \leq \beta^{c-1}$. Then it is easy to verify that a nonzero solution for the original instance $\mathbf{A}$ is given by $\mathbf{z} = (z'_1 \cdot \mathbf{z}_1, \ldots, z'_{m^{c-1}} \cdot \mathbf{z}_{m^{c-1}}) \in \mathbb{Z}^{m^c}$, and that $\|\mathbf{z}\| \leq \|\mathbf{z}'\| \cdot \max_i \|\mathbf{z}_i\| \leq \beta^c$. Finally, the total number of calls to $\mathcal{A}$ is $\sum_{i=0}^{c-1} m^i < m^c$, as claimed. □

3.2 Direct Reduction

As mentioned above, the large worst-case approximation factor associated with the use of Proposition 3.2 is undesirable, as is (to a lesser extent) the restriction that $\gcd(X - X, q) = 1$. To eliminate these drawbacks, we next give a direct proof that SIS is collision resistant for small q, based on the assumed hardness of worst-case lattice problems. The underlying approximation factor for these problems can be as small as $\tilde{O}(\beta\sqrt{n})$, which matches the best known factors obtained by previous proofs (which require a larger modulus q). Our new proof combines ideas from [19, 12] and Proposition 3.2, as well as a new convolution theorem for discrete Gaussians which strengthens similar ones previously proved in [22, 6].

Our proof of the convolution theorem is substantially different and, we believe, technically simpler than the prior ones. In particular, it handles the sum of many Gaussian samples all at once, whereas previous proofs used induction from a base case of two samples. With the inductive approach, it is technically complex to verify that all the intermediate Gaussian parameters (which involve harmonic means) satisfy the hypotheses. Moreover, the intermediate parameters can depend on the order in which the samples are added in the induction, leading to unnecessarily strong hypotheses on the original parameters.

Theorem 3.3. *Let Λ be an n-dimensional lattice, $\mathbf{z} \in \mathbb{Z}^m$ a nonzero integer vector, $s_i \geq \sqrt{2}\|\mathbf{z}\|_\infty \cdot \eta(\Lambda)$, and $\Lambda + \mathbf{c}_i$ arbitrary cosets of Λ for $i = 1, \ldots, m$. Let $\mathbf{y}_i$ be independent vectors with distributions $D_{\Lambda+\mathbf{c}_i, s_i}$, respectively. Then the distribution of $\mathbf{y} = \sum_i z_i\mathbf{y}_i$ is statistically close to $D_{Y,s}$, where $Y = \gcd(\mathbf{z})\Lambda + \mathbf{c}$, $\mathbf{c} = \sum_i z_i\mathbf{c}_i$, and $s = \sqrt{\sum_i (z_i s_i)^2}$.*

In particular, if $\gcd(\mathbf{z}) = 1$ and $\sum_i z_i\mathbf{c}_i \in \Lambda$, then $\mathbf{y}$ is distributed statistically close to $D_{\Lambda,s}$.

Proof. First we verify that the support of $\mathbf{y}$ is

$$\sum_i z_i(\Lambda + \mathbf{c}_i) = \sum_i z_i\Lambda + \sum_i z_i \cdot \mathbf{c}_i = \gcd(\mathbf{z})\Lambda + \sum_i z_i \cdot \mathbf{c}_i = Y.$$

So it remains to prove that each $\mathbf{y} \in Y$ has probability (nearly) proportional to $\rho_s(\mathbf{y})$.

For the remainder of the proof we use the following convenient scaling. Define the diagonal matrices $\mathbf{S} = \mathrm{diag}(s_1, \ldots, s_m)$ and $\mathbf{S}' = \mathbf{S} \otimes \mathbf{I}_n$, and the mn-dimensional lattice $\Lambda' = \bigoplus_i (s_i^{-1}\Lambda) = (\mathbf{S}')^{-1} \cdot \Lambda^{\oplus m}$, where $\bigoplus$ denotes the (external) direct sum of lattices and $\Lambda^{\oplus m} = \mathbb{Z}^m \otimes \Lambda$ is the direct sum of m copies of Λ. Then by independence of the $\mathbf{y}_i$, it can be seen that $\mathbf{y}' = (\mathbf{S}')^{-1} \cdot (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ has discrete Gaussian distribution $D_{\Lambda'+\mathbf{c}'}$ (with parameter 1), where $\mathbf{c}' = (\mathbf{S}')^{-1} \cdot (\mathbf{c}_1, \ldots, \mathbf{c}_m)$.

The output vector $\mathbf{y} = \sum_i z_i\mathbf{y}_i$ can be expressed, using the mixed-product property for Kronecker products, as

$$\mathbf{y} = (\mathbf{z}^T \otimes \mathbf{I}_n) \cdot (\mathbf{y}_1, \ldots, \mathbf{y}_m) = (\mathbf{z}^T \otimes \mathbf{I}_n) \cdot \mathbf{S}' \cdot \mathbf{y}' = ((\mathbf{z}^T\mathbf{S}) \otimes \mathbf{I}_n) \cdot \mathbf{y}'.$$

So, letting $\mathbf{Z} = ((\mathbf{z}^T\mathbf{S}) \otimes \mathbf{I}_n)$, we want to prove that the distribution of $\mathbf{y} \sim \mathbf{Z} \cdot D_{\Lambda'+\mathbf{c}'}$ is statistically close to $D_{Y,s}$.

Fix any vectors $\mathbf{x}' \in \Lambda' + \mathbf{c}'$ and $\bar{\mathbf{y}} = \mathbf{Z}\mathbf{x}' \in Y$, and define the proper sublattice

$$L = \{\mathbf{v} \in \Lambda' \colon \mathbf{Z}\mathbf{v} = \mathbf{0}\} = \Lambda' \cap \ker(\mathbf{Z}) \subsetneq \Lambda'.$$

It is immediate to verify that the set of all $\mathbf{y}' \in \Lambda' + \mathbf{c}'$ such that $\mathbf{Z}\mathbf{y}' = \bar{\mathbf{y}}$ is $(\Lambda' + \mathbf{c}') \cap \ker(\mathbf{Z}) = L + \mathbf{x}'$. Let $\mathbf{x}$ be orthogonal projection of $\mathbf{x}'$ onto $\ker(\mathbf{Z}) \supset L$. Then we have

$$\Pr[\mathbf{y} = \bar{\mathbf{y}}] = \frac{\rho(L + \mathbf{x}')}{\rho(\Lambda' + \mathbf{c}')} = \rho(\mathbf{x}' - \mathbf{x}) \cdot \frac{\rho(L + \mathbf{x})}{\rho(\Lambda' + \mathbf{c}')}.$$

Below we show that $\eta(L) \leq 1$, which implies that $\rho(L + \mathbf{x})$ is essentially the same for all values of $\mathbf{x}'$, and hence for all $\bar{\mathbf{y}}$. Therefore, we just need to analyze $\rho(\mathbf{x}' - \mathbf{x})$.

Since $\mathbf{Z}^T$ is an orthogonal basis for $\ker(\mathbf{Z})^\perp$, each of whose columns has Euclidean norm $s = (\sum_i (z_i s_i)^2)^{1/2}$, we have $\mathbf{x}' - \mathbf{x} = (\mathbf{Z}^T\mathbf{Z}\mathbf{x}')/s^2$, and

$$\|\mathbf{x}' - \mathbf{x}\|^2 = \langle \mathbf{x}', \mathbf{Z}^T\mathbf{Z}\mathbf{x}' \rangle / s^2 = \|\mathbf{Z}\mathbf{x}'\|^2/s^2 = (\|\bar{\mathbf{y}}\|/s)^2.$$

Therefore, $\rho(\mathbf{x}' - \mathbf{x}) = \rho_s(\bar{\mathbf{y}})$, and so $\Pr[\mathbf{y} = \bar{\mathbf{y}}]$ is essentially proportional to $\rho_s(\bar{\mathbf{y}})$, i.e., the statistical distance between $\mathbf{y}$ and $D_{Y,s}$ is negligible.

It remains to bound the smoothing parameter of L. Consider the m-dimensional integer lattice $Z = \mathbb{Z}^m \cap \ker(\mathbf{z}^T) = \{\mathbf{v} \in \mathbb{Z}^m : \langle \mathbf{z}, \mathbf{v} \rangle = 0\}$. Because $(Z \otimes \Lambda) \subseteq (\mathbb{Z}^m \otimes \Lambda)$ and $\mathbf{S}^{-1}Z \subset \ker(\mathbf{z}^T\mathbf{S})$, it is straightforward to verify from the definitions that

$$(\mathbf{S}')^{-1} \cdot (Z \otimes \Lambda) = ((\mathbf{S}^{-1}Z) \otimes \Lambda)$$

is a sublattice of L. It follows from Corollary 2.7 and by scaling that

$$\eta(L) \leq \eta((\mathbf{S}')^{-1} \cdot (Z \otimes \Lambda)) \leq \eta(\Lambda) \cdot \tilde{bl}(Z)/\min s_i.$$

Finally, $\tilde{bl}(Z) \leq \min\{\|\mathbf{z}\|, \sqrt{2}\|\mathbf{z}\|_\infty\}$ because Z has a full-rank set of vectors $z_i \cdot \mathbf{e}_j - z_j \cdot \mathbf{e}_i$, where index i minimizes $|z_i| \neq 0$, and j ranges over $\{1, \ldots, m\} \setminus \{i\}$. By assumption on the s_i, we have $\eta(L) \leq 1$ as desired, and the proof is complete. □

Remark 3.4. Although we will not need it in this work, we note that the statement and proof of Theorem 3.3 can be adapted to the case where the $\mathbf{y}_i$ respectively have *non-spherical* discrete Gaussian distributions $D_{\Lambda_i + \mathbf{c}_i, \sqrt{\Sigma_i}}$ with positive definite "covariance" parameters $\Sigma_i \in \mathbb{R}^{n \times n}$, over cosets of possibly different lattices Λ_i. (See [22] for a formal definition of these distributions.)

In this setting, by scaling Λ_i and Σ_i we can assume without loss of generality that $\mathbf{z} = (1, 1, \ldots, 1)$. The theorem statement says that $\mathbf{y}$'s distribution is close to a discrete Gaussian (over an appropriate lattice coset) with covariance parameter $\Sigma = \sum \Sigma_i$, under mild assumptions on $\sqrt{\Sigma_i}$. In the proof we simply let $\mathbf{S}'$ be the block-diagonal matrix with the $\sqrt{\Sigma_i}$ as its diagonal blocks, let $\Lambda' = (\mathbf{S}')^{-1} \cdot \bigoplus_i \Lambda_i$, and let $\mathbf{Z} = (\mathbf{z}^T \otimes \mathbf{I}_n) \cdot \mathbf{S}' = [\sqrt{\Sigma_1} \mid \cdots \mid \sqrt{\Sigma_m}]$. Then the only technical difference is in bounding the smoothing parameter of L.

The convolution theorem implies the following simple but useful lemma, which shows how to convert samples having a broad range of parameters into ones having parameters in a desired narrow range.

Lemma 3.5. *There is an efficient algorithm which, given a basis $\mathbf{B}$ of some lattice Λ, some $R \geq \sqrt{2}$ and samples $(\mathbf{y}_i, s_i)$ where each $s_i \in [\sqrt{2}, R] \cdot \eta(\Lambda)$ and each $\mathbf{y}_i$ has distribution D_{Λ, s_i}, with overwhelming probability outputs a sample $(\mathbf{y}, s)$ where $s \in [R, \sqrt{2}R] \cdot \eta(\Lambda)$ and $\mathbf{y}$ has distribution statistically close to $D_{\Lambda,s}$.*

Proof. Let $\omega_n = \omega(\sqrt{\log n})$ satisfy $\omega_n \leq \sqrt{n}$. The algorithm draws $2n^2$ input samples, and works as follows: if at least n^2 of the samples have parameters $s_i \leq R \cdot \eta(\Lambda)/(\sqrt{n} \cdot \omega_n)$, then with overwhelming probability they all have lengths bounded by $R \cdot \eta(\Lambda)/\omega_n$ and they include n linearly independent vectors. Using such vectors we can construct a basis $\mathbf{S}$ such that $\|\widetilde{\mathbf{S}}\| \leq R \cdot \eta(\Lambda)/\omega_n$, and with the sampling algorithm of [12, Theorem 4.1] we can generate samples having parameter $R \cdot \eta(\Lambda)$.

Otherwise, at least n^2 of the samples $(\mathbf{y}_i, s_i)$ have parameters $s_i \geq \max\{R/n, \sqrt{2}\} \cdot \eta(\Lambda)$. Then by summing an appropriate subset of those $\mathbf{y}_i$, by the convolution theorem we can obtain a sample having parameter in the desired range. □

The next lemma is the heart of our reduction. The novel part, corresponding to the properties described in the second item, is a way of using a collision-finding oracle to reduce the Gaussian width of samples drawn from a lattice. The first item corresponds to the guarantees provided by previous reductions.

Lemma 3.6. *Let m, n be integers, $S = \{\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\} \mid \|\mathbf{z}\| \leq \beta \wedge \|\mathbf{z}\|_\infty \leq \beta_\infty\}$ for some real $\beta \geq \beta > 0$, and q an integer modulus with at most $\mathrm{poly}(n)$ integer divisors less than β_∞. There is a probabilistic polynomial time reduction that, on input any basis $\mathbf{B}$ of a lattice Λ and sufficiently many samples $(\mathbf{y}_i, s_i)$ where $s_i \geq \sqrt{2}q \cdot \eta(\Lambda)$ and $\mathbf{y}_i$ has distribution D_{Λ,s_i}, and given access to an $\mathrm{SIS}(m, n, q, S)$ oracle (that finds collisions $\mathbf{z} \in S$ with nonnegligible probability) outputs (with overwhelming probability) a sample $(\mathbf{y}, s)$ with $\min s_i/q \leq s \leq (\beta/q) \cdot \max s_i$, and $\mathbf{y} \in \Lambda$ such that:*

- $\mathbb{E}[\|\mathbf{y}\|] \leq (\beta\sqrt{n}/q) \cdot \max s_i$, *and for any subspace $H \subset \mathbb{R}^n$ of dimension at most $n-1$, with probability at least $1/10$ we have $\mathbf{y} \notin H$.*
- *Moreover, if each $s_i \geq \sqrt{2}\beta_\infty q \cdot \eta(\Lambda)$, then the distribution of $\mathbf{y}$ is statistically close to $D_{\Lambda,s}$*

Proof. Let $\mathcal{A}$ be the collision-finding oracle. Without loss of generality, we can assume that whenever $\mathcal{A}$ outputs a valid collision $\mathbf{z} \in S$, we have that $\gcd(\mathbf{z})$ divides q. This is so because for any integer vector $\mathbf{z}$, if $\mathbf{Az} = \mathbf{0} \bmod q$ then also $\mathbf{A}((g/d)\mathbf{z}) = \mathbf{0} \bmod q$, where $d = \gcd(\mathbf{z})$ and $g = \gcd(d, q)$. Moreover, $(g/d)\mathbf{z} \in S$ holds true and $\gcd((g/d)\mathbf{z}) = \gcd(\mathbf{z}, q)$ divides q. Let d be such that $\mathcal{A}$ outputs, with non-negligible probability, a valid collision $\mathbf{z}$ satisfying $\gcd(\mathbf{z}) = d$. Such a d exists because $\gcd(\mathbf{z})$ is bounded by β_∞ and divides q, so by assumption there are only polynomially many possible values of d. Let $q' = q/d$, which is an integer. By increasing m and using standard amplification techniques, we can make the probability that $\mathcal{A}$ outputs such a collision (satisfying $\mathbf{z} \in S$, $\mathbf{Az} = \mathbf{0} \pmod q$ and $\gcd(\mathbf{z}) = d$) exponentially close to 1.

Let $(\mathbf{y}_i, s_i)$ for $i = 1, \ldots, m$ be input samples, where $\mathbf{y}_i$ has distribution D_{Λ,s_i}. Write each $\mathbf{y}_i$ as $\mathbf{y}_i = \mathbf{B}\mathbf{a}_i \bmod q'\Lambda$ for $\mathbf{a}_i \in \mathbb{Z}_{q'}^n$. Since $s_i \geq q'\eta(\Lambda)$ the distribution of $\mathbf{a}_i$ is statistically close to uniform over $\mathbb{Z}_{q'}^n$. Let $\mathbf{A} = [\mathbf{a}_1 \mid \cdots \mid \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$, and choose $\mathbf{A'} \in \mathbb{Z}_d^{n \times m}$ uniformly at random. Since $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_{q'}^{n \times m}$, the matrix $\mathbf{A} + q'\mathbf{A'}$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$. Call the oracle $\mathcal{A}$ on input $\mathbf{A} + q'\mathbf{A'}$, and obtain (with overwhelming probability) a nonzero $\mathbf{z} \in S$ with $\gcd(\mathbf{z}) = d$, $\|\mathbf{z}\| \leq \beta$, $\|\mathbf{z}\|_\infty \leq \beta_\infty$ and $(\mathbf{A} + q'\mathbf{A'})\mathbf{z} = \mathbf{0} \bmod q$. Notice that $q'\mathbf{A'}\mathbf{z} = q\mathbf{A'}(\mathbf{z}/d) = \mathbf{0} \bmod q$ because $(\mathbf{z}/d)$ is an integer vector. Therefore $\mathbf{Az} = \mathbf{0} \bmod q$. Finally, the reduction outputs $(\mathbf{y}, s)$, where $\mathbf{y} = \sum_i z_i\mathbf{y}_i/q$ and $s = \sqrt{\sum_i (s_i z_i)^2}/q$. Notice that $z_i\mathbf{y}_i \in q\Lambda + \mathbf{B}(z_i\mathbf{a}_i)$ because $\gcd(\mathbf{z}) = d$, so $\mathbf{y} \in \Lambda$.

Notice that s satisfies the stated bounds because $\mathbf{z}$ is a nonzero integer vector. We next analyze the distribution of $\mathbf{y}$. For any fixed $\mathbf{a}_i$, the conditional distribution of each $\mathbf{y}_i$ is $D_{q'\Lambda + \mathbf{B}\mathbf{a}_i, s_i}$, where $s_i \geq \sqrt{2}\eta(q'\Lambda)$. The claim on $\mathbb{E}[\|\mathbf{y}\|]$ then follows from [19, Lemma 2.11 and Lemma 4.3] and Hölder's inequality. The claim on the probability that $\mathbf{y} \notin H$ was initially shown in the preliminary version of [19]; see also [24, Lemma 3.15].

Now assume that $s_i \geq \sqrt{2}\beta_\infty q \cdot \eta(\Lambda) \geq \sqrt{2}\|\mathbf{z}\|_\infty \cdot \eta(q'\Lambda)$ for all i. By Theorem 3.3 the distribution of $\mathbf{y}$ is statistically close to $D_{Y/q,s}$ where $Y = \gcd(\mathbf{z}) \cdot q'\Lambda + \mathbf{B}(\mathbf{Az})$. Using $\mathbf{Az} = \mathbf{0} \bmod q$ and $\gcd(\mathbf{z}) = d$, we get $Y = q\Lambda$. Therefore $\mathbf{y}$ has distribution statistically close to $D_{\Lambda,s}$, as claimed. □

Building on Lemma 3.6, our next lemma shows that for any $q \geq \beta \cdot n^{\Omega(1)}$, a collision-finding oracle can be used to obtain Gaussian samples of width close to $2\beta\beta_\infty \cdot \eta(\Lambda)$.

Lemma 3.7. *Let m, n, q, S as in Lemma 3.6, and also assume $q/\beta \geq n^\delta$ for some constant $\delta > 0$. There is an efficient reduction that, on input any basis $\mathbf{B}$ of an n-dimensional lattice Λ, an upper bound $\eta \geq \eta(\Lambda)$, and given access to an $\mathrm{SIS}(m, n, q, S)$ oracle (finding collisions $\mathbf{z} \in S$ with nonnegligible probability), outputs (with overwhelming probability) a sample $(\mathbf{y}, s)$ where $\sqrt{2}\beta_\infty \cdot \eta \leq s \leq 2\beta_\infty\beta \cdot \eta$ and $\mathbf{y}$ has distribution statistically close to $D_{\Lambda,s}$.*

Proof. By applying the LLL basis reduction algorithm [13] to the basis $\mathbf{B}$, we can assume without loss of generality that $\|\widetilde{\mathbf{B}}\| \leq 2^n \cdot \eta(\Lambda)$. Let ω_n be an arbitrary function in n satisfying $\omega_n = \omega(\sqrt{\log n})$ and $\omega_n \leq \sqrt{n}/2$.

The main procedure, described below, produces samples having parameters in the range $[1, q] \cdot \sqrt{2}\beta_\infty \cdot \eta$. On these samples we run the procedure from Lemma 3.5 (with $R = \sqrt{2}\beta_\infty q \cdot \eta$) to obtain samples having parameters in the range $[\sqrt{2}, 2] \cdot \beta_\infty q \cdot \eta$. Finally, we invoke the reduction from Lemma 3.6 on those samples to obtain a sample satisfying the conditions in the Lemma statement.

The main procedure works in a sequence of phases $i = 0, 1, 2, \ldots$. In phase i, the input is a basis $\mathbf{B}_i$ of Λ, where initially $\mathbf{B}_0 = \mathbf{B}$. The basis $\mathbf{B}_i$ is used in the discrete Gaussian sampling algorithm of [12, Theorem 4.1] to produce samples $(\mathbf{y}, s_i)$, where $s_i = \max\{\|\widetilde{\mathbf{B}}_i\| \cdot \omega_n, \sqrt{2}\beta_\infty\eta\} \geq \sqrt{2}\beta_\infty\eta$ and $\mathbf{y}_i$ has distribution statistically close to D_{Λ,s_i}. Phase i either manages to produce a sample $(\mathbf{y}, s)$ with s in the desired range $[1, q] \cdot \sqrt{2}\beta_\infty\eta$, or it produces a new basis $\mathbf{B}_{i+1}$ for which $\|\widetilde{\mathbf{B}}_{i+1}\| \leq \|\widetilde{\mathbf{B}}_i\|/2$, which is the input to the next phase. The number of phases before termination is clearly polynomial in n, by hypothesis on $\mathbf{B}$.

If $\|\widetilde{\mathbf{B}}_i\| \cdot \omega_n \leq \sqrt{2}q\beta_\infty\eta$, then this already gives samples with $s_i \in [1, q]\sqrt{2}\beta_\infty\eta$ in the desired range, and we can terminate the main phase. So, we may assume that $s_i = \|\widetilde{\mathbf{B}}_i\| \cdot \omega_n \geq \sqrt{2}q\beta_\infty\eta$. Each phase i proceeds in some constant $c \geq 1/\delta$ number of sub-phases $j = 1, 2, \ldots, c$, where the inputs to the first sub-phase are the samples $(\mathbf{y}, s_i)$ generated as described above. We recall that these samples satisfy $s_i \geq \sqrt{2}q\beta_\infty\eta$. The same will be true for the samples passed as input to all other subsequent subphases. So, each subphase receives as input samples $(\mathbf{y}, s)$ satisfying all the hypotheses of Lemma 3.6, and we can run the reduction from that lemma to generate new samples $(\mathbf{y}', s')$ having parameters s' bounded from above by $s_i \cdot (\beta/q)^j$, and from below by $\sqrt{2}\beta_\infty\eta$. If any of the produces samples satisfies $s' \leq q\sqrt{2}\beta_\infty\eta$, then we can terminate the main procedure with $(\mathbf{y}', s')$ as output. Otherwise, all samples produced during the subphase satisfy $s' > q\sqrt{2}\beta_\infty\eta$, and they can be passed as input to the next sub-phase. Notice that the total runtime of all the sub-phases is $\mathrm{poly}(n)^c$, because each invocation of the reduction from Lemma 3.6 relies on $\mathrm{poly}(n)$ invocations of the reduction in the previous sub-phase; this is why we need to limit the number of sub-phases to a constant c.

If phase i ends up running all its sub-phases without ever finding a sample with $s' \in [1, q]\sqrt{2}\beta_\infty\eta$, then it has produced samples whose parameters are bounded by $(\beta/q)^c \leq s_i \leq s_i/\sqrt{n}$. It uses n^2 of these samples, which with overwhelming probability have lengths all bounded by $s_i/\sqrt{n}$, and include n linearly independent vectors. It transforms those vectors into a basis $\mathbf{B}_{i+1}$ with $\|\widetilde{\mathbf{B}}_{i+1}\| \leq s_i/\sqrt{n} \leq \|\widetilde{B}\|_i\omega_n/\sqrt{n} \leq \|\widetilde{\mathbf{B}}_i\|/2$, as input to the next phase. □

We can now prove our main theorem, reducing worst-case lattice problems with $\max\{1, \beta\beta_\infty/q\} \cdot \tilde{O}(\beta\sqrt{n})$ approximation factors to SIS, when $q \geq \beta \cdot n^{\Omega(1)}$.

Theorem 3.8. *Let m, n be integers, $S = \{\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\} \mid \|\mathbf{z}\| \leq \beta \wedge \|\mathbf{z}\|_\infty \leq \beta_\infty\}$ for some real $\beta \geq \beta_\infty > 0$, and $q \geq \beta \cdot n^{\Omega(1)}$ be an integer modulus with at most $\mathrm{poly}(n)$ integer divisors less than β_∞. For some $\gamma = \max\{1, \beta\beta_\infty/q\} \cdot O(\beta\sqrt{n})$, there is an efficient reduction from $\mathrm{SIVP}^\eta_\gamma$ (and hence also from standard $\mathrm{SIVP}_{\gamma \cdot \omega_n}$) on n-dimensional lattices to S-collision finding for $\mathrm{SIS}(m, n, q)$ with non-negligible advantage.*

Proof. Given an input basis $\mathbf{B}$ of a lattice Λ, we can apply the LLL algorithm to obtain a 2^n-approximation to $\eta(\Lambda)$, and by scaling we can assume that $\eta(\Lambda) \in [1, 2^n]$. For $i = 1, \ldots, n$, we run the procedure described below for each hypothesized upper bound $\eta_i = 2^i$ on $\eta(\Lambda)$. Each call to the procedure either fails, or returns a set of linearly independent vectors in Λ whose lengths are all bounded by $(\gamma/2) \cdot \eta_i$. We return the first such obtained set (i.e., for the minimal value of i). As we show below, as long as $\eta_i \geq \eta(\Lambda)$ the procedure returns a set of vectors with overwhelming probability. Since one $\eta_i \in [1, 2) \cdot \eta(\Lambda)$, our reduction solves $\mathrm{SIVP}^\eta_\gamma$ with overwhelming probability, as claimed.

The procedure invokes the reduction from Lemma 3.7 with $\eta = \eta_i$ to obtain samples with parameters in the range $[\sqrt{2}\beta_\infty, \sqrt{2}\beta\beta_\infty] \cdot \eta$. On these samples we run the procedure from Lemma 3.5 with $R = \max\{\sqrt{2}q, \sqrt{2}\beta\beta_\infty\}$ to obtain samples having parameters in the range $[R, \sqrt{2}R] \cdot \eta$. On such samples we repeatedly run (using independent samples each time) the reduction from Lemma 3.6. After enough runs, we obtain with overwhelming probability a set of linearly independent lattice vectors all having lengths at most $(\gamma/2) \cdot \eta$, as required. □

4 Hardness of LWE with Small Uniform Errors

In this section we prove the hardness of inverting the LWE function even when the error vectors have very small entries, provided the number of samples is sufficiently small. We proceed similarly to [23, 4], by using the LWE assumption (for discrete Gaussian error) to construct a lossy family of functions with respect to a uniform distribution over small inputs. However, the parameterization we obtain is different from those in [23, 4], allowing us to obtain *pseudorandomness* of LWE under *very small* (e.g., binary) inputs, for a number of LWE samples that exceeds the LWE dimension.

Our results and proofs are more naturally formulated using the SIS function family. So, we will first study the problem in terms of SIS, and then reformulate the results in terms of LWE using Proposition 2.9. We recall that the main difference between this section and Section 3, is that here we consider parameters for which the resulting functions are essentially injective, or more formally, statistically second-preimage resistant. The following lemma gives sufficient conditions that ensure this property.

Lemma 4.1. *For any integers m, k, q, s and set $X \subseteq [s]^m$, the function family $\mathrm{SIS}(m, k, q)$ is (statistically) ϵ-second preimage resistant with respect to the uniform input distribution $\mathcal{U}(X)$ for $\epsilon = |X| \cdot (s'/q)^k$, where s' is the largest factor of q smaller than s.*

Proof. Let $\mathbf{x} \leftarrow \mathcal{U}(X)$ and $\mathbf{A} \leftarrow \mathrm{SIS}(m, k, q)$ be chosen at random. We want to evaluate the probability that there exists an $\mathbf{x}' \in X \setminus \{\mathbf{x}\}$ such that $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \pmod q$, or, equivalently, $\mathbf{A}(\mathbf{x} - \mathbf{x}') = \mathbf{0} \pmod q$. Fix any two distinct vectors $\mathbf{x}, \mathbf{x}' \in X$ and let $\mathbf{z} = \mathbf{x} - \mathbf{x}'$. The vector $\mathbf{A}\mathbf{z} \pmod q$ is distributed uniformly at random in $(d\mathbb{Z}/q\mathbb{Z})^k$, where $d = \gcd(q, z_1, \ldots, z_m)$. All coordinates of $\mathbf{z}$ are in the range $z_i \in \{-(s-1), \ldots, (s-1)\}$, and at least one of them is nonzero. Therefore, d is at most s' and $|d\mathbb{Z}^k_q| = (q/d)^k \geq (q/s')^k$. By union bound (over $\mathbf{x}' \in X \setminus \{\mathbf{x}\}$) for any $\mathbf{x}$, the probability that there is a second preimage $\mathbf{x}'$ is at most $(|X| - 1)(s'/q)^k$. □

We remark that, as shown in Section 3, even for parameter settings that do not fall within the range specified in Lemma 4.1, $\mathrm{SIS}(m, k, q)$ is collision resistant, and therefore also (computationally) second-preimage-resistant. This is all that is needed in the rest of this section. However, when $\mathrm{SIS}(m, k, q)$ is not statistically second-preimage resistant, the one-wayness proof that follows (see Theorem 4.5) is not very interesting: typically, in such settings, $\mathrm{SIS}(m, k, q)$ is also statistically uninvertible, and the one-wayness of $\mathrm{SIS}(m, k, q)$ directly follows from Lemma 2.2. So, below we focus on parameter settings covered by Lemma 4.1.

We prove the one-wayness of $\mathcal{F} = \mathrm{SIS}(m, k, q, X)$ with respect to the uniform input distribution $\mathcal{X} = \mathcal{U}(X)$ by building a lossy function family $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ where $\mathcal{L}$ is an auxiliary function family that we will prove to be uninvertible and computationally indistinguishable from $\mathcal{F}$. The auxiliary family $\mathcal{L}$ is derived from the following function family.

Definition 4.2. *For any probability distribution $\mathcal{Y}$ over $\mathbb{Z}^\ell$ and integer $m \geq \ell$, let $\mathcal{I}(m, \ell, \mathcal{Y})$ be the probability distribution over linear functions $[\mathbf{I} \mid \mathbf{Y}] \colon \mathbb{Z}^m \to \mathbb{Z}^\ell$ where $\mathbf{I}$ is the $\ell \times \ell$ identity matrix, and $\mathbf{Y} \in \mathbb{Z}^{\ell \times (m-\ell)}$ is obtained choosing each column of $\mathbf{Y}$ independently at random from $\mathcal{Y}$.*

We anticipate that we will set $\mathcal{Y}$ to the Gaussian input distribution $\mathcal{Y} = D^\ell_{\mathbb{Z},\sigma}$ in order to make $\mathcal{L}$ indistinguishable from $\mathcal{F}$ under a standard LWE assumption. But for generality, we prove some of our results with respect to a generic distribution $\mathcal{Y}$.

The following lemma shows that for a bounded distribution $\mathcal{Y}$ (and appropriate parameters), $\mathcal{I}(m, \ell, \mathcal{Y})$ is (statistically) uninvertible.

Lemma 4.3. *Let $\mathcal{Y}$ be a probability distribution on $[\mathcal{Y}] \subseteq \{-\sigma, \ldots, \sigma\}^n$, and let $X \subseteq \{-s, \ldots, s\}^m$. Then $\mathcal{I}(m, \ell, \mathcal{Y})$ is ϵ-uninvertible with respect to $\mathcal{U}(X)$ for $\epsilon = (1 + 2s(1 + \sigma(m - \ell)))^\ell / |X|$.*

Proof. Let $f = [\mathbf{I} \mid \mathbf{Y}]$ be an arbitrary function in the support of $\mathcal{I}(m, \ell, \mathcal{Y})$. We know that $|y_{i,j}| \leq \sigma$ for all i, j. We first bound the size of the image $|f(X)|$. By the triangle inequality, all the points in the image $f(X)$ have ℓ_∞ norm at most

$$\|f(\mathbf{u})\|_\infty \leq \|\mathbf{u}\|_\infty (1 + \sigma(m - \ell)) \leq s(1 + \sigma(m - \ell)).$$

The number of integer vectors (in $\mathbb{Z}^\ell$) with such bounded ℓ_∞ norm is

$$(1 + 2s(1 + \sigma(m - \ell)))^\ell.$$

Dividing by the size of X and using Lemma 2.4, the claim follows. □

Lemma 4.3 applies to any distribution $\mathcal{Y}$ with bounded support. When $\mathcal{Y} = D^\ell_{\mathbb{Z},\sigma}$ is a discrete Gaussian distribution, a slightly better bound can be obtained. (See also [4], which proves a similar lemma for a different, non-uniform input distribution $\mathcal{X}$.)

Lemma 4.4. *Let $\mathcal{Y} = D^\ell_{\mathbb{Z},\sigma}$ be the discrete Gaussian distribution with parameter $\sigma > 0$, and let $X \subseteq \{-s, \ldots, s\}^m$. Then $\mathcal{I}(m, \ell, \mathcal{Y})$ is ϵ-uninvertible with respect to $\mathcal{U}(X)$, for $\epsilon = O(\sigma m s / \sqrt{\ell})^\ell / |X| + 2^{-\Omega(m)}$.*

Proof. Again, by Lemma 2.4 it is enough to bound the expected size of $f(X)$ when $f \leftarrow \mathcal{I}(m, \ell, \mathcal{Y})$ is chosen at random. Remember that $f = [\mathbf{I} \mid \mathbf{Y}]$ where $\mathbf{Y} \leftarrow D^{\ell \times (m-\ell)}_{\mathbb{Z},\sigma}$. Since the entries of $\mathbf{Y} \in \mathbb{R}^{\ell \times (m-\ell)}$ are independent mena-zero subgaussians with parameter σ, by a standard bound from the theory of random matrices, the largest singular value $s_1(\mathbf{Y}) = \max_{\mathbf{0} \neq \mathbf{x} \in \mathbb{R}^m} \|\mathbf{Y}\mathbf{x}\| / \|\mathbf{x}\|$ of $\mathbf{Y}$ is at most $\sigma \cdot O(\sqrt{\ell} + \sqrt{m - \ell}) =$

$\sigma \cdot O(\sqrt{m})$, except with probability $2^{-\Omega(m)}$. We now bound the ℓ_2 norm of all vectors in the image $f(X)$. Let $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in X$, with $\mathbf{u}_1 \in \mathbb{Z}^\ell$ and $\mathbf{u}_2 \in \mathbb{Z}^{m-\ell}$. Then

$$\begin{aligned} \|f(\mathbf{u})\| &\leq \|\mathbf{u}_1 + \mathbf{Y}\mathbf{u}_2\| \\ &\leq \|\mathbf{u}_1\| + \|\mathbf{Y}\mathbf{u}_2\| \\ &\leq \left(\sqrt{\ell} + s_1(\mathbf{Y})\sqrt{m-\ell}\right) s \\ &\leq \left(\sqrt{\ell} + \sigma \cdot O(\sqrt{m})\sqrt{m-\ell}\right) s \\ &= O(\sigma m s). \end{aligned}$$

The number of integer points in the ℓ-dimensional zero-centered ball of radius $R = O(\sigma m s)$ can be bounded by a simple volume argument, as $|f(X)| \leq (R + \sqrt{\ell}/2)^n V_\ell = O(\sigma m s/\sqrt{\ell})^\ell$, where $V_\ell = \pi^{\ell/2}/(\ell/2)!$ is the volume of the ℓ-dimensional unit ball. Dividing by the size of X and accounting for the rare event that $s_1(\mathbf{Y})$ is not bounded as above, we get that $\mathcal{I}(m, \ell, \mathcal{Y})$ is ϵ-uninvertible for $\epsilon = O(\sigma m s/\sqrt{\ell})^\ell/|X| + 2^{-\Omega(m)}$. $\square$

We can now prove the one-wayness of the SIS function family by defining and analyzing an appropriate lossy function family. The parameters below are set up to expose the connection with LWE, via Proposition 2.9: $\mathrm{SIS}(m, m-n, q)$ corresponds to LWE in n dimensions (given m samples), whose one-wayness we are proving, while $\mathrm{SIS}(\ell = m - n + k, m - n, q)$ corresponds to LWE in $k \leq n$ dimensions, whose pseudorandomness we are assuming.

Theorem 4.5. *Let q be a modulus and let $\mathcal{X}, \mathcal{Y}$ be two distributions over $\mathbb{Z}^m$ and $\mathbb{Z}^\ell$ respectively, where $\ell = m - n + k$ for some $0 < k \leq n \leq m$, such that*

1. *$\mathcal{I}(m, \ell, \mathcal{Y})$ is uninvertible with respect to input distribution $\mathcal{X}$,*
2. *$\mathrm{SIS}(\ell, m-n, q)$ is pseudorandom with respect to input distribution $\mathcal{Y}$, and*
3. *$\mathrm{SIS}(m, m-n, q)$ is second-preimage resistant with respect to input distribution $\mathcal{X}$.*

Then $\mathcal{F} = \mathrm{SIS}(m, m-n, q)$ is one-way with respect to input distribution $\mathcal{X}$.

In particular, if $\mathrm{SIS}(\ell, m-n, q)$ is pseudorandom with respect to the discrete Gaussian distribution $\mathcal{Y} = D^\ell_{\mathbb{Z},\sigma}$, then $\mathrm{SIS}(m, m-n, q)$ is $(2\epsilon + 2^{-\Omega(m)})$-one-way with respect to the uniform input distribution $\mathcal{X} = \mathcal{U}(X)$ over any set $X \subseteq \{-s, \ldots, s\}^m$ satisfying

$$(C'\sigma m s/\sqrt{\ell})^\ell/\epsilon \leq |X| \leq \epsilon \cdot (q/s')^{m-n},$$

where s' is the largest divisor of q that is smaller than or equal to $2s$, and C' is the universal constant hidden by the $O(\cdot)$ notation from Lemma 4.4.

Proof. We will prove that $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a lossy function family, where $\mathcal{F} = \mathrm{SIS}(m, m-n, q)$ and $\mathcal{L} = \mathrm{SIS}(\ell, m-n, q) \circ \mathcal{I}(m, \ell, \mathcal{Y})$. It follows from Lemma 2.3 that both $\mathcal{F}$ and $\mathcal{L}$ are one-way function families with respect to input distribution $\mathcal{X}$. Notice that $\mathcal{F}$ is second-preimage resistant with respect to $\mathcal{X}$ by assumption. The indistinguishability of $\mathcal{L}$ and $\mathcal{F}$ follows immediately from the pseudorandomness of $\mathrm{SIS}(\ell, m-n, q)$ with respect to $\mathcal{Y}$, by a standard hybrid argument. So, in order to prove that $(\mathcal{L}, \mathcal{F}, \mathcal{X})$ is a lossy function family, it suffices to prove that $\mathcal{L}$ is uninvertible with respect to $\mathcal{X}$. This follows applying Lemma 2.5 to the function family $\mathcal{I}(m, \ell, \mathcal{Y})$, which is uninvertible by assumption. This proves the first part of the theorem.

Now consider the particular instantiation. Let $\mathcal{X} = \mathcal{U}(X)$ be the uniform distribution over a set $X \subseteq \{-s, \ldots, s\}^m$ whose size satisfies the inequalities in the theorem statement, and let $\mathcal{Y} = D_{\mathbb{Z},\sigma}^{\ell}$. Since $|X|(s'/q)^{m-n} \leq \epsilon$, by Lemma 4.1, $\mathrm{SIS}(m, m-n, q)$ is (statistically) second-preimage resistant with respect to input distribution $\mathcal{X}$. Moreover, since $(C\sigma ms/\sqrt{\ell})^{\ell}/|X| \leq \epsilon$, by Lemma 4.4, $\mathcal{I}(m, \ell, \mathcal{Y})$ is $(\epsilon + 2^{-\Omega(m)})$-uninvertible with respect to input distribution $\mathcal{X}$. □

In order to conclude that the LWE function is pseudorandom (under worst-case lattice assumptions) for uniformly random small errors, we combine Theorem 4.5 with Corollary 2.14, instantiating the parameters appropriately. For simplicity, we focus on the important case of a prime modulus q. Nearly identical results for composite moduli (e.g., those divisible by only small primes) are also easily obtained from Corollary 2.14, or by using either Theorem 2.13 or Theorem 2.12.

Theorem 4.6. *Let $0 < k \leq n \leq m - \omega(\log k) \leq k^{O(1)}$, $\ell = m - n + k$, $s \geq (Cm)^{\ell/(n-k)}$ for a large enough universal constant C, and q be a prime such that $\max\{3\sqrt{k}, (4s)^{m/(m-n)}\} \leq q \leq k^{O(1)}$. For any set $X \subseteq \{-s, \ldots, s\}^m$ of size $|X| \geq s^m$, the $\mathrm{SIS}(m, m-n, q)$ (equivalently, $\mathrm{LWE}(m, n, q)$) function family is one-way (and pseudorandom) with respect to the uniform input distribution $\mathcal{X} = \mathcal{U}(X)$, under the assumption that SIVP_γ is (quantum) hard to approximate, in the worst case, on k-dimensional lattices to within a factor $\gamma = \tilde{O}(\sqrt{k} \cdot q)$.*

A few notable instantiations are as follows. To obtain pseudorandomness for binary errors, we need $s = 2$ and $X = \{0, 1\}^m$. For this value of s, the condition $s \geq (Cm)^{\ell/(n-k)}$ can be equivalently be rewritten as

$$m \leq (n-k) \cdot \left(1 + \frac{1}{\log_2(Cm)}\right),$$

which can be satisfied by taking $k = n/(C' \log_2 n)$ and $m = n(1 + 1/(c \log_2 n))$ for any desired $c > 1$ and a sufficiently large constant $C' > 1/(1 - 1/c)$. For these values, the modulus should satisfy $q \geq 8^{m/(m-n)} = 8n^{3c} = k^{O(1)}$, and can be set to any sufficiently large prime $p = k^{O(1)}$.[1]

Notice that for binary errors, both the worst-case lattice dimension k and the number $m - n$ of "extra" LWE samples (i.e., the number of samples beyond the LWE dimension n) are both sublinear in the LWE dimension n: we have $k = \Theta(n/\log n)$ and $m - n = O(n/\log n)$. This corresponds to both a stronger worst-case security assumption, and a less useful LWE problem. By using larger errors, say, bounded by $s = n^{\epsilon}$ for some constant $\epsilon > 0$, it is possible to make both the worst-case lattice dimension k and number of extra samples $m - n$ into (small) linear functions of the LWE dimension n, which may be sufficient for some cryptographic applications of LWE. Specifically, for any constant $\epsilon < 1$, one may set $k = (\epsilon/3)n$ and $m = (1 + \epsilon/3)n$, which are easily verified to satisfy all the hypotheses of Theorem 4.6 when $q = k^{O(1)}$ is sufficiently large. These parameters correspond to $(\epsilon/3)n = \Omega(n)$ extra samples (beyond the LWE dimension n), and to the worst-case hardness of lattice problems in dimension $(\epsilon/3)n = \Omega(n)$. Notice that for $\epsilon < 1/2$, this version of LWE has much smaller errors than allowed by previous LWE hardness proofs, and it would be subject to subexponential-time attacks [2] if the number of samples were not restricted. Our result shows that if the number of samples is limited to $(1 + \epsilon/3)n$, then LWE maintains its provable security properties and conjectured exponential-time hardness in the dimension n.

One last instantiation allows for a linear number of samples $m = c \cdot n$ for any desired constant $c \geq 1$, which is enough for most applications of LWE in lattice cryptography. In this case we can choose (say)

[1] Here we have not tried to optimize the value of q, and smaller values of the modulus are certainly possible: a close inspection of the proof of Theorem 4.6 reveals that for binary errors, the condition $q \geq 8n^{3c}$ can be replaced by $q \geq n^{c'}$ for any constant $c' > c$.

$k = n/2$, and it suffices to set the other parameters so that

$$s \geq (Cm)^{2c-1} \quad \text{and} \quad q \geq (4s)^{c/(c-1)} \geq 4^{c/(c-1)} \cdot (Ccn)^{2c+1+1/(c-1)} = k^{O(1)}.$$

(We can also obtain better lower bounds on s and q by letting k be a smaller constant fraction of n.) This proves the hardness of LWE with uniform noise of polynomial magnitude $s = n^{O(1)}$, and any linear number of samples $m = O(n)$. Note that for $m = cn$, any instantiation of the parameters requires the magnitude s of the errors to be at least n^{c-1}. For $c > 3/2$, this is more noise than is typically used in the standard LWE problem, which allows errors of magnitude as small as $O(\sqrt{n})$, but requires them to be independent and follow a Gaussian-like distribution. The novelty in this last instantiation of Theorem 4.6 is that it allows for a much wider class of error distributions, including the uniform distribution, and distributions where different components of the error vector are correlated.

Proof of Theorem 4.6. We prove the one-wayness of $\mathrm{SIS}(m, m-n, q)$ (equivalently, $\mathrm{LWE}(m, n, q)$ via Proposition 2.9) using the second part of Theorem 4.5 with $\sigma = 3\sqrt{k}$. Using $\ell \geq k$ and the primality of q, the conditions on the size of X in Theorem 4.5 can be replaced by simpler bounds

$$\frac{(3C'ms)^\ell}{\epsilon} \leq |X| \leq \epsilon \cdot q^{m-n},$$

or equivalently, the requirement that the quantities $(3C'ms)^\ell/|X|$ and $|X|/q^{m-n}$ are negligible in k. For the first quantity, letting $C = 4C'$ and using $|X| \geq s^m$ and $s \geq (4C'm)^{\ell/(n-k)}$, we get that $(3C'ms)^\ell/|X| \leq (3/4)^{-\ell} \leq (3/4)^{-k}$ is exponentially small (in k). For the second quantity, using $|X| \leq (2s+1)^m$ and $q \geq (4s)^{m/(m-n)}$, we get that $|X|/q^{m-n} \leq (3/4)^m$ is also exponentially small.

Theorem 4.5 also requires the pseudorandomness of $\mathrm{SIS}(\ell, m-n, q)$ with respect to the discrete Gaussian input distribution $\mathcal{Y} = D^\ell_{\mathbb{Z},\sigma}$, which can be based on the (quantum) worst-case hardness of SIVP on k-dimensional lattices using Corollary 2.14. (Notice the use of different parameters: $\mathrm{SIS}(m, m-n, q)$ in Corollary 2.14, and $\mathrm{SIS}(m-n+k, m-n, q)$ here.) After properly renaming the variables, and using $\sigma = 3\sqrt{k}$, the hypotheses of Corollary 2.14 become $\omega(\log k) \leq m - n \leq k^{O(1)}$, $3\sqrt{k} < q < k^{O(1)}$, which are all satisfied by the hypotheses of the Theorem. The corresponding assumption is the worst-case hardness of SIVP_γ on k-dimensional lattices, for $\gamma = k\omega_k q/\sigma = \sqrt{k}\omega_k q/3 = \tilde{O}(\sqrt{k}q)$, as claimed. This concludes the proof of the one-wayness of LWE.

The pseudorandomness of LWE follows from the sample-preserving search-to-decision reduction of [17]. □

References

[1] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[2] S. Arora and R. Ge. New algorithms for learning in presence of errors. In *ICALP (1)*, pages 403–415, 2011.

[3] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737, 2012.

[4] M. Bellare, E. Kiltz, C. Peikert, and B. Waters. Identity-based (lossy) trapdoor functions and applications. In *EUROCRYPT*, pages 228–245, 2012.

[5] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[6] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography*, pages 1–16, 2011.

[7] J.-Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *FOCS*, pages 468–477, 1997.

[8] Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.

[9] D. Dadush, C. Peikert, and S. Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *FOCS*, pages 580–589, 2011.

[10] N. Döttling and J. Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. Manuscript. To appear in Eurocrypt 2013, 2013.

[11] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.

[12] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[13] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[14] D. Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM J. Comput.*, 34(1):118–169, 2004.

[15] D. Micciancio. Duality in lattice cryptography. In *Public Key Cryptography*, 2010. Invited talk.

[16] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[17] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484, 2011.

[18] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[19] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[20] D. Micciancio and O. Regev. Lattice-based cryptography. In *Post Quantum Cryptography*, pages 147–191. Springer, February 2009.

[21] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

[22] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97, 2010.

[23] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

[24] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[25] D. Wagner. A generalized birthday problem. In *CRYPTO*, pages 288–303, 2002.

Accelerating Computations on Encrypted Data with an FPGA[1]

By David Bruce Cousins and Kurt Rohloff, Raytheon BBN Technologies

Healthcare, financial, government, and military organizations depend on encryption to secure sensitive data. Historically, this data has had to be decrypted before it could be processed or analyzed. As a result, data processing had to be performed on secured hardware, eliminating the possibility of using the cloud or other low-cost, third-party computing resources.

At the Symposium on the Theory of Computing in 2009, Craig Gentry of IBM presented a fully homomorphic encryption (FHE) scheme that made it possible to send sensitive data to an unsecured server, process it there, and receive an encrypted result, without ever decrypting the original data. While FHE was a major theoretical breakthrough, actual FHE implementations are many orders of magnitude too slow to be of practical use, particularly for large encryption keys and ciphertexts.

As a step toward a practical FHE implementation, we have developed a somewhat homomorphic encryption (SHE) scheme that, with modifications, can be converted into an FHE scheme. Current FHE implementations depend on complicated operations that are inefficient when performed on a CPU, and our goal was to take advantage of the parallelism and pipelining of FPGAs using MATLAB®, Simulink®, and HDL Coder™. Homomorphic encryption is an active area of study, and new advances are being made regularly. By using MATLAB and Simulink instead of a lower-level programming language, we can keep pace with these developments by rapidly implementing improvements to the algorithms.

Homomorphic Encryption Basics

FHE enables secure and private computation using encrypted data. In theory, computations can be carried out using just two FHE operations: EvalAdd and EvalMult. These operations are similar to binary XNOR and AND operations, but they operate on encrypted bits, and their result remains encrypted. Current FHE schemes are based on computationally intensive stochastic lattice theory problems. Stochastics introduce noise into each EvalAdd or EvalMult operation. The amount of noise increases rapidly with the number of operations performed. FHE schemes address this buildup of noise by periodically running a bootstrapping algorithm on the intermediate results. One problem with this approach is that the bootstrapping algorithm is computationally expensive. A second is that current FHE schemes entail modular arithmetic with a large modulus. The operations required are memory-intensive and inefficient when performed on standard CPUs.

SHE schemes avoid the need for bootstrapping by limiting the number of EvalAdd and EvalMult operations that must be performed to keep noise below an acceptable threshold. SHE schemes can be augmented with bootstrapping operations to produce FHE schemes, provided that the additional number of operations for bootstrapping itself keeps the noise below this threshold.

[1] Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited.)

Implementing Modulo Add and Multiply in MATLAB and Simulink

FHE schemes are based on EvalAdd and EvalMult, elementwise vector addition and multiplication operations performed in modulo q arithmetic where q is a large prime integer. In MATLAB, EvalAdd is expressed simply as

```
c = mod(a+b, q)
```

and EvalMult is expressed as

```
c = mod(a.*b, q)
```

These straightforward representations make it easy to perform calculations on a CPU, but to exploit the parallelism of FPGAs we needed to develop efficient software implementations of EvalAdd and EvalMult in VHDL®. We began by prototyping algorithms in MATLAB—for example, the EvalAdd operation with inputs bounded by the modulus q:

```
c=a+b;

cgteq = (c>=q);

c(cgteq)=c(cgteq)-q;
```

Our initial MATLAB representation served as a reference for the Simulink model, which would be used for hardware implementation and HDL code generation (Figure 1). With the initial MATLAB representation, we were able to explore several theoretical approaches because MATLAB handled large complicated vector and matrix operations quickly and naturally. Additionally, we were able to use Fixed-Point Designer™ to generate bit-accurate fixed-point solutions, including modeling rollover in our adders, with only minor modifications to our systems.

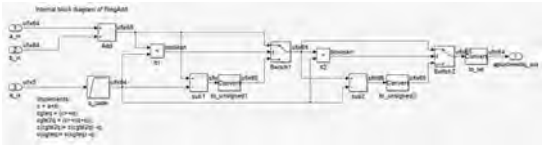


Figure 1. Simulink model of the modulo add operation.

Once we had converged on a sound approach, a detailed Simulink model was created for HDL implementation. With Simulink, we were able to lay out the logical components in the design and automatically generate optimized HDL code. We were also able to add capabilities, including supporting multiple moduli and optimizing our models to use table lookups, in a controlled, incremental way.

The model can process one pair of inputs on each clock cycle.

To verify the model, we compared the results it produced with the results from our MATLAB code.

Modulo multiplication is much more complicated than modulo addition. To manage this complexity and enable more efficient pipelining in the generated HDL, we have developed a multiple-word modulo multiply operation based on the Barrett reduction algorithm. Figure 2 shows the structure of a pipelined, four-stage multiple-word multiplier, where each stage operates on a subword of the large overall word.

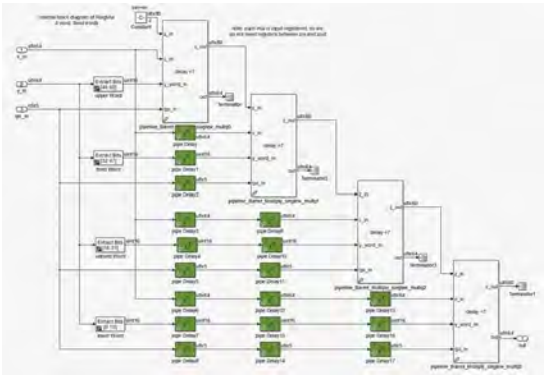


Figure 2. Simulink model of the 64-bit Barrett modulo multiply operation, showing four stages.

Figure 3 shows the detailed model of a single stage, also pipelined for efficient HDL generation. Here again, the flexibility of Simulink enables us to specify the actual word widths at run time. The same models are used for generating 48-bit as well as 64-bit arithmetic HDL.

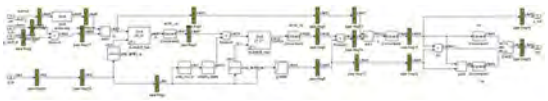


Figure 3. Simulink model of a single stage of the Barrett modulo multiply operation.

Implementing the Chinese Remainder Transform

Our SHE scheme uses the Chinese remainder transform (CRT) to simplify the structure of our add and multiply operations. CRT resembles the discrete Fourier transform (DFT), but uses modular integer arithmetic instead of complex arithmetic. We implemented the CRT as an EvalMult operation followed by a fast numeric transform (like a fast Fourier transform [FFT] but with modulo arithmetic).

To create the Simulink model for the FFT shown in Figure 3, we started with a standard streaming FFT, reordered the inputs, and converted from complex arithmetic to modulo arithmetic using integer fixed-point arithmetic (Figures 4a, 4b, and 4c).

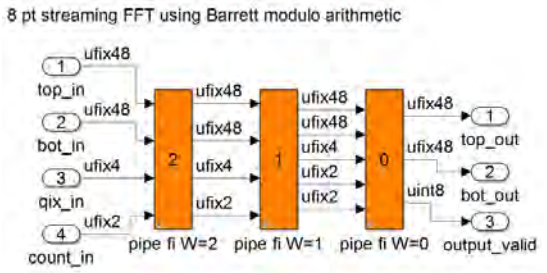


Figure 4a. Streaming fast numeric transform modeled in Simulink.

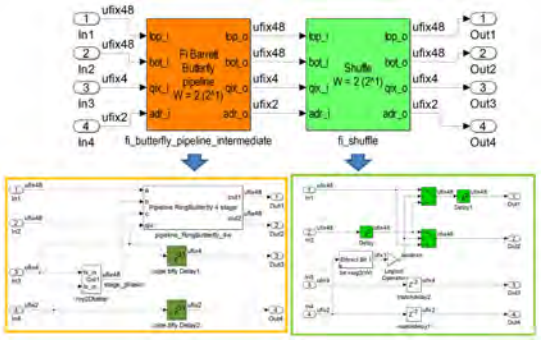


Figure 4b. Detail of a single stage of the fast numeric transform showing the butterfly and shuffle models.

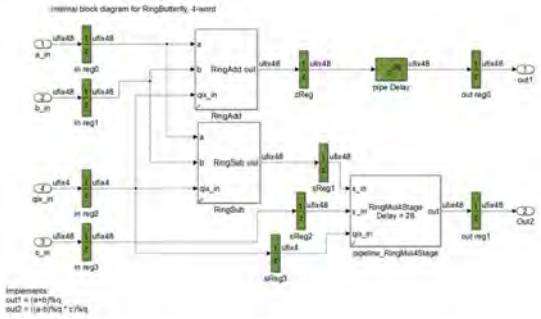


Figure 4c. Detail of the modulo Butterfly operation of the fast numeric transform.

We were able to develop a streaming model that consisted of two basic blocks that are parameterized at run time. By cascading log2(N) blocks together we can assemble FFTs of various power-of-2 sizes. This approach allows us to process two input samples every clock cycle. We have implemented and benchmarked up to a 2^14 size CRT on a Xilinx® Virtex®-7 in this manner.

Moving from Floating Point to Fixed Point and from Model to VHDL

When we started prototyping in MATLAB we used floating-point math, which provided a quick, easy way to understand the computations that we needed to support. We then used Fixed-Point Designer to transition from floating-point to fixed-point arithmetic with varying integer bit-widths.

Our MATLAB code and Simulink models use the same fixed-point variables and produce output in the same format, simplifying verification of test results. When running simulations, we can specify the bit-width of the input data. The intermediate mathematical operations are automatically sized by Fixed-Point Designer, enabling us to use same models for inputs of varying bit-widths.

The transition to fixed-point math is a required step for VHDL implementation. We generated the required VHDL from our Simulink models using HDL Coder. We simulated the VHDL using Mentor Graphics® ModelSim®, and synthesized the code on a Xilinx Virtex FPGA. The generated VHDL can be used on FPGAs from different vendors, enabling us to benchmark across multiple platforms. We also use HDL Verifier™ to validate and demonstrate the generated VHDL running on a Xilinx ML 605 evaluation board.

Accelerating Development

The combination of MATLAB, Simulink, Fixed-Point Designer, HDL Coder, and HDL Verifier enables us to develop, implement, and improve our encryption scheme much faster than would be possible with traditional methods. Speed is essential to our efforts because FHE theory is evolving so rapidly. Several times a year, innovations come to light that require a rewrite of our code and subsequent changes to our model. We estimate that development would take two to three times longer if we were working in C, Python, or another

low-level language. Without Simulink and HDL Coder, production of VHDL code would be similarly slowed because our team does not have experience writing VHDL from scratch.

As we continue to increase the capabilities of our SHE scheme and improve its performance, rapid prototyping in MATLAB and Simulink and automatic VHDL code generation with HDL Coder remain central to our development efforts.

Products Used

- MATLAB
- Simulink
- Fixed-Point Designer
- HDL Coder
- HDL Verifier

Learn More

- Implementing MATLAB Algorithms in FPGAs and ASICs

See more articles and subscribe at mathworks.com/newsletters.

Published 2013
92098v00

An FPGA Co-Processor Implementation of Homomorphic Encryption

David Bruce Cousins, John Golusky, Kurt Rohloff, Daniel Sumorok
Raytheon BBN Technologies
Cambridge, Massachusetts USA
{dcousins, jgolusky, dsumorok, krohloff}@bbn.com

***Abstract*— One of the goals of the DARPA PROCEED program has been accelerating the development of a practical Fully Homomorphic Encryption (FHE) scheme. For the past three years, this program has succeeded in accelerating various aspects of the FHE concept toward practical implementation and use. FHE is a game-changing technology to enable secure, general computation on encrypted data on untrusted off-site hardware, without the data ever being decrypted for processing. FHE schemes developed under PROCEED have achieved multiple orders of magnitude improvement in computation, but further means of acceleration, such as implementations on specialized hardware, such as an FPGA can improve the speed of computation even further.**

The current interest in FHE computation resulted from breakthroughs demonstrating the existence of FHE schemes [1, 2] that allowed arbitrary computation on encrypted data. Specifically, our contribution to the Proceed program has been the development of FPGA based hardware primitives to accelerate the computation on encrypted data using an FHE cryptosystem based on NTRU-like lattice techniques [3] with additional with additional support for efficient key switching and modulus reduction operations to reduce the frequency of bootstrapping operations [4]. Cipher texts in our scheme are represented as rectangular matrices of 64-bit integers. This bounding of the operand sizes has allowed us to take advantage of modern code generation tools developed by Mathworks to implement VHDL code for FPGA circuits directly from Simulink models. Furthermore the implicit parallelism of the scheme allows for large amounts of pipelining in the implementation in order to achieve efficient throughput. The resulting VHDL is integrated into an AXI4 bus "Soft System on Chip" using Xilinx platform studio and a Microblaze soft core processor running on aVirtex7 VC707 evaluation board. This report presents new Simulink primitives that had to be developed to deal with these new requirements.

Keywords—Fully Homomorphic Encryption; Co-processor; SIMULINK; FPGA

I. Introduction - A Quick Review of Fully- and Somewhat- Homomorphic Encryption

Our team recently published our work to design, implement and evaluate a scalable FHE scheme which addresses the limitations for secure arbitrary computation [4]. Our implementation uses a variation of a not previously implemented bootstrapping scheme [5] simplified for power-of-2 rings. We also use a "double-Chinese Remainder Transform (CRT)" representation of cipher texts which is discussed in [6]. With this double-CRT representation, we can select parameters so that cipher texts are secure when represented as matrices of 64 bit integers, but still support the secure execution of programs on commodity computing devices without expending unnecessary computational overhead manipulating large multi-hundred-bit or even multi-thousand-bit integers. Additionally, the parallelism implicit in this data representation is easily exploited to achieve efficiencies during implementation.

Our implementation encrypts a plaintext bit into a two dimension array of 64 bit unsigned integers[1]. We use a residue number system implementation to represent cipher texts as T sets of length-N integer vectors. A ring in the tower entry t has a unique modulus q_t which bounds all entries in that ring. The n dimension is known as the ring size, and the t dimension as the tower size. This representation allows us to operate in parallel on the smaller bit width modulo q_t values instead of on a single modulus q of much larger bit width, where $q = q_1 * q_2 * \ldots * q_T$ for pairwise co-prime moduli q_i..

As outlined in [4], our implementation requires only a few elementary operations to be implemented on the FPGA hardware in order to achieve large run time speedups over conventional CPU implementations. These operations are:

- RingAdd: $c_{n,t} = (a_{n,t} + b_{n,t}) \% q_t$
- RingSub: $c_{n,t} = (a_{n,t} - b_{n,t}) \% q_t$
- RingMul: $c_{n,t} = (a_{n,t} * b_{n,t}) \% q_t$

All three of the above operations can be parallelized or pipelined over both n and t . Also required are the

- CRT and Inverse CRT, which are implemented as a Number Theoretic Transform [7] coupled with a pre- or post-RingMul with an appropriate Twiddle Vector.
- Round: A function to perform modulo rounding using different tower moduli (detailed below).

In our cryptosystem, two key operations are defined: EvalAdd and EvalMult. When our parameters are chosen such that a single plaintext bit is encrypted, the resulting operations on the encrypted data are XOR and AND respectively. These

Sponsored by Air Force Research Laboratory (AFRL) Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

[1] While the actual number of bits is determined by the parameter selection of the cryptosystem, we select 64 as our maximum dimension for FPGA implementation.

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

two operations allow us to implement any Boolean operation of input cipher text [2].

Our crypto-system, like many FHE systems, is random (noisy) in nature. Because of this, only a limited number of operations can be performed on the encrypted data before the noise dominates and decryption is no longer guaranteed. EvalAdd does not add noise to the system, so an unlimited number of such operations are allowed to be chained together. EvalMult however does add noise, and this limits the number of such operations that can be chained together. The double CRT representation allows a very straightforward implementation that controls this noise. This requires the use of both key switching and modulus reduction whenever an EvalMult is performed. The combination of these three steps is known as a Composed EvalMult (CEM). The property of CEM is that for a pair of inputs of a given tower size *t*, the output is a cipher text of tower size *t-1*. Thus for an initial tower size of *T*, at most *(T-1)* CEM operations can be performed before the noise in the cypher-text grows beyond the point where it can be reliably decrypted. An implementation that has a limit to the allowable number of Homomorphic operations is called *Somewhat Homomorphic*.

The dimensions of the cryptosystem are determined algorithmically, and are a function of security required and the number of CEM operations required to implement the desired application. If the number of operations required by the application exceeds O(16), then a Bootstrapping operation will be required to reset the noise generated by the cryptographic operations. Bootstrapping is currently on the order of 10 CEM equivalent operations for reasonable security parameters. Bootstrapping has the property of taking a cipher text of tower size *t*, and generating a new 'refreshed' cipher text of the system's original tower size *T*. Thus an unlimited number of operations can be performed on the data. This kind of implementation is called *Fully Homomorphic*. The remainder of our paper will discuss the current FPGA implementation of the functions required for Somewhat Homomorphic operation. Our planned implementation of the functions needed for Fully Homomorphic operation will be implemented in our final phase of the program this year.

II. VHDL Implementations of Fast Modulus Arithmetic and Chinese Remainder Transforms (CRT) Using Simulink-Based Models

A. Optimisations and Refinements To Previous Implementations

We have previously reported on our Simulink-based implementations of the three modulus arithmetic functions, as well as the forward and inverse CRT functions[8, 9]. Our current work has updated these implementations to allow VHDL code generation with a doubling of circuit clock speeds to 200 MHz. This was done by performing the following optimizations.

Mathworks determined that by selecting synchronous vs. asynchronous reset in the Simulink to HDL generation parameters, the resulting VHDL mapped more efficiently into the registers built into the DSP48E blocks on the Vertex 7 FPGA, increasing the efficiency of the resulting mapped VHDL by eliminating extra routing traces.

The previous circuits were designed to run at a minimum speed of 100 MHz. We determined that adding explicit pipelining stages in the form of delay lines to the model enabled the Xilinx tools to better optimize FPGA mapping during place and route pipelining stages. Specifically pipelines were added between arithmetic operations within the RingAdd (4 stages), RingSub (3 stages), RingMul (188 stages) models. Since our target ring size can be as large as 2^{14}, and all the towers of a variable are processed sequentially, the delay incurred from filling the pipeline is expected to be minimal.

Once the models were maximally pipelined, we identified several large (64 by 64 bit) product blocks within our RingMul Barret multiplication implementation [9, 10] as being the slowest components, and re-implemented them as an expanded multiplication model consisting of four parallel 32 by 32 bit products, and a pipelined accumulation of partial sums. This further increased the achievable clock speeds. We discovered that adding additional pipelines of length four, both before and after each resulting smaller product block further allowed the Xilinx optimizer to break these product blocks into multiple DSP48E multipliers in a distributed fashion. This allowed the RingMul circuit to perform at speeds in excess of 350 MHz, well in excess of our target 200 MHz.

Several of our circuits utilize lookup tables, both for storing the moduli q_i and for storing various twiddle table entries for the CRT and inverse CRT. Our previous direct implementation of the table lookup using the Simulink Lookup function block maps the resulting ROM directly into gate circuitry. This can increase the place and route drastically for very large tables, and also can result in less efficient circuits. Mathworks determined that by placing an additional delay line, with a `"ResetType = none"` HDL block property let the Xilinx tools map the table to block ram in the FPGA, which is a more efficient utilization of resources on the chip.

B. FPGA Hardware Selection

Our FPGA selection was driven by the need for a large number of hardware multipliers on the chip. Due to cost constraints we wanted to use a commercial off-the-shelf FPGA board for our experiments. Our selection of the Virtex 7 VC707 evaluation board was driven by the following sizing requirements. Our target ring size of 2^{14} requires 1110 DSP48 blocks for the CRT and the same number for the inverse CRT. The VC707 has a Virtex 7 485T chip which contains 2800 such blocks, more than sufficient to implement our projected set of FHE primitives. Additionally, we require on-board DDR memory for storage of encrypted variables, and high speed Ethernet and PCI interfaces to exchange data with the host computer. All these are present on the VC707.

[2]Any arbitrary Boolean function can be constructed from NAND operations. Since NOT(a) == XOR(a, 1), and NAND(a, b) == NOT(AND(a, b)), the two Homomorphic operations are a sufficient set.

C. FPGA System Architecture

The design goal of our FPGA system was to be able to operate as an attached processor to accelerate the FHE primitive operations in way that allows one to chain together several operations in order to minimize the overhead due to data transfer. An attached processor design was developed in which a software programmable microcontroller would manage I/O communications with the host via Ethernet or PCI memory map, manage on board data storage in the form of an encrypted register file, and manage data transfer to and from the FHE primitive modules in as efficient manner as possible.

We decided to use the Xilinx Platform Studio Microblaze soft core processor and AXI4 interconnect architecture to implement the attached FHE processor. Fig. 1 shows a system block diagram of the resulting system. The Xilinx platform studio enables us to implement our FHE primitives as streaming co-processors on the AXI bus. An AXI4 lite bus is used to set control parameters of our Ring operation circuits, such as ring size, and tower size.

The main AXI4 interconnect is a 256 bit bus connecting the DDR ram with the various FHE primitives. The I/O rate into and out of DDR memory limits the overall processing speed of the system. Our RingAdd, RingSub and RingMul primitives each require two input streams and one output streams. Fig. 2 shows how we currently integrate our FHE primitives with the AXI4 stream interconnect. Each of these three operations is parallelized across ring elements as well as tower indices. These data streams are implemented using a pair of AXI4 DMA controllers, each handling one input and one output. Data is clocked in and out of the bus at 400 MHz, and streamed via individual AXIS buses between the DMAs to the AXI stream blocks where they are buffered with FIFOs and split into eight parallel 64 bit input data streams, and four 64 bit output data streams. Current implementations of these three functions are clocked at 100 MHz, so four parallel instantiations of each operation are used to keep the I/O pipelines full. Future implementations of these primitives are planned to be clocked at 200 MHz, and as such will only support two instantiations in parallel.

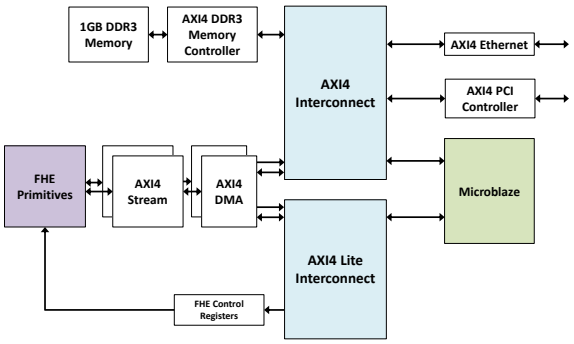


Figure 1: System block diagram showing major components and the AXI4 interconnect.

The forward and inverse CRT modules require slightly different interfaces. CRT operations are parallelizable across tower entry but not across ring index. Thus CRT's cannot be parallelized in the same way as the Ring operations. Currently we have a single CRT or inverse CRT at a time operating at 100 MHz. Future implementations will run these two operations at 200 MHz, but the multiplier resources required for the planned ring size of 2^{14} will prohibit mapping more than one forward and one inverse CRT onto the 485T chip.

D. Microblaze Software Architecture

The Xilinx platform studio is used to implement a Microblaze soft core processor. The system architecture is based on the demo hardware self-test example that is provided with the Xilinx board. The software architecture is based on the web-service example provided with the Xilinx Virtex 6 ML605 evaluation kit, updated with the Xilinx SGMII 144 Ethernet controller. The software controlling the system on the Microblaze is written in C code. The PC "host" end of the software interface is also written in C. The host interface currently is implemented in two versions. The first is a stand-alone test bench that can test and exercise the operation of the attached FHE processor. The second version interfaces with Matlab via a file interchange mechanism to support demonstration Homomorphic Encryption application programs. The interfaces use either Ethernet or PCI bus I/O based on compile flags.

The system software is multithreaded to allow the use of Ethernet TCP/IP socket I/O. A network thread manages socket level I/O between the host and the attached processor. Another thread reads the incoming messages from the socket, parses the commands received and dispatches execution to various subroutines. The PCI interface is written to emulate the buffer I/O of the Ethernet interface, allowing the same software to be used for both Ethernet and PCI operation.

The DDR3 ram is partitioned into a set of register data structures, as well as a set of internal registers to store constants used in our encryption schemes. Each register can hold one encrypted bit in the form of a two dimensional vector of

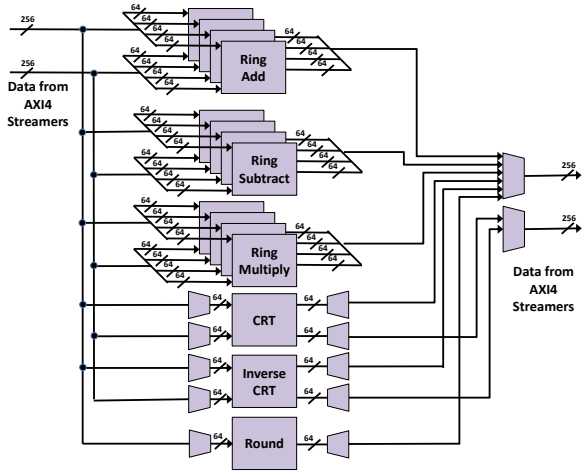


Figure 2: Integration of FHE primitives with the AXI stream data streams.

unsigned long longs that are allocated out of DDR ram. One dimension (the fastest index) is the ring size *N* and is a compiled constant. The other dimension, the tower size, varies with the state of the register. Typically registers are loaded into the FHE coprocessor with a fixed starting number of the tower elements (up to MAX_TOWER_SIZE = 32 elements). We eliminate the highest tower entries one by one as each CEM operation is performed.

The registers are allocated out of heap in the DDR ram. There are three flavors of registers: Input, Output and Scratch. This design decision was made in order to allow us to later segregate I/O and scratch registers into different memory locations if that were to increase throughput (allowing simultaneous host access to the I/O registers while the FGPA was processing with the Scratch registers. The quantity of each register type is software defined at compile time but there is usually a small numbers of Input and Output registers and as many Scratch registers as will utilize all the available heap space. Control structures mark the current tower size of each register, and if the register is used or not. Registers are allocated so they are aligned to 32 byte address boundaries in order to allow the AXI4 DMA engines to move the register data into and out of the FHE primitives. This format allows the contents of an entire register (all used towers) to be streamed with only one DMA transfer.

The communication protocol between the PC host and the FPGA board is message based. The messages are in ASCII. Messages can span multiple socket buffers; with multiple socket calls made until enough text has been parsed to complete a message (double cr/lf indicates the end of a message). Each message can contain several instructions to the processor, separated by cr/lf). Each processor instruction is then parsed. The parsing test starts with a keyword that defines the rest of the instruction format. The keywords are shown in Table 1. The system's assembly language has the syntax shown in Table 2.

TABLE I. CONTROL PROTOCOL KEYWORDS

Key word	Function
LOAD	Transfer the contents of the message (ASCII) into a particular Input register.
GET	Request the contents of a particular output register to be loaded into an ASCII message buffer and sent back to the host.
STATUS	Generates a short report on the FPGA board console for debugging showing the contents of all used registers, a listing of the current program loaded.
PROG	Loads a sequence of operations to be performed on the register data, in a simple assembly language.
RUN	Starts a software Finite State Machine to run the stored program to completion.
CRT, ICRT, CEM	A single command that will LOAD two registers, perform a forward CRT, inverse CRT or Composed EvalMult on them and GET the resulting output. Used for accelerating applications that only require these three operations.
RESET	Resets the system to its original state.

TABLE II. AVALIABLE OPCODES FOR HOMOMORPHIC ENCRYPTED PROGRAMS

Opcode	Example	Description
LOAD	R1 = LOAD(In0)	Moves data from an input register to scratch register, all active tower elements are moved.
STORE	Out4 = STORE(R3)	Moves data from a scratch register to output register, all active tower elements are moved.
RADD	R2 = RADD(R3, R4)	Sets up DMAs of the two input and one output registers to the RingAdd circuit. All active tower elements are processed I one large data flow.
RSUB	R2 = RSUB(R3, R4)	Same as RingAdd, except the RingSub circuit is the target/source of the I/O DMAs.
RMUL	R2 = RMUL(R3, R4)	Same as RingAdd, except the RingMul circuit is the target/source of the I/O DMAs.
CRT	R3= CRT(R1, R2)	Same as RingAdd, except the input and output registers are used as endpoints for pairs of DMA transfers, each moving one half of the ring data. Note second input register is used as a scratch register so is contents are destroyed.
ICRT	R2 = ICRT(R4, R5)	Same as CRT except an inverse CRT circuit is used.
EMULC	R2 = EMULC(R3, R4)	Executes a ComposedEvalMult, in software which in turns executes several Ring primitives (see below). Note that output register is one tower smaller than the input registers.

An example simple program in now given in Table 3. The program first moves encrypted data from input register 0, to scratch register 0, then repeats the process for a second input variable to register 1. It then computes a RingAdd, RingSub and RingMul using the two inputs, and storing the result in scratch registers 2, 3 and 4 respectively. It then stores those three results in output registers 0, 1 and 2 respectively.

Typical system operation would be for the user to execute two LOAD commands to load the contents of input registers 0 and 1 with encrypted data (the encryption being done on the secure host). The user then executes a RUN command to allow the Homomorphic operations to be run on the unsecure FPGA processor. Then subsequent calls to GET commands will

TABLE III. SAMPLE PROGRAM

```
R0 = LOAD(In0)
R1 = LOAD(In1)
R2 = RADD(R0,R1)
R3 = RSUB(R0,R1)
R4 = RMUL(R0,R1)
Out0 = STORE(R2)
Out1 = STORE(R3)
Out2 = STORE(R4)
```

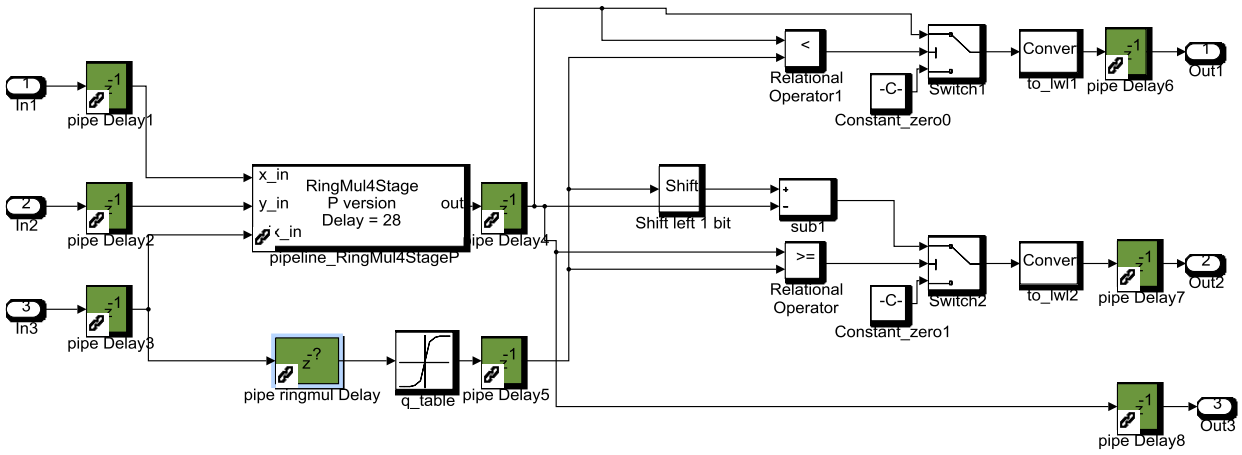


Figure 3: Simulink Model of Round Function.

transfer the resulting encrypted result data back to the host. Finally decryption would be done on the secure host.

E. Microcode Implementation of ComposedEvalMult

As mentioned above, one of the new functions implemented in our system is the ComposedEvalMult (CEM) which is fully detailed in [4]. This function is implemented in our software controller as a series of C function calls, all but one of which are executed with previously existing primitives. First, a RingMul operation performs the multiply. Next a key-switch operation is performed consisting of another RingMul of the product with a hint variable defined by the cryptosystem. Then, a modulus reduction operation is performed on the single highest tower entry of the result which consists of an inverse CRT and a new Rounding operation.

This Rounding operation is implemented as a new hardware function because it contains operations not available in the other ring functions. Fig. 3 shows the Simulink Model consisting of a modified EvalMult operation (using a modified set of moduli qi), and a pair of operations selected by the range of the result which ensure the output is bounded within an appropriate range. The operations are performed in a pipelined manner as well, to allow execution at 200 MHz.

The result of the rounding operation is a pair of new ring vectors that are then in turn applied to each remaining tower entry to reduce the noise accumulated by the initial product. These vectors are first processed with a series of RingAdds, RingSubs and a CRT using each of the corresponding ring moduli. The end result is that the highest tower ring is eliminated from the cipher text, and the overall noise of the system remains at a usable (i.e. de-cryptable) level.

III. Current Results and Next Steps

Our presentation will include I/O timing, run-time and chip utilization details of our attached processor performing the suite of ring primitives on various ring sizes, based on the implementation in our Virtex 7 VC707 evaluation board.

Future plans for our FPGA system include adding all Ring primitives that will be required to accelerate the Bootstrapping operation described in [4]. The CRT and inverse CRT operations will be modified to allow the Number Theoretic Transform (NTT) portion [7] to be combined into one circuit, saving a large amount of FPGA multiplier resources. Additionally, multiple ring sizes will be supported by modifications to the NTT to support multiple power-of-two ring sizes. This will allow us to support the Ring Reduction operation in [4] for increased computational efficiency. Note that all of the other primitives can at arbitrary ring sizes. The final target ring size is 2^{14}, which will support relatively secure computation.

Acknowledgment

We would like to acknowledge Christopher Peikert for all his numerous invaluable contributions to the theoretical and practical implementation aspects of this project.

References

[1] C. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic encryption scheme. In Kenneth Paterson, editor, Advances in Cryptology – EUROCRYPT 2011, volume 6632 of Lecture Notes in Computer Science, chapter 9, pages 129–148. Springer, 2011.

[2] D. Micciancio. A first glimpse of cryptography's Holy Grail. Comm. ACM 53, 3 (March 2010), 96-96.

[3] V. Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings". In Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, chapter 1, pages 1–23. Springer Berlin / Heidelberg, Berlin.

[4] K. Rohloff, D. B. Cousins, "A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU." 2nd Work-shop on Applied Homomorphic Cryptography and Encrypted Computing (WAHC). Mar. 7, 2014.

[5] J. Alperin-Sheriff and C. Peikert. "Practical bootstrapping in quasilinear time". In Ran Canetti and JuanA. Garay, editors, Advances in Cryptology CRYPTO 2013, volume 8042 of Lecture Notes in Computer Science, pages 1–20. Springer Berlin Heidelberg, 2013.

[6] C. Gentry, S. Halevi, and N. Smart. "Homomorphic evaluation of the AES circuit." In Reihaneh Safavi-Naini and Ran Canetti, editors, Advances in Cryptology CRYPTO 2012, volume 7417 of Lecture Notes

in Computer Science, pages 850–867. Springer Berlin / Heidelberg, 2012.

[7] H. Cohen A Course in Computational Algebraic Number Theory. New York: Springer-Verlag, 1993.

[8] D. Cousins, K. Rohloff, C. Peikert, R. Schantz "Scalable Implementation of Primitives for Homomorphic EncRyption – FPGA implementation using Simulink" 2011 High Perfor-mance Extreme Computing Workshop Sep 21-22 2011, Lex-ington MA

[9] D. Cousins, K. Rohloff, C. Peikert, R. Schantz "An Update on SIPHER (Scalable Implementation of Primitives for Ho-momorphic EncRyption) – FPGA implementation using Simulink" 2012 IEEE Conference on High Performance Ex-treme Computing (HPEC) Sep 10-12 2012, Waltham MA

[10] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster Interleaved Modular Multiplication Based on Barrett and Montgomery Reduction Methods", IEEE Transactions on Computers, Vol. 59, No. 12, Dec 2010

Computing with Data Privacy:

Steps toward Realization

David W. Archer | Galois
Kurt Rohloff | New Jersey Institute of Technology

Two new cryptographic methods—linear secret sharing (LSS) and fully homomorphic encryption (FHE)—allow computing on sensitive data without decrypting it. LSS and FHE differ in speed, ease of use, computational primitives, and cost.

Users often don't trust computing environments such as shared clouds to perform computation on sensitive data. Only recently has it become possible to address this trust concern with general-purpose computation on encrypted data. In this article, we discuss two forms of such computation: linear secret sharing (LSS)[1] and fully homomorphic encryption (FHE).[2]

In LSS, a user or group of users, each with private data, encrypts the data and sends it to a group of untrusted servers. These servers share the computation without decrypting the data and return still-encrypted results. In FHE, a user encrypts data and sends it to a single untrusted server, which computes an encrypted answer and returns it to the user.

Computation time for both approaches is many orders of magnitude slower than computation "in the clear." In addition, LSS requires multiple servers to perform computation and significant communication bandwidth among them. FHE typically imposes significant expansion in ciphertext size relative to plaintext, which affects both memory utilization and network bandwidth.

We created prototypes including LSS- and homomorphic encryption (HE)–based variations of voice-over-IP (VoIP) teleconferencing systems using Amazon Elastic Cloud nodes to mix encrypted voice streams from iPhone handsets, an LSS-based email guard using regular expression search to determine which messages to transmit, and an FHE-based email guard using string comparison to filter email.

Protocols, Adversary Models, and Security Guarantees

Here, we describe our secure computation systems as well as applicable adversary models and security guarantees.

Linear Secret Sharing

In LSS, multiple proxies collaboratively compute a function on behalf of one or more clients.[1] Each client distributes to each proxy a share of its secret input. Each share is essentially random—a fixed linear function of the secret input and random values selected by the client. Thus, no proxies learn anything about the input. LSS works because its systems exhibit homomorphisms to mathematical structures of interest such as the integers, allowing parties holding shares to compute functions of secrets by arithmetically manipulating only their shares of those secrets.

 Copublished by the IEEE Computer and Reliability Societies

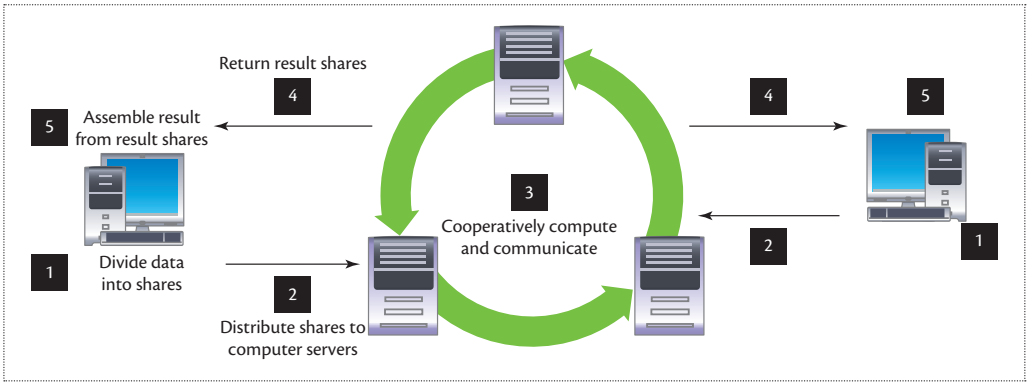


Figure 1. Our linear secret sharing (LSS) protocol. Each client encrypts its input with three cipher streams, producing a metashare (step 1). Each client transmits its metashare to the coordination server (not shown) over a secure channel, which in turn distributes these metashares to the three proxies (step 2). Each proxy computes its share from each metashare by decrypting the metashare using one cipher stream (that it and the client providing the metashare both know), and then performs the desired computation, communicating with other proxies as needed over secure channels (step 3). Each proxy encrypts its result share using a distinct cipher stream it shares with the client, and then sends it to the coordination server, which computes the XOR of all result shares into a result metashare and forwards this to clients (step 4). Each client decrypts the metashare to obtain the computation result (step 5).

As a simple example, suppose clients Alice and Bob agree to add secret inputs X and Y that they respectively hold. Assume X and Y are in $[0 \ldots 2^n - 1]$ for natural number n. Alice computes three shares of X by choosing random X_1 and X_2 from $[0 \ldots 2^n - 1]$, and then choosing X_3 such that $X = (X_1 + X_2 + X_3) \bmod 2^n$. Alice then distributes these three shares to the proxies over secure channels, such that each proxy holds one distinct share. Bob does the same for Y. The proxies add their shares, resulting in each proxy holding one of $X_1 + Y_1$, $X_2 + Y_2$, or $X_3 + Y_3$. Note that none of these result shares reveal anything about $X + Y$ to the proxies that hold them. The proxies send these result shares to Alice or Bob over secure channels. Bob or Alice then adds them together to obtain $X + Y$.

While communication from clients to proxies in typical LSS systems is direct, we found that having mobile clients distribute shares directly to each proxy resulted in substantial loading of client Wi-Fi channels. To address such Wi-Fi overload, we extended our applications' communication model to introduce an untrusted coordination server. Clients cryptographically combine all three shares they compute into a single metashare that's sent to the coordination server. This server, which we locate in a richer bandwidth environment along with the proxies, distributes the metashare to all three proxies, which compute their own shares from the metashare and preshared key material.

The core of our LSS system, ShareMonad, consists of a Haskell-embedded (www.haskell.org) domain-specific language (DSL) for expressing LSS computation, a compiler to transform ShareMonad code into abstract syntax trees suitable for interpretation, and a three-proxy LSS interpreter. Each proxy in a ShareMonad application runs this interpreter. Clients and coordination servers run application code that interoperates with the proxy code. Thus, each of our LSS applications consists of a composition of code running on clients, coordination server code, and ShareMonad code running on proxies.

Our LSS DSL provides operations including addition, subtraction, multiplication, unsigned division, comparisons, bitwise shift right, conversion between $[0 \ldots 2^n - 1]$ and bit vector representations, table lookups, and operations on bit vectors. ShareMonad protocols currently assume an honest but curious adversary: proxies are assumed to compute and communicate as agreed but might observe attached channels and local computations.

As Figure 1 shows, our LSS protocols typically proceed in several steps:

1. Each client encrypts its input with three cipher streams, producing a metashare.
2. Each client transmits its metashare to the coordination server (not shown) over a secure channel, which in turn distributes these metashares to the three proxies.
3. Each proxy computes its share from each metashare by decrypting the metashare using one cipher stream (that it and the client providing the metashare both know), and then performs the desired computation, communicating with other proxies as needed over secure channels.

4. Each proxy encrypts its result share using a distinct cipher stream it shares with the client, and then sends it to the coordination server, which computes the XOR of all result shares into a result metashare and forwards this to clients.
5. Each client decrypts the metashare to obtain the computation result.

We compute metashares and shares as follows. We distribute in advance a cryptographic key between each client *CL* and each proxy *A*, *B*, and *C*. This key seeds stream ciphers used to form metashares from input data. To compute the metashare X_m of secret X, a stream cipher is used to generate a random value R_A that undergoes bitwise XOR with X. *CL* repeats this process with the cryptographic key it shares with *B* and *C*, obtaining $X_m = X \text{ XOR } R_A \text{ XOR } R_B \text{ XOR } R_C$, which it sends to the coordination server to be forwarded to all three proxies. *A* uses the key it shares with *CL* to compute R_A, which it uses to compute its share of X, $X_1 = X_m \text{ XOR } X_A = X \text{ XOR } R_B \text{ XOR } R_C$. *B* and *C* similarly compute their shares X_2 and X_3, respectively. Note that X can trivially be recovered from these shares: $X = X_1 \text{ XOR } X_2 \text{ XOR } X_3$.

Once shares are computed, computation of the desired function proceeds on the proxies. In the case of addition, no communication among proxies is required: *A* computes result share $R_1 = X_1 + Y_1$, *B* computes $R_2 = X_2 + Y_2$, and *C* computes $R_3 = X_3 + Y_3$. Once computation is complete, *A*, *B*, and *C* send R_1, R_2, and R_3, respectively, to the coordination server, which computes the values' bitwise XOR, and forwards this single meta-result to the clients for final decryption.

Note that naively following this return transmission protocol would reveal all shares of the computation result to the untrusted coordination server. We avoid this security lapse by having *A*, *B*, and *C* encrypt R_1, R_2, and R_3, respectively, using keys shared between *A*, *B*, *C*, and the client to enable decryption by the client.

Some computations, such as $X \times Y$, require communication among proxies. $X \times Y = (X_1 + X_2 + X_3) \times (Y_1 + Y_2 + Y_3)$ involves not only locally computable terms such as $X_1 \times Y_1$ but also terms such as $X_2 \times Y_3$. These terms require that each proxy communicate its share to one other proxy. We follow the method that Dan Bogdanov and his colleagues described: sharing among proxies occurs in symmetric rounds, and each proxy adds new entropy to its share before sending that share to a neighbor.[5] Thus, even though proxies communicate their shares to other proxies, the communicated values don't allow those proxies to gain any knowledge of the original secret.

We ensure privacy in each portion of our protocol in Figure 1, except those that execute on the trusted client platforms:

- Passphrase sharing prior to computation is handled by well-known asymmetric cryptographic (public-key infrastructure) protocols. The Advanced Encryption Standard (AES) and the National Institute of Standards and Technology SP 800-90 standard provide cryptographically secure random numbers for creating shares.
- Transmission of metashares X_m from client to coordination server and onward to proxies is protected by the entropy added during creation of X_m.
- Local computation on the proxies is protected from observation because it's performed only on cryptographic shares. We prevent accumulation of too many shares on a single proxy by introducing additional entropy during the sharing process, as we described.
- Transmission from proxies to the coordination server is protected by encryption of result shares introduced by the proxies, which prevents the server from combining result shares to obtain the result in the clear. Transmission from the coordination server to the clients is also protected by this encryption.

Homomorphic Encryption

Like all secure encryption schemes, secure HE schemes make it intractable, under certain computational hardness assumptions, to recover information about plaintext from its encrypted ciphertext.[2,3] We use a representative approach to HE that employs a multi-dimensional lattice over a finite field. We use a vector basis to represent the lattice. Each plaintext input to the computation is encrypted to a ciphertext encoded as a vector—represented as a large matrix—not in the lattice. Security is based on the *closest-vector problem*: a known hard problem of finding the lattice vector with the least distance to a given vector—in our case, the ciphertext vector.

Computation on encrypted data proceeds by manipulating ciphertext matrix representations. However, encryption embeds noise into these representations. As computation proceeds, this noise grows. If too much noise accrues, decryption might identify the wrong lattice vector and thus return the wrong plaintext. We can decrease ciphertext noise by increasing the dimensionality of the ciphertext's matrix while maintaining security. Increasing the dimensionality of the matrix allows for more computation to be performed before too much noise accumulates but also results in computationally difficult manipulations of large matrices. Even with such noise reduction, noise still accumulates, ultimately limiting the depth of the computation available. FHE systems such as ours avoid this limitation by *bootstrapping*—periodically performing a cryptographic operation that resets the noise level without compromising security. Craig Gentry described an early form

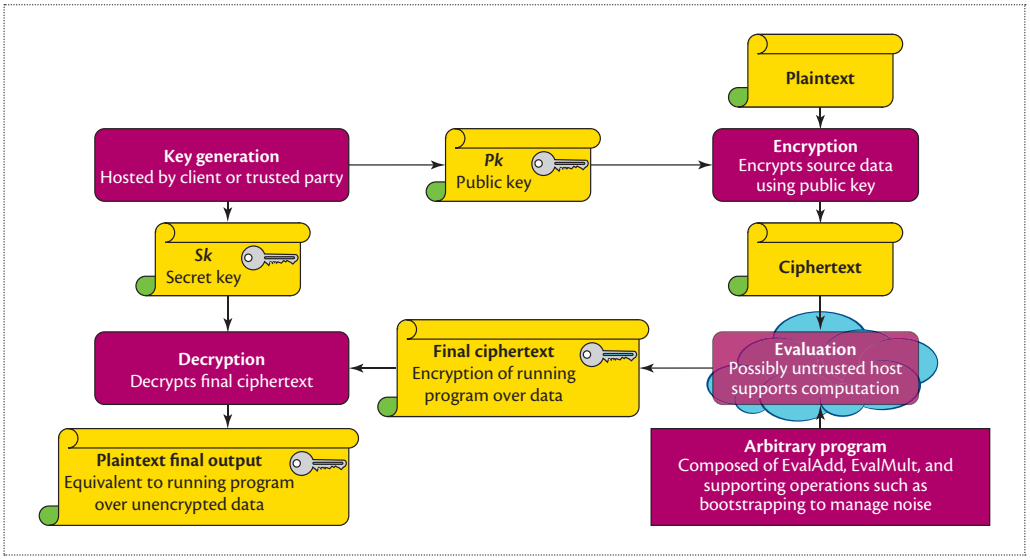


Figure 2. Dataflow in our fully homomorphic encryption (FHE) system. The key infrastructure (upper left) runs on a trusted host and uses the NTRU public-key approach to generate key pairs public key (*Pk*) and secret key (*Sk*). *Pk* is shared with a data source (on the right) that encodes plaintext messages as mod *p* integers and then encrypts the data using that key. A program source (lower right) provides a program, implemented as a Boolean circuit, to be evaluated over the encrypted data. The ciphertext, a public-key encryption of *Sk*, and the program are sent to a computation host (cloud, lower right). The result ciphertext is sent to the client (lower left) that decrypts it using *Sk* to obtain the plaintext result.

of bootstrapping and the resulting capability to perform arbitrary-depth secure computation.[2]

Figure 2 shows our FHE system's high-level dataflow. The key infrastructure on the upper left runs on a trusted host and uses the NTRU[4] public-key approach to generate key pairs consisting of a public key *Pk* and secret key *Sk*. The *Pk* is shared with a data source (on the right) that encodes plaintext messages as mod *p* integers and then encrypts the data using that key to generate the initial ciphertext. A program source (on the lower right) provides a program, implemented as a Boolean circuit, to be evaluated over the encrypted data. The initial ciphertext, a public-key encryption of the corresponding *Sk*, and the program are sent to a computation host (shown as a cloud, on the lower right). The resulting final ciphertext is sent to the client (on the lower left) that decrypts it using *Sk* to obtain the plaintext result. The protocols we use are secure against "honest but curious" adversaries such as an untrusted host performing the computation honestly while seeking to discover secret inputs.

Our FHE programs comprise two computational primitives: *EvalAdd* (addition) and *EvalMult* (multiplication). We use these primitives to construct operations for encryption, decryption, and bootstrapping. We implement modulus reduction, ring reduction, and key-switching operations to enable larger depth of computation before bootstrapping, without decreasing security. (In this article, the term *ring* refers to a mathematical ring over the integers.) We also implement specialized primitives, such as ring addition, ring multiplication, and Chinese Remainder Theorem (CRT), because manipulating ciphertexts in CRT representation is more efficient than in power basis representations.

Some early homomorphic systems relied on encoding a single bit of plaintext in each ciphertext. EvalAdd and EvalMult operations were thus simplified into Boolean XOR and AND operations but offered no computation parallelism. Ciphertext-to-plaintext expansion in such systems is quite large: in one of our early examples, the ciphertext expansion ratio was 2^{23}. In contrast, our system encrypts mod *p* integers ($p > 2$) instead of single bits, and we leverage single-instruction, multiple data (SIMD) approaches to pack multiple mod *p* integers into each ciphertext, thus computing parallel operations on these packed integers. Although this approach offers more efficiency, leveraging its inherent parallelism can make algorithm design challenging.

We use a variation of the double-CRT approach along with a residue number system (based on the CRT over the integers) to circumvent the problem of large ciphertext moduli and correspondingly large ciphertext size. For ring dimension *n*, each ciphertext is represented by an $n \times t$ matrix of *t* length–*n* integer vectors of mod q_i values for

pairwise coprime moduli q_i. This contrasts with some previous FHE systems that represent ciphertexts as a single integer vector mod Q, where $Q = q_1 * \bullet \bullet \bullet * q_t$. In our system, the number of moduli, t, grows to support the secure execution of larger programs, but the number of moduli $q_1, \ldots, q_t$ does not. With this representation, we securely represent ciphertexts as matrices of 64-bit integers yet still execute efficiently on commodity computing hardware that would make computation over the multihundred-bit or multithousand-bit single-vector integer representations used in previous systems infeasible.

The security level of lattice-based homomorphic encryption systems isn't often expressed in terms of the work factor used to describe security in typical cryptosystems. Instead, security is typically expressed as the *root Hermite factor* δ, a representation of the hardness of the closest-vector problem. A lattice-based encryption system becomes more secure as δ approaches 1. We selected the value $\delta = 1.007$ for our work, which corresponds roughly to the work factor required to crack AES 128-bit encryption.

The maximum depth of computation d that can be supported between bootstraps and the ring dimension n, which correlates directly to the length of ciphertext vectors, significantly impacts both δ and performance. We have found that with $n = 1{,}6384$ and $d = 16$, we achieve $\delta = 1.007$ while supporting significant computation, such as searching several pages of encrypted text for an encrypted keyword, between bootstraps. With $n = 16{,}384$ and efficient packing of ciphertexts, each ciphertext expands to between 10^3 and 10^6 times larger than the corresponding plaintext.

Our system runs in a compiled C environment auto-generated from Matlab implementations (www.mathworks.com/products/matlab). We use parallelism to take advantage of multicore processors in a Linux environment. At $\delta = 1.007$, we encrypt ciphertexts in less than 100 milliseconds in such environments and decrypt in approximately 1 millisecond. EvalAdd on ciphertexts takes several milliseconds, whereas EvalMult takes approximately 500 milliseconds and bootstrapping takes approximately five minutes.

Real-World Potential for FHE and LSS Implementations

Here, we present our prototype applications and their limitations.

VoIP Teleconferencing

Typical VoIP implementations don't provide end-to-end encryption. Instead, they rely on a trusted server to receive content from clients, decrypt that content, reencrypt it, and then distribute it to other clients. This trusted server is a weak point in securing VoIP communication.

Our teams independently developed LSS and FHE VoIP audio conferencing approaches that provide end-to-end security with performance suitable for three or more simultaneous users and high-quality audio. No prior work has demonstrated the application of these technologies to streaming applications such as VoIP. Both our prototypes use Apple iPhone 5s handsets, Amazon cloud-based virtual servers, and suitably modified open source VoIP client and server code.

LSS-based VoIP. Figure 3 shows our LSS VoIP architecture. Each iPhone runs a version of the Mumble VoIP client application (http://mumble.sourceforge.net) with the following modifications: Mumble audio processing samples the microphone at 16 Kbps and logarithmically compresses this to a standard 8-bit μ-LAW floating-point representation.[6] We added encryption for turning each sample into a metashare by computing XOR of each sample with elements drawn from three AES 128-bit counter-mode cipher streams seeded from pre-placed passphrases. The network interface packs 1,440 sample metashares (90 milliseconds of audio data) into each transmitted network packet.

As Figure 3 shows, each client creates and then sends each metashare packet via Wi-Fi (802.11ac) to an Apple Airport Extreme wireless access point, which forwards it to a virtualized coordination server in the Amazon Elastic Cloud Service (ECS). This virtual machine runs a modified version of uMurmur (https://code.google.com/p/umurmur) to handle user session management and audio stream routing. Our uMurmur variant distributes each client audio packet to each of three proxies, gathers result share packets from those proxies after computation, computes XOR on the result shares together sample-wise, and sends the resulting metashare to clients for decryption.

Our proxies, which are also virtual machines hosted in the Amazon ECS, run our ShareMonad audio processing application. Each proxy recreates one of the three entropy streams and uses this to compute its share of each sample from the received metashares. Collectively, the proxies obliviously decode each logarithmically compressed audio stream to a linear, integer representation; mix all decoded audio streams together; clip the resulting audio signal; and recompress the result for distribution.

This computation repeats for each participating client, omitting that client's audio stream so users don't hear their own voices. Each audio stream result share is sent back to the coordination server, where it undergoes XOR with shares from other proxies and is then sent to client handsets for decryption and playback.

A hand-optimized approach required 12 seconds of processing per 1,440-sample block for four users,

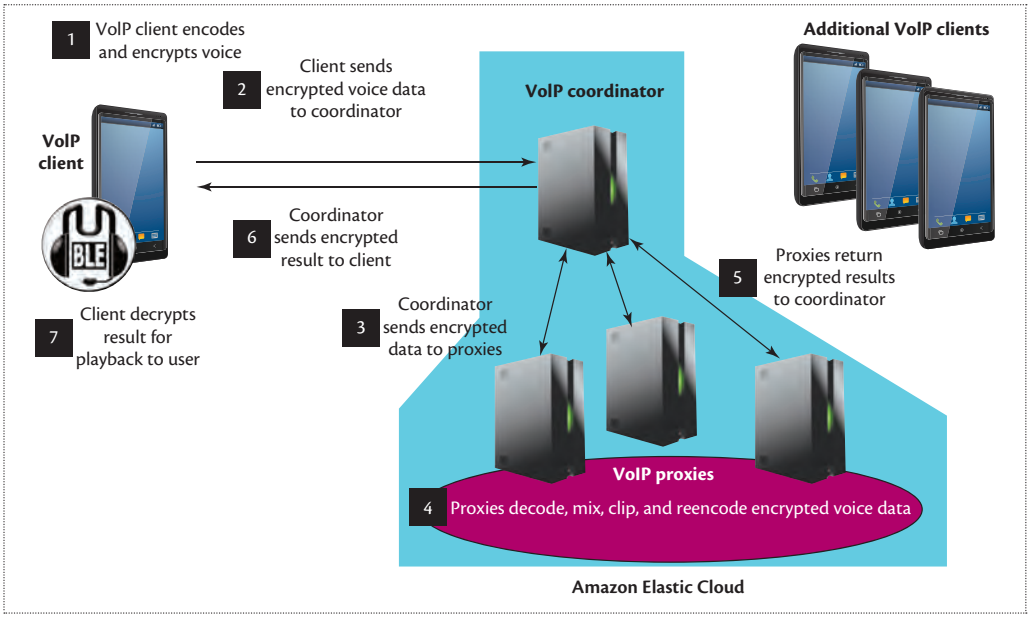


Figure 3. LSS-based voice-over-IP (VoIP) system architecture. iPhone 5s VoIP clients sample audio input at 16 Kbps, encode samples to a standard 8-bit μ-LAW floating-point representation, and encrypt the resulting encoded samples using three Advanced Encryption Standard 128-bit counter-mode cipher streams. Clients send packets of 1,440 encrypted samples (90 ms of audio) over Wi-Fi 802.11ac and through the Internet to proxy servers in the Amazon Elastic Cloud that decode, add, and clip the sample streams without decrypting them. The resulting combined audio stream is reencrypted and sent back to the clients for decryption and playback.

exceeding the 90-millisecond limit required to maintain processing at streaming rates. Applying an LSS index lookup over a public table[7] of precomputed results for the decode-mix-clip-encode function let us reduce this delay to 25 milliseconds, allowing sufficient time to meet the 90-millisecond goal and compensate for network delays between handsets and servers. With this optimization, we achieved streaming throughput for up to four voices at 16 Kbps audio rates, enabling users to communicate clearly.

We used 16-core (C3 size) Amazon cloud servers as proxies, resulting in roughly 80 percent CPU utilization. In contrast, plaintext processing at this performance level requires only a small portion of a single CPU core. Memory use was small and not a constraining factor. Network bandwidth available in our Amazon cloud instances was sufficient with no special optimization.

In the absence of collusion among the proxies, our solution provides two layers of AES 128-bit security at each proxy. Each proxy receives metashares encrypted with three AES 128-bit counter-mode cipher streams yet has access to only one of these cipher streams. Thus, adversaries observing from any one proxy can learn nothing of the plaintext audio samples used as input. Adversaries observing from the coordination server can learn nothing about the input from the metashares it conveys, because that server holds none of the cipher streams used for encryption and decryption. Because each proxy adds new layers of encryption (using cipher streams to which the coordination server has no access) to the result shares it sends back to the coordination server, that server similarly can learn nothing of the computation result.

FHE-based VoIP. We developed an FHE-based approach to secure VoIP teleconferencing that requires only a single proxy. This advance is built on a vocoder technology that takes voice samples from each client as input and encodes those samples as vectors of integers that are then encrypted. This vocoder is linear and can be used with an additive HE scheme to provide an encrypted VoIP teleconferencing capability. Encoded voice samples are encrypted at each iPhone client with the client's public key, using the additive HE scheme.

For our prototype, all clients use the same key, because our focus is on demonstrating the practical feasibility of an FHE computation rather than on well-understood security concerns. The resulting ciphertexts are sent to a VoIP mixer that queues and adds the ciphertext from the clients without decrypting the data or sharing keys. The resulting added ciphertext is sent

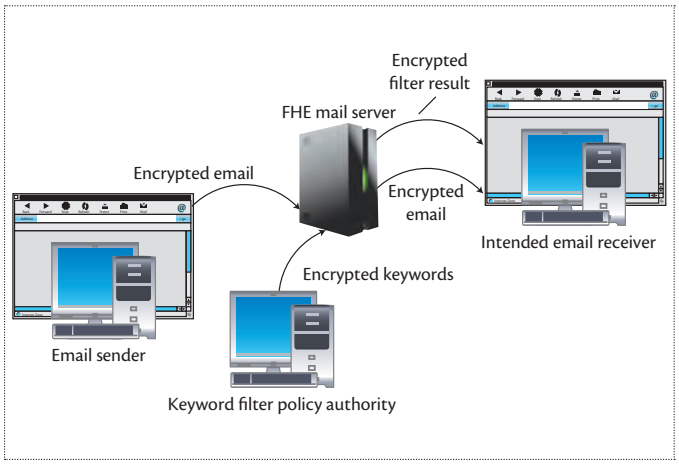


Figure 4. Email border guard system architecture. In the LSS version of the border guard, the mail server connects to three proxies (not shown in the figure). An email client plug-in computes a sent message's metashare using key material shared a priori with the proxies and sends the metashare to the mail server, which distributes it to the proxies. The proxies collectively search the encrypted message, producing shares of a Boolean indication of a regular expression match. In the FHE version, the client homomorphically encrypts the message and sends it to a single proxy that searches the encrypted message for matches with a predefined set of strings, also producing a Boolean indication of a match. The mail server (in the LSS case) or the client (in the FHE case) receives the computation result and uses it to determine whether to forward the message.

back to the clients. When decrypted with the clients' private key using the additive homomorphic decryption scheme, decoded using our decoding scheme, and played back to the clients, the resulting audio is a mix of all the clients' audio streams.

Our FHE-based VoIP uses a prototype architecture similar to the LSS-based VoIP teleconferencing capability, but with lower end-to-end latency. When we ran this system with a server in Virginia and clients in Massachusetts, the total latency was on the order of 80 milliseconds, with the latency roughly split among communication, encryption, and decryption. The mixing latency was nearly trivial, taking less than 1 millisecond.

With our FHE-based approach, no keys are stored on the teleconference server, so privacy is preserved even if adversaries view all communication links and server operations. Trust in the communication links or teleconference server isn't required to provide privacy. The security level provided in the current demo is roughly at the level of AES 128-bit encryption, but parallels between the security levels of our encryption scheme and other current standards aren't exact. We can increase our teleconference capability's security level arbitrarily at the expense of bandwidth requirements or voice quality by modifying the sampling rate and dynamic range of the sampled voice data.

Email Border Guards

Providing privacy using email encryption and achieving information security using trusted-party email filtering at network boundaries are mutually exclusive goals. Either email must be decrypted to verify compliance to InfoSec policies (compromising privacy), or those policies must be enforced by each user prior to message encryption (compromising trust in filtering). We explored solutions to this problem by studying applications in which transaction throughput is important. In our solutions, users encrypt email messages on their trusted computer. The messages are sent to an untrusted mail server for forwarding to a destination. This mail server also acts as a *border guard*, checking each email message for certain content and passing it on to its destination only if that content is absent. The border guard performs this content checking without decrypting the messages.

LSS-based regular expression search email guard. We use the Claws email client and a typical email server, along with plug-ins to each via standard APIs, to search each outbound encrypted email for occurrences of text that match a set of prespecified regular expressions, forwarding messages that do not include such matches and rejecting those that do.

Figure 4 shows our system architecture. In the LSS version, the mail server connects to three proxies that perform the LSS computation (not shown in the figure). When a user sends a message, a plug-in to the Claws client computes the message's metashare using key material shared a priori with the proxies. The email client sends the metashare to the mail server, where a Milter (www.milter.org) plug-in distributes it to the proxies, each of which derives its share. The regular expression set is compiled into a Boolean circuit and distributed to the proxies in advance. The proxies collectively compute the regular expression search on the message, using an adaptation of a mechanism that transforms regular expressions into finite automata.[8] Each server produces one share of the Boolean indication of whether any regular expressions match against any portion of the encrypted message corpus. Our Milter plug-in combines these shares to obtain a plaintext Boolean answer, which it passes to the mail server. The mail server then accepts and forwards the message, or it drops the message and informs the sender's client, as appropriate.

We performed several experiments on this system, optimizing the resulting Boolean circuit to consider different numbers of regular expression characters. Processing 16 message characters at a time was the point of diminishing returns. For a typical 1-Kbyte email ASCII message and a set of regular expressions that roughly represents classification markings that might be used in

a government setting, checking an email message took approximately 90 seconds using quad-core, 3-GHz Intel architecture blade servers as proxies. CPU utilization averaged approximately 90 percent during processing, and memory utilization was minimal.

FHE-based encrypted keyword search email guard. We developed a prototype FHE application that searches for encrypted keywords in encrypted text. This method relies on a homomorphic string comparison operation that's repeated for all keywords in all locations of an encrypted message. As in the LSS method, we imported this technology into an email guard–type scenario to provide outsourced email filtering based on email clients' keywords of interest. Because the result of the string comparison is only available to the mail server in encrypted form, our protocol sends the encrypted result back to the client, where it's decrypted to reveal whether the message should be sent. Thus, our prototype assumes an honest sender and requires an extra round-trip between client and server. Figure 4 shows a sketch of this technology.

We're currently running this implementation at a low security level ($\delta = 1.08$) to enable the email system to be interactive with fast response times. Our initial implementation uses a ring dimension of $n = 512$ and encrypts emails with a supported depth of computation $d = 12$. This results in an effective ciphertext modulus q represented with 430 bits. With these parameter configurations, we can sort over encrypted paragraph-long emails with five- to six-character words in less than a minute. Result decryption runs in a matter of seconds.

We could tune this FHE-based email guard to an extremely secure setting ($\delta = 1.0055$ or less) using our current implementation with a similar depth of computation. We would choose a ring dimension of 16,384 and an effective ciphertext modulus Q represented with 521 bits. Encryption runtime at these settings is on the order of minutes, encrypted message filtering would take hours on a nonparallelized server, and decryption would take a matter of seconds.

In a world in which Bob and Alice need to work together but are no longer comfortable sharing their secrets, or where Alice needs Charlie's help to process data but feels uncomfortable with Charlie (or the ever-lurking Eve) seeing the data, secure computation holds promise. However, secure computation methods differ; each has its distinct tradeoffs, security models, and caveats. Our experiments show that some practical applications are emerging, but substantive work remains to be done to make secure computation practical for broad classes of applications. ■

Acknowledgments

We greatly appreciate the contributions of Drew Dean of SRI International in editing and reviewing this manuscript. The Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory (AFRL), and the Office of Naval Research (ONR) under contracts FA8750-11-C-0098 and N00014-11-C-0333 sponsored this work. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the US Department of Defense or the US government. Distribution statement "A" (approved for public release, distribution unlimited).

References

1. R. Cramer, I. Damgard, and U. Maurer, "General Secure Multi-party Computation from Any Linear Secret-Sharing Scheme," *Proc. 19th Int'l Conf. EuroCrypt*, 2000, pp. 316–334.
2. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proc. 41st Ann. ACM Symp. Theory of Computing* (STOC 09), 2009, pp. 169–178.
3. Z. Brakerski and V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages," *Proc. 31st Ann. Conf. Advances in Cryptology*, 2011, pp. 505–524.
4. J. Hoffstein, J. Pipher, and J. Silverman, "NTRU: A Ring-Based Public-Key Cryptosystem," *Proc. 3rd Int'l Symp. Algorithmic Number Theory*, LNCS 1423, Springer, 1998, pp. 267–288.
5. D. Bogdanov et al., "High-Performance Secure Multiparty Computation for Data Mining Applications," *Int'l J. Information Security*, vol. 11, no. 6, 2012, pp. 403–418.
6. *Pulse Code Modulation (PCM) of Voice Frequencies*, Int'l Telecommunication Union, ITU-T Recommendation G.711, 1993.
7. J. Launchbury et al., "Application-Scale Secure Multiparty Computation," *Proc. 23rd Ann. European Symp. Programming*, LNCS 8410, 2014, pp. 8–26.
8. S. Fischer, F. Huch, and T. Wilke, "A Play on Regular Expressions: Functional Pearl," *ACM SIGPLAN Notices*, vol. 45, no. 9, 2010, pp. 357–368.

David W. Archer is a research lead at Galois. His research interests include information provenance and trustworthiness, and information assurance. Archer received a PhD in computer science from Portland State University. Contact him at dwa@galois.com.

Kurt Rohloff is an associate professor of computer science at the New Jersey Institute of Technology. His research interests include homomorphic encryption, large-scale distributed computing, and secure computation. Rohloff received a PhD in electrical engineering and computer science from the University of Michigan. Contact him at kurt.rohloff@njit.edu.

Scalable, Practical VoIP Teleconferencing with End-to-End Homomorphic Encryption

K. Rohloff, D. Sumorok, D. Cousins, Raytheon BBN Tech.

***Abstract*—In this paper we present a practical new approach to scalable and secure VoIP teleconferencing where data remains encrypted even at the VoIP teleconferencing server. Until now, VoIP teleconferencing services have required either 1) all teleconferencing clients to maintain point-to-point links with other clients or 2) a teleconferencing server which can access and manipulate all VoIP streams unencrypted, in the clear. We present a new approach which uses a teleconferencing server which manipulates only encryptions of the VoIP streams, thus avoiding the respective scalability and security issues of previous classes of approaches. Our new approach relies on recent advances in practical homomorphic encryption to provide end-to-end encryption. Voice data is sampled, encoded and encrypted at the VoIP teleconference clients, sent over a generic network such as the open Internet to the encryption-enabled server, and mixed at the server without decrypting the VoIP data. The encrypted result of the mixing is sent back to the clients for decryption, decoding and playback to the users. The homomorphic encryption basis of our secure VoIP teleconferencing capability is a modification of NTRU, and can provides fully homomorphic and post-quantum features, but we only use additive homomorphic capabilities in this work. We discuss our working prototype of this secure, practical VoIP teleconferencing capability running on iOS clients and the lowest-cost Amazon AWS server. Our prototype provides full-duplex 100kbs throughput and average 90ms latency, which is higher quality than many commercial VoIP services such as Skype and GoToMeeting, besides being much more secure. We present the design of our VoIP system, with a particular focus on the VoIP encoding/decoding scheme and homomorphic mixing operations. These encoding/decoding operations and the encrypted homomorphic mixing, coupled with an efficient, usable implementation compatible with commodity hardware are the primary advance we have been able to leverage to enable secure VoIP teleconferencing with end-to-end encryption. We present experimental results to show the scalability, performance and voice quality trade-offs of our design and implementations when used over local-area, national and intercontinental distances.**

I. Introduction

There has been an unmet technological need to provide a scalable capability for multiple geographically distributed people need to simultaneously converse as a group at the same time over data networks. This need, until know, has been partially served by either have physically secure dedicated point-to-point communication links as provided by dedicated circuits, or through physically unsecured point-to-point communication links which are made secure with point-to-point encryption technologies [1], [2].

These prior approaches to secure communication are either not scalable or have not adequately addressed several important vulnerabilities. Physically secure communication links are not feasible over broad geographic areas such as for trans-continental and inter-continental communication. Point-to-point encryption solutions do no scale because when there are more than a handful of participants in a teleconference call, a large number of point-to-point communication links are practically difficul to setup and maintain, often leading to latency issues which would degrade the quality of user experiences. For these reasons, there is a need for such a technology to provide scalable, secure and practical teleconferencing services which can be used to host multi-party negotiations, planning, education and information distribution of a sensitive nature.

The needs for scalable, secure and practical teleconferencing services has been partially met with Voice over IP (VoIP) teleconferencing technologies where users can converse with one another by encoding data for transmissions between users over IP data networks. VoIP provides a fundamentally scalable and practical approach to teleconferencing, especially with the advent of global packet-switched information networks. Unfortunately, existing VoIP teleconferencing capabilities such as GoToMeeting, Skype and Mumble among others have not been both scalable and secure against data leaking to adversaries who wish to snoop on private or even proprietary group communication. These technologies have been vulnerable to man-in-the-middle attacks of various types [3].

Although modern VoIP teleconferencing technologies have been good at protecting data in transit between clients and a server, all bets are off when the data reaches a server where it needs to be mixed. The majority of widely used existing VoIP teleconferencing capabilities require a central VoIP server to mix all of the VoIP signals from clients which are then sent back to the clients. Until now this has required the VoIP server to have access to unencrypted VoIP data. That is, the VoIP mixing operation, which merges the VoIP streams from the clients, has until now needed to be performed in the clear, on unencrypted VoIP data. This creating a possible opportunity for adversaries to snoop on otherwise protected VoIP data if the adversaries gain access to the VoIP server.

The mixing of VoIP data in the clear is adequate when the VoIP server is fully trusted by all participants. However, VoIP teleconferencing servers are often hosted in a semi-secure environment, such as by commodity cloud providers such as Amazon AWS or Microsoft Azure. Some users, such as in less technologically developed regions of the globe, might not have access to low-cost cloud environments, requiring the deployment of VoIP servers on local hardware which are less

Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflec the officia policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited.)

Note: Distribution statement and contract citation are contractually required by the author's sponsors for all review and publishing activities, including for double-blind reviews.

secure against compromise or seizure.

The limitation of required server trust has until now prevented the use of VoIP teleconferencing technologies from being used in less-than-fully trusted situations, such as in the cloud where the underlying server hardware and the cloud providers cannot be fully trusted to not leak information. This induces an unfortunate trade-off of for this architecture of either requiring all participants to maintain group conversations in the clear in untrusted environments or paying a higher cost of maintaining access to a trusted VoIP server if secure teleconferencing is needed.

Taken together, these technological deficiencies point to a need for a VoIP teleconferencing capability where VoIP data can never decrypted except on the clients which have access to decryption keys. Thus, unlike previous VoIP attack analyses which focus on signaling attacks [4], [5] during VoIP call set-up, we are particularly interested in protecting against man-in-the-middle attacks which involve compromise the VoIP server.

In this paper we present a secure, scalable and practical method to protect against the leakage of sensitive VoIP teleconferences even on VoIP teleconference servers that have been fully compromised. We provide a method for parties to have privacy-preserving teleconferences where communication privacy is maintained despite all communications of the clients being observed during the teleconference, even at the teleconference mixer. The basis of our approach is a method for additive homomorphic encryption such that all clients have a common private key. The clients encode their voice samples with a additive encoding scheme, encrypt their encoded voice data with an additive homomorphic encryption scheme, send their encrypted voice samples to a mixer which performs an encrypted homomorphic addition on the encrypted voice and sends the results back to the clients. The clients then decrypt, decode and play back the result. Our scheme relies on the pre-sharing of a common private key for an additive homomorphic encryption scheme, but it is possible in principle to practically generalize beyond this pre-shared key design.

We implemented this scheme to run on commodity iPhone clients and the current lowest-cost Amazon EC2 server. Our capability provides end-to-end encryption of all VoIP data from the VoIP clients hosted on the phones with no decryption at the server. This implementation is secure, relying on a post-quantum encryption scheme which protects even against quantum computing attacks on the encrypted data. This capability also provides relatively high sound quality with full-duplex 100kbs data rates. This initial implementation is intended to be a proof-of-concept capability, with the possibility of improving upon this technology with existing key management technologies [6] and session initiation technologies [7] with additional engineering investment and little or no research risk.

Our new approach relies on recent advances in practical homomorphic encryption to provide end-to-end encryption. With this approach, encrypted VoIP data is mixed on a VoIP teleconference server without decrypting data at the server or sharing decryption keys with the server. We present our working prototype of this secure, practical VoIP teleconferencing capability running on iOS clients and the lowest-cost Amazon AWS server and provide experimental analyses of this implementation. Our innovation is in the VoIP encoding/decoding scheme and homomorphic mixing operations, coupled with an efficient, usable implementation compatible with commodity hardware.

The paper is organized as follows. Section II discusses the design goals of our encrypted VoIP teleconferencing system. Section III discusses the overall design of our end-to-end encrypted VoIP teleconferencing capability. Section V discusses the engineering trade-offs associated with parameter selection. Section VI discusses how we implemented our design to run on iOS clients and Linux servers. Section VII presents experimental results of deploying our end-to-end encrypted VoIP teleconferencing capability on the open Internet. Section VIII discusses related work on relevant technologies. Section IX presents a discussion of our capability and ongoing work.

II. Design Goals

We identified several design goals and metrics of performance with which to evaluate and reason over our end-to-end encrypted VoIP teleconferencing designs and implementations. Our primary high-level design goals and metrics are:

1) **Sound Quality**: The end-to-end encrypted VoIP teleconferencing capability should provide sound quality at least as good as a Public Switched Telephone Network (PSTN), preferably with full-duplex.
2) **Latency**: The end-to-end encrypted VoIP teleconferencing capability should provide an end-to-end latency ideally of less than 100ms for trans-continental VoIP teleconference session, a generally accepted reasonable latency for VoIP technologies, but more latency is acceptable for inter-continental operations.
3) **Scalability**: The end-to-end encrypted VoIP teleconferencing capability should be able to support four people speaking simultaneously while ten's of participants listen without degradation in sound quality or latency.
4) **Secure**: The end-to-end encrypted VoIP teleconferencing capability should provide an encryption work factor roughly at least as good as the work factor for AES-128. This means that the VoIP data, when encrypted, should require at least as much computational effort to obtain the unencrypted data without a key as is needed for AES-128, a commonly used point-to-point secure encryption technology.
5) **Resource Efficient**: FHE schemes have been known to require encrypted data which is much larger than the original source data. Early schemes provided a ciphertext expansion of several orders of magnitude larger than the source data. The end-to-end encrypted VoIP teleconferencing capability should ideally require less than an order of magnitude ciphertext expansion.
6) **Wide Geographic Area**: The end-to-end encrypted VoIP teleconferencing capability should operate with users and the VoIP mixing server over a wide geographic area, ideally trans-continental if not inter-continental without an unacceptable degradation in sound quality or latency.
7) **Portable**: The end-to-end encrypted VoIP teleconferencing capability should be easily ported to other client and server types.

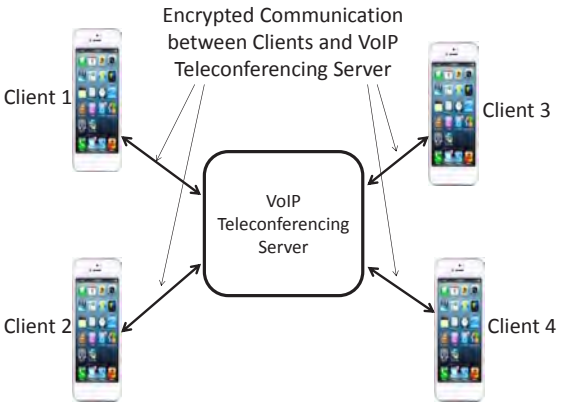


Fig. 1: High-Level VoIP Teleconferencing Design

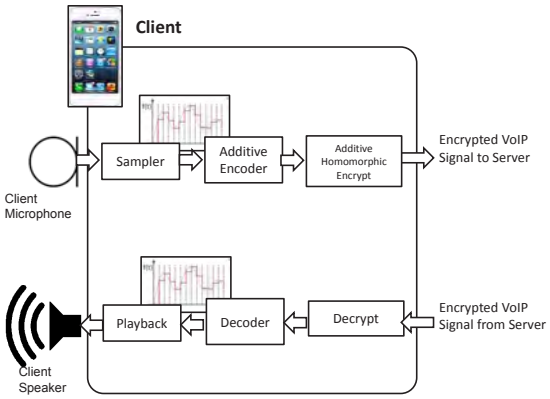


Fig. 2: High-Level Data Client Internal Data Processing

8) **Easily Deployable**: The end-to-end encrypted VoIP teleconferencing capability should be easy to deploy, such as with small binaries.
9) **Usable**: The end-to-end encrypted VoIP teleconferencing capability should be intuitive and easy to use.
10) **Extensible**: The end-to-end encrypted VoIP teleconferencing capability should be easy to modify to add additional and more advanced functionality at a later date.

III. Teleconferencing Architecture

Figure 1 shows a high level example illustrative application of this privacy-preserving VoIP teleconferencing technology with end-to-end encryption. Each of the clients samples users' voice data, encodes it, encrypts it and sends the result to the VoIP mixer. The mixer sends a result back which is then decrypted, decoded and played back to the clients' users. Any encryption system could be used that supports an additive homomorphism which could be implemented in a practical manner. A representational scheme that supports additive homomorphisms is NTRU which can be made both Somewhat Homomorphic (SHE) and Fully Homomorphic (FHE) in addition to additive homomorphic.

Our approach uses a shared secret key, but more general designs are possible that generalize beyond this initial shared secret key design. Input voice streams from clients are sampled and homomorphically encrypted using a clients public key. The encrypted voice samples are sent to an FHE-enabled VoIP server that does not have access to encryption keys. The VoIP server combines and balances the encrypted audio feeds. The combined output is then forwarded to the client handsets, where it is decrypted and played back for the user. Our FHE-based solution processes streaming audio at 10 kBytes/s per voice.

The output of the processing is sent to the client, where it is decrypted using the clients private key. No keys are stored on the teleconference server, so privacy is preserved even if an adversary views all communication links and operations on the server. No trust of the communication links or teleconference server is required to provide privacy. The level of security provided in the current prototype is roughly at the level of AES-128, but parallels between the security levels of the encryption scheme and other current standards are not exact. We can increase the security of our teleconference capability to be arbitrarily higher at the expense of voice quality by decreasing sampling rate and dynamic range.

Figure 2 shows how the clients support data f ows internally. In the top of the diagram, data from the microphone is sampled and fed to the encoder, encrypted using an additive homomorphic encryption scheme and sent to the mixer. As seen in the bottom of the f gure, the result returned from the mixer is decrypted, decoded and played back over a speaker.

Figure 3 shows how the VoIP mixer takes encrypted input from various clients and returns a common output. For a representational VoIP system with clients $(c_1, c_2, c_3, \ldots, c_m)$, a client c_i would want $(c_1 + c_2 + \ldots + c_{i-1} + c_{i+1} + \ldots + c_m)$. This summation can be performed in a tree fashion as illustrated in Figure 3. For our representational NTRU scheme, the ciphertexts are vectorized in blocks of m, and all additions are performed modulo some large integer q pre-specif ed by the key generator.

Our encoder/decoder is additive so that we can rely on an additive homomorphism such as the EvalAdd operation to mix VoIP signals. Because we require only an eff cient secure EvalAdd operation to support encrypted VoIP mixing, our design builds on the recent eff cient FHE design and implementation discussed in [8]. We simplif ed this prior work such that we remove the ability to support EvalMult operations. As such, because we only need to support much smaller circuits, we do not need the parallelism capabilities as discussed in [8] for our VoIP application and integration with the existing Mumble/Murmur open-source VoIP systems. We also use much smaller parameters than the designs advocated in [8] because we require much more greatly reduced functionality. Thus, the basis of our encryption approach is a special limited version of FHE called Additive Homomorphic Encryption which allows an untrusted computation host to compute the encrypted sum of encrypted integers.

A. Client Vocoder

We have a developed a vocoder technology which takes voice samples from a client and encodes the voice samples as

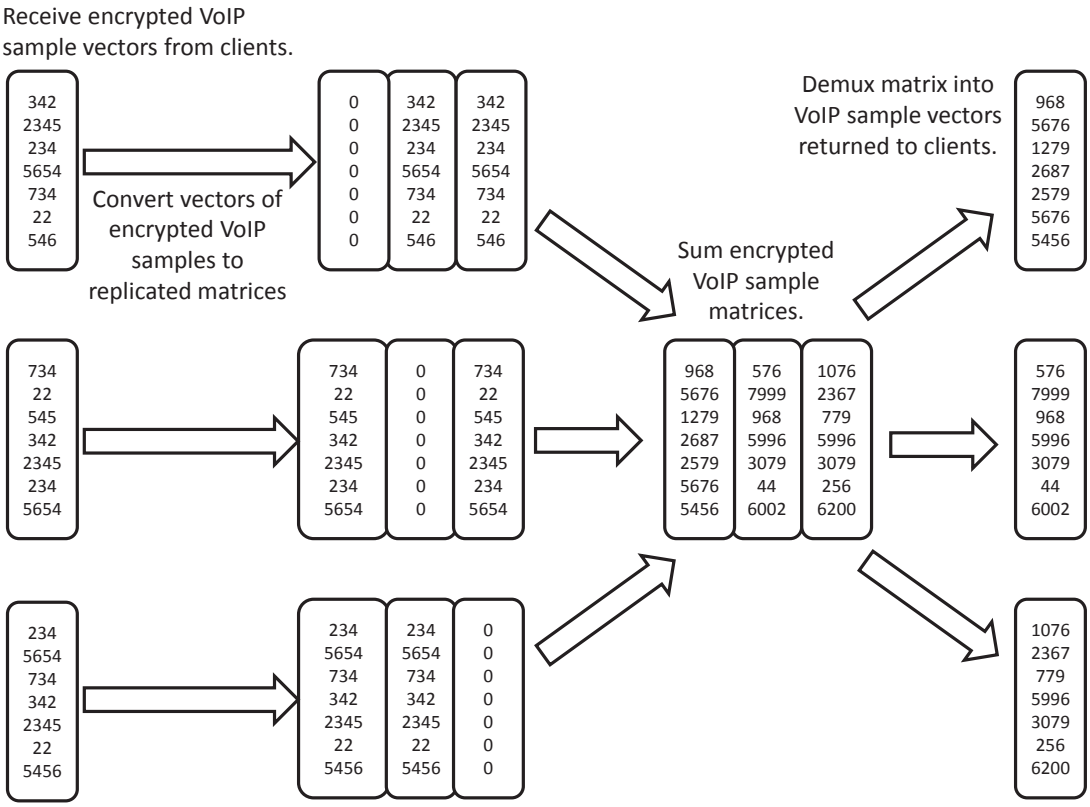


Fig. 3: Encrypted VoIP Server Mixing for Three Clients

vectors of integers. This vocoder is linear so that it can be used, for example, with an additive homomorphic encryption scheme to provide an encrypted VoIP teleconferencing capability. In this example, the encoded voice samples are encrypted using the additive homomorphic encryption scheme. This operations are performed on multiple clients. The resulting ciphertexts are sent to a VoIP mixer which queues and adds the ciphertexts from the clients. The resulting added ciphertext can be sent back to the clients. When decrypted with the additive homomorphic decryption scheme, decoded using our decoding scheme and played back to the clients, the resulting audio is a mixing of the audio from the clients.

Our encoding goal is to convert a length-m data frame of y-bit VoIP samples into a length-n frame of integers with the property that $Encode(input_1) + Encode(input_2) = Encode(input_1 + input_2)$. As seen in the left hand side of Figure 4, we split the length m sample input into multiple blocks of $n = 2^(floor(log2(m)))$-length vectors and a single $mod(m-n)$-length vector if $mod(m-n) > 0$. The first step is to shift the samples so they are centered around 0, mod 2^y. For the zth block of samples, we multiply the integers in this block by $2^(y+z-1)$. We also pad the $m-n$ block of samples with $2n-m$ 0s so this vector is n samples long. As seen on the right hand side of Figure 4, we sum these vectors. These operations are all highly efficient as they only involve splitting vectors, multiplication be two and bitwise concatenation, which are all extremely efficient to implement. This result is the encoded vector and has the property that $Encode(input_1) + Encode(input_2) = Encode(input_1 + input_2)$. This encoded data is subsequently used for encryption.

Figure 5 shows our decoding process. On the right hand side of this figure we take the input vector. We make copies of this block and perform an integer division by $2^(y+2*z-1)$ for the zth block. We then concatenate these vectors and return the result. Like for the encoding operation, these operations are all highly efficient as they only involve splitting vectors, multiplication be two and bitwise concatenation, which are all extremely efficient to implement.

IV. Homomorphic Encryption and Key Generation

In this subsection we describe the additive homomorphic cryptosystem we use to construct the end-to-end encrypted VoIP capability built on [8]. This cryptosystem is very similar to the NTRU system [9], though it was not until recently that its homomorphic properties were noticed independently by López-Alt et al. [10] and Gentry et al. [11]. A more general version of this cryptosystem was discussed in [8], but we discuss here a more limited version of the cryptosystem of [8] which is simplified for more efficient end-to-end VoIP encryption.

The discussion of this simplified cryptosystem has a high degree of overlap with the more general cryptosystem. Our

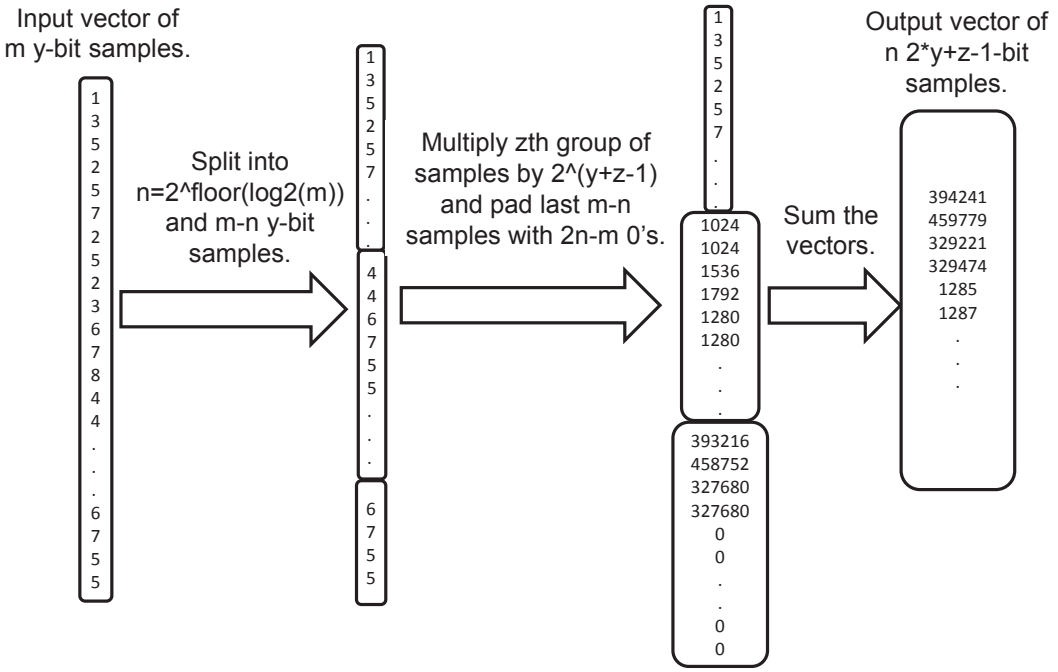


Fig. 4: Encrypted VoIP Encoding

simplif cations reside primarily in the encryption and decryption operations, but we include the full key generation and evaluation addition operations which are also modif ed, but to a lesser extent, for the sake of completeness. The modif cations are primarily in the avoidance of any ciphertext decomposition to parallelize operations when the ciphertext modulus $q > 2^{64}$. We have found that we can parameterize the cryptosystem to support the vector addition of adequately large plaintext vectors such that requiring a larger ciphertext modulus is not needed. As such, because we can limit ourselves to 64-bit operations, our simplif ed cryptosystem can be implemented to run highly eff ciently on native 64- and 32-bit processors without the parallelism advances obtained in [8] for more eff cient more general computations.

The simplif ed cryptosystem is based around the manipulation of power-of-2 cyclotomic rings for ease and eff ciency of implementation. For ring dimension n which is a power of 2, def ne the ring $R = \mathbb{Z}[x]/(x^n+1)$ (i.e., integer polynomials modulo $x^n + 1$). For a positive integer q, def ne the quotient ring $R_q = R/qR$ (i.e., integer polynomials modulo $x^n + 1$, with coeff cients from $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$).

For the cryptosystem the message space is R_p for some integer $p \geq 2$. We use a mod-q Chinese Remainder Transform (CRT) representation of elements to provide fast addition. These CRT representations are discussed extensively in [12]

The basic operations of the scheme are as follows:

- KeyGen: choose a short $f \in R$ such that $f = 1 \bmod p$ and f is invertible modulo q, and a short $g \in R$. Output the public key $pk = h = g \cdot f^{-1} \bmod q$ and the secret key $sk = f$.
  We choose the short elements f and g from centered discrete Gaussians. E.g., we can let $f = p \cdot f' + 1$ for some Gaussian-distributed f'. Note that such an f will have expectation (center) 1.
- Enc$(pk = h, \mu \in R_p)$: choose a short $r \in R$ and a short $m \in R$ such that $m = \mu \bmod p$. Output $c = p \cdot r \cdot h + m \bmod q$.
  Concretely, m can naively be chosen as $m = p \cdot m' + \mu$ for a Gaussian-distributed m', but again, such an m is not zero-centered.
- Dec$(sk = f, c \in R_q)$: compute $\bar{b} = f \cdot c \bmod q$, and lift it to the integer polynomial $b \in R$ with coeff cients in $[-q/2, q/2)$. Output $\mu = b \bmod p$.

The additive homomorphic operations are def ned as follows:

- EvalAdd(c_0, c_1): output $c = c_0 + c_1 \bmod q$.

V. Parameter Selection Tradeoffs

We need to choose parameters for both the vocoder and the cryptosystem so that:

- VoIP signal data is encoded into VoIP plaintext.

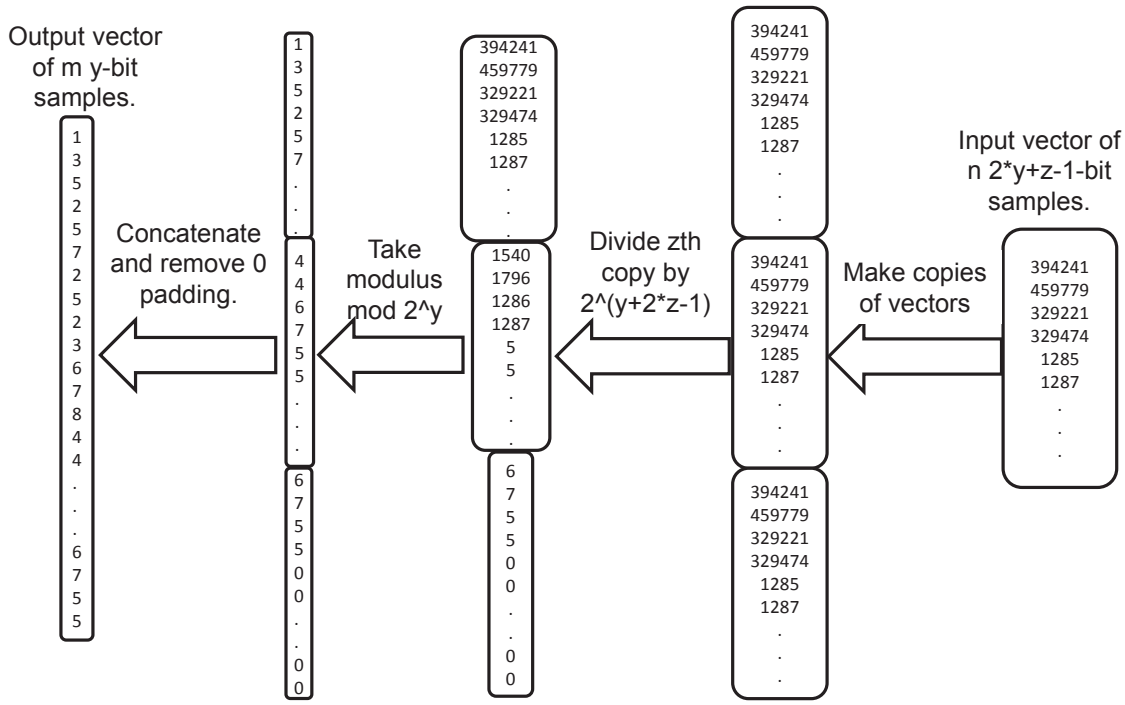


Fig. 5: Encrypted VoIP Decoding

- The VoIP plaintext can be securely encrypted into VoIP ciphertext.
- The summation of multiple VoIP ciphertexts can be successfully decrypted back into VoIP plaintext.
- The output VoIP plaintext can be decoded into an undistorted VoIP signal.
- All these operations need to be run efficiently on commodity hardware, such as 64- and 32-bit ARM and x86 processors.

Generally, these concerns mean that:

- The bitwidth of the VoIP data P needs to be sufficiently large so that given a VoIP integer signal vectors from y speakers $v_1, v_2, \ldots, v_y$, we are guaranteed that $v_1 + v_2 + \cdots + v_y = (v_1 + v_2 + \cdots + v_y mod P)$.
- The number of layers in the encodings (and hence the ring dimension n) and the plaintext modulus $p = 2^x$ need to be sufficiently large with respect to P so that for the encodings $z_1, z_2, \ldots, z_y$ where $z_i = encode(v_i)$, $z_1 + z_2 + \cdots + z_y = (z_1 + z_2 + \cdots + z_y modp)$.
- The ciphertext modulus needs to be sufficiently small that we can support computations on the ciphertext efficiently. For modern smart phones this means that the ciphertext modulus is at most 2^64, so we can use native 64-bit computations.
- The selection of parameters needs to provide a non-trivial root Hermite factor to provide security guarantees.

The selection of the ring dimension n and ciphertext modulus q parameters depends heavily on the desired security level and the plaintext modulus p. The plaintext modulus p depends on the VoIP data modulus P, the number of VoIP streams that need to be mixed without distortion y and the VoIP data bitwidth P. We capture the primary concerns influencing the selection of a ring dimension n and the modulus q at a high level as follows:

We choose to add discrete Gaussian noise to the fresh ciphertexts where $r = 3$ represents the selected probability distribution parameter as suggested in [8]. We have found theoretically that the smallest modulus q needs to satisfy the expression

$$q > 4pr\sqrt{n}w \tag{1}$$

in order to ensure successful decryption, where the parameter $w \approx 4$ represents an "assurance" measure for correct decryption (essentially, the probability of decryption failure is bounded by the probability that a normally distributed variable is more than $w\sqrt{2\pi}$ standard deviations from its mean), and $p \cdot r$ is the Gaussian parameter of the noise used in fresh ciphertexts.

(Hence r is the Gaussian parameter of the underlying NTRU-like problem.)

The most recent experimental evidence [13] suggests that $\delta = 1.007$ would require roughly 2^{40} core-years on recent Intel Xeon processors to break. Using the estimates from [14], [15], we found that in order to achieve a security level δ for a depth of computation $d = t - 1$ using the modulus q, we need to ensure that

$$n \geq \lg(q)/(4\lg(\delta)). \quad (2)$$

VI. Implementation

We evaluated our end-to-end encrypted VoIP capability by implementing our vocoder and homomorphic encryption library and then integrating them with an existing open-source VoIP teleconferencing capability. This activity resulted in an end-to-end encrypted VoIP client for teleconferencing clients running in an Apple iOS environment composed of a) the open-source Mumble VoIP client modified integrated with b) a custom linear codec of our design written in ANSI C and c) an FHE encryption library ported from Matlab to ANSI C. We also wrote and deployed the VoIP server capability running on Linux computing devices to perform the homomorphic mixing operation. We describe the implementation of this capability in this section.

A. Codec and Homomorphic Encryption Implementation

As with the cryptosystem design, our implementation used for an additive homomorphic encryption library is a customization of the design introduced in [8]. We implemented our scheme in the Mathworks Matlab environment and used the Matlab coder toolkit [16] to generate an ANSI C library of our implementation. We believe that additional performance improvements could be obtained by implementing our HE scheme natively in C.

We chose to implement our scheme in Matlab using the Matlab fixed-point toolbox because it provides an interpreted computation environment for rapid prototyping with native support for vector and matrix manipulation which simplifies implementation development. We found the Matlab syntax to be a natural fit for writing software to support the primitive lattice operations needed for our CRT-based NTRU-inspired homomorphic encryption design. The Matlab fixed-point toolbox also provides a path toward generated HDL implementations of our design that can be deployed for practical use on highly parallel computing hardware such as FPGAs. Part of our vision for the use of our SHE design is to develop an FPGA implementation of FHE [17], [18].

We implemented the vocoder capability in native ANSI C. We compiled this capability using the gcc tool to create a vocoder library which we then integrated with the homomorphic encryption library and a VoIP teleconferencing substrate.

B. VoIP Teleconferencing Substrate

Rather than construct a VoIP capability from whole clothe, we decided to construct an end-to-end encrypted VoIP teleconferencing capability by integrating our additive homomorphic encryption library and our vocoder library with an existing open-source VoIP teleconferencing library. We selected the Mumble VoIP library (http://mumble.sourceforge.net) for this integration because the Mumble is mature, offers high sound quality and runs on a variety of platforms.

We decided to implement our end-to-end encrypted VoIP teleconferencing capability for iOS clients because the native iOS development environment uses Objective C, a dialect of ANSI C. However, even though we only developed iOS clients, there is no reason our client library could not be integrated in other environments such as for Android, Windows, Mac or Blackberry clients.

By integrating with the Mumble library, our end-to-end encrypted VoIP library has the same use and deploy models as the standard Mumble capability. Notably, Mumble clients present the user a simple, easy to use, graphical user interface that can be easily understood with minimal training. An image of the modified client running on an iPod Touch can be seen in Figure 6 where the client is running in push-to-talk mode. This client is indistinguishable from the standard iOS Mumble client. The Mumble software can also be deployed through an app store model, or as binaries which can be loaded onto iOS devices through XCode.

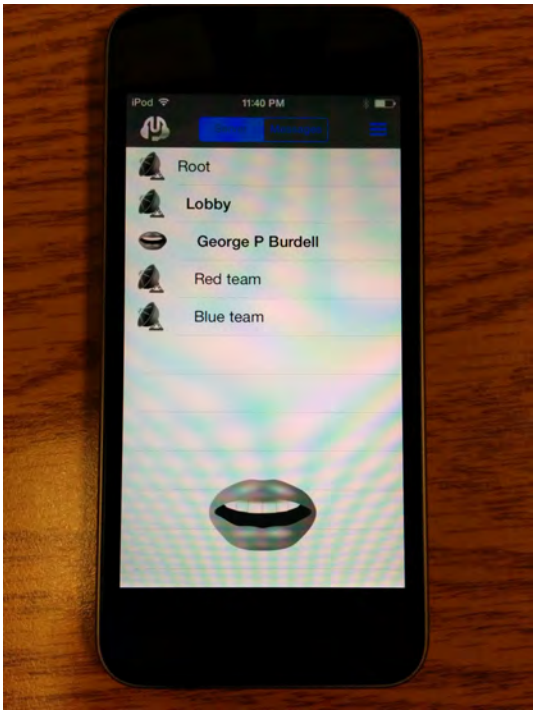


Fig. 6: The Push-To-Talk Client GUI

We integrated the iOS capability so that client handsets encrypt their audio streams using the clients public key. The proxy server computes over that encrypted data without decrypting the data or sharing keys. The output of the processing is sent to the client, where it is decrypted using the clients private key. No keys are stored on the teleconference server, so privacy is preserved even if an adversary views all communication links and operations on the server.

This integration was relatively straight forward with several notable exceptions to reduce packet drops and improve sound quality:

1) The client application generated voice packets that contained 480 samples at 48 KHz, or 10 ms worth of sound. The sound driver, however, generated slightly larger packets. As a result, the period of the sound packets was slightly larger than 10 ms, and every so often two sound packets were generated back to back. The original server set a 10 ms timer and just accepted one packet every 10 ms. We added a small queue at the server so we did not drop packets when we received two packets in a row very quickly.
2) We generated new frame numbers at the server as opposed to re-using the client frame numbers. The clients correlated the frame numbers with time. This cut down on the time jitter with regard to frame numbers.
3) The encryption and decryption operations for our applications were processor intensive, and were run in batches of several audio packets at once. We moved the encryption and decryption operations to a low priority thread and had the higher priority thread accept and queue new audio packets (both from the network, and from the microphone). This helped prevent a situation where we audio packets were dropped because we were too busy decrypting or encrypting.

The goal of our changes were to reduce the drop rate of packets which was an issue with initial prototypes. This, in turn, allowed us to increase audio sampling rate. As a result, we were sampling 10 bit samples at a rate of 48kHz. This conf guration provides a sound quality substantially better than PSTN as long as there are only a few packet drops.

After sampling the audio, we queue and encode 90ms blocks of this data into our encoder. We designed the system to accommodate 4 speakers, resulting in the homomorphic mixer to add 4 10-bit integers homomorphically, resulting in a 12-bit plaintext without the encoding layering. If we use a ring dimension $n = 1024$, we are required to use 2-layer encoding and have a resulting plaintext modulus of $p = 2^{24} = 16777216$. This encoding and encryption results in a root Hermite factor of $\delta = 1.006$ which is currently believed to be at least as secure as AES-128. With these parameter settings we observed that when running on an iPhone 5s, the encoding and encryption operation took a mean time of 9.2ms and decryption and decoding took 4.6ms. The summation on the VoIP server took 0.5ms. Transport of encrypted VoIP traff c from Cambridge MA to the Northern Virginia Amazon AWS servers took an average of 15ms. This results in a mean latency much less than our 100ms threshold for VoIP traff c, well within the bounds of reasonable, both in theory and in practice.

VII. Experimental Results

We experimentally evaluated the performance of the VoIP service by deploying our encrypted VoIP servers in each of the Amazon AWS data centers across the world. We then connected iPod Touch clients to each of the servers through various connection types in the metro area of a United States city in southern New England. These connections included 802.11n wireless enterprise gateway connected to a high-speed enterprise Internet connection, the 4G LTE, 3G and 2G connections over the T-mobile commercial wireless service and an AT&T DSL connection in a rural area outside the city.

We measured the upload and download throughput of the connections, the drop rate of VoIP packets routed through the various server locations and the subjective quality of the VoIP teleconference session as def ned by the experimenters. The upload and download throughput was measured by Ookla throughput measurement app [19] on the client devices. VoIP drop rates were measured experimentally by modifying the VoIP servers to measure drop rates. Voice quality was measured in comparison to PSTN voice quality where "Excellent" means the VoIP conversation was better than PSTN, "Good" means the VoIP conversation was comparable PSTN, "Poor" means the VoIP conversation was worse than PSTN but still usable for communication, and "Unusable" means the connection was useless for communication.

All of the experiments were run over a 2 hour period on a weekday evening using 2 iPod Touch clients with servers deployed on the Amazon AWS t1.micro instances [20]. Each of the clients were on independent connections to the Internet at all times, so there was low likelihood of one client contributing substantially to congestion for the other client.

Table I shows the upload and download throughput observed by each of the clients for each of the connections. Note that the rural DSL service provided better throughput than the 2G connection and better download throughput than the 3G connection.

TABLE I: Experimentally Measured Data Throughput in Mb/s for Connection Types

Connection Type	Upload Rate Mb/s	Download Rate Mb/s
Enterprise 802.11n	38.22	36.53
4G LTE	35.82	17
3G	6.31	0.43
2G	0.2	0.16
Rural DSL	2.55	0.47

Table II shows the packet drop rates observed at each of the servers at the various Amazon AWS locations for the various client connection types. Note that distance between the client and server had only a minor impact on drop rates, while the connection type had a very large impact on drop rates. This implies that the connection could be a bottleneck for the VoIP service.

Table III shows the subjective VoIP teleconference quality measurements observed through each of the servers at the various Amazon AWS locations for the various client connection types. Note that distance between the client and server had almost no observed impact on voice quality, while the connection type had a very large impact on voice quality.

We observed that all of the various connections supported acceptable VoIP teleconference capabilities except for the 2G connections. Over all of the acceptable connections, the lowest upload or download throughput observation was on the 3G download: 0.43Mb/s Because the VoIP download and upload data f ows are symmetric, this implies at least a 0.43Mb/s upload and download throughput connection is required to support VoIP teleconferencing using our prototype.

TABLE II: Packet Drop Rates For Various Server Locations and Client Internet Connection Types.

Server Location	Client Location	Enterprise 802.11n	4G LTE	3G	2G	Rural DSL
N. Virginia	S. New England	0%	10%	10%	66%	33%
Oregon	S. New England	0%	2%	3%	71%	35%
N. California	S. New England	0%	7%	8%	67%	34%
Ireland	S. New England	0%	7%	7%	73%	38%
Singapore	S. New England	5%	2%	2%	68%	39%
Tokyo	S. New England	1%	3%	4%	69%	37%
Sydney	S. New England	5%	3%	3%	67%	34%
Sao Paulo	S. New England	0.30%	4%	6%	76%	34%

TABLE III: Teleconference Quality For Various Server Locations and Client Internet Connection Types.

Server Location	Client Location	Enterprise 802.11n	4G LTE	3G	2G	Rural DSL
N. Virginia	S. New England	Excellent	Good	Good	Unusable	Poor
Oregon	S. New England	Excellent	Good	Good	Unusable	Poor
N. California	S. New England	Excellent	Good	Good	Unusable	Poor
Ireland	S. New England	Excellent	Good	Good	Unusable	Poor
Singapore	S. New England	Excellent	Good	Good	Unusable	Poor
Tokyo	S. New England	Excellent	Good	Good	Unusable	Poor
Sydney	S. New England	Excellent	Good	Good	Unusable	Poor
Sao Paulo	S. New England	Excellent	Good	Good	Unusable	Poor

In addition to our tests of connection-server pairings, we also tested the scalability of the number of clients that could be supported on a single server. For this experiment we connected 7 iPod Touch and/or iPhone 5s clients at various connections on the eastern United States seaboard to a single VoIP server in the Amazon AWS Northern Virginia data center. With these 7 connections running simultaneously with 4 people speaking simultaneously we were able to hold as good as a conversation possible with 4 people speaking simultaneously and no voice distortion was observed by the 3 non-speaking client users.

VIII. Related Work

Up to now, advances in secure VoIP technologies have focused on providing security for data in transit [1], [2] among other general security challenges such as DDoS attacks [21], identity and key management [22] among many others [23], [24]. These are all important challenges for secure VoIP teleconferencing capabilities, but a reliance on point-to-point encryption between participants has too often led to complicated VoIP teleconferencing systems and protocols [25]. In general, the complicated layering of protection mechanisms is often diff cult to execute in practice, leading to overly complicated systems which are diff cult to build and maintain. Further, these complicated systems are often diff cult to perform security audits on [26]–[28]. Although all of the partial security solutions have worked very well in isolation and have served their purposes as a rule, the at time complicated layering of these protocols has resulted in the introduction of possible security holes which has enabled data leakage.

To the best of our understanding, there have been no VoIP technologies which provides end-to-end encryption. Our solution seeks to provide a clean-slate data protection capability that is also compatible, or at least easily integrated with existing VoIP protocols and architectures. Because we provide end-to-end data encryption, our solution protects data against leakage even when layered with existing VoIP protocols for signaling and transport. Besides providing security against data leakage due to compromised servers, end-to-end encrypted VoIP teleconferencing has the possibility for greatly simplifying existing VoIP protocols, resulting in much simpler implementations and designs, thus resulting in more eff cient VoIP implementations that are easier to audit.

The basis of our design and implemented prototype for end-to-end encrypted VoIP teleconferencing is driven by and builds on recent recent breakthroughs in practical Fully Homomorphic Encryption (FHE). Recent breakthroughs in Homomorphic Encryption have shown that it is theoretically possible to securely run arbitrary computations over encrypted data without decrypting the data [29], [30]. There has been recent work on designing and implementing variations of homomorphic encryption schemes [10], [31]–[39]. These implementations have become increasingly practical with published results on both the runtime of isolated secure computing operations for some implementation [34], [37], [38] and evaluations of composite functions like AES [33], [36], [39].

Current approaches to design FHE schemes rely on a special, highly complex and computationally diff cult operation called bootstrapping [40] to support the encrypted execution of arbitrary functions. As such, we use a simplif cation of the general FHE designs called ”leveled” homomorphic encryption or Somewhat Homomorphic Encryption (SHE) the supports limited-depth computations, such as vector addition, which is much more eff cient because it does not require the use of bootstrapping.

Besides the runtime challenges of HE designs, there are serious applications issues associated with data structures and representations [39]. Furthermore, it has not been well explored how to convert existing data structures and algorithms into forms that can be eff ciently executed using FHE technologies. This is because FHE provides a very different computation model from existing RAM computing devices and the porting of known data structures and algorithms (such as for VoIP mixing) is non-trivial, especially for highly eff cient encrypted execution of these algorithms over the encrypted input data. As an example of limitations, early uses of FHE relied on encrypting individual bits in ciphertext. These limitations, in addition to the inherent computational cost of secure

computing using known FHE schemes, has until now prevented the practical use of FHE. Our innovation comes from designing a set of data structures, data encoding method (which we refer to as a vocoder) and a homomorphic mixing operation which supports a practical implementation of end-to-end encrypted VoIP teleconferencing.

In particular, a key innovation of ours is to go beyond simple bit-per-ciphertext encodings by placing entire VoIP data frames into each ciphertext. These codec designs are in some sense much simpler than existing modern codecs, such as the mu-law encoders [41] which are much more common in modern VoIP systems. There have been prior known approaches to Additive Homomorphic Encryption, such as Paillier encryption [42], but these approaches have not been practically employed to support encrypted VoIP mixing. Further, there has been no prior work that has investigated the data structures required to support end-to-end encrypted VoIP teleconferencing with homomorphic mixing.

There have been few other approaches to providing secure VoIP teleconferencing that approach to providing security properties such as end-to-end encryption. Most relevant is the work in [43] which discusses a VoIP teleconferencing approach based on Secure Multi-Party Computation (SMC) [44]. This prior SMC-based approach is also built by modifying the Mumble/Murmur software and our team received implementation advice from the authors of [43]. Unlike our HE-based approach which requires only one untrusted server for end-to-end encrypted VoIP teleconferencing, the MPC-based approach in [43] requires that every participant in the teleconference have at least one trusted server.

IX. Discussion and Ongoing Work

Given our initial design goals, our implementation and our experimentation, our assessment is that we met our initial design criteria. More specif cally with respect to the previously discussed design goals:

1) **Sound Quality**: The end-to-end encrypted VoIP teleconferencing capability provide sound quality at least as good as a Public Switched Telephone Network (PSTN) with full-duplex capability.
2) **Latency**: The end-to-end encrypted VoIP teleconferencing capability provides an end-to-end latency of less than 90ms for trans-continental conversations, better than our design goal.
3) **Scalability**: The end-to-end encrypted VoIP teleconferencing capability should be able to supports four people speaking simultaneously and we have experimentally verif ed that participants can listen to the audio stream without a performance impact.
4) **Secure**: The end-to-end encrypted VoIP teleconferencing capability provides a remote Hermite factor of 1.006 which provides an encryption work factor at least as good as the work factor for AES-128 to the best current understanding of both of these cryptosystems.
5) **Resource Efficient**: The ciphertext expansion of our encrypted VoIP data is roughly at a factor of 5. This is highly eff cient as compared to many other encryption schemes, especially current homomorphic encryption schemes.
6) **Wide Geographic Area**: We have tested the end-to-end encrypted VoIP teleconferencing capability with speakers in multiple eastern US states (Virginia and Massachusetts) and with the server running in the Amazon AWS cloud in Northern Virginia. We have also tested in a similar scenario with the VoIP server in many Amazon AWS locations and clients on the eastern seaboard of the United States for various connection types. In all of these situations performance was not compromised due to geography and the main determinant of quality was connection throughput.
7) **Portable**: The end-to-end encrypted VoIP teleconferencing capability is easily ported to other client and server types. There is no reason our ANSI C libraries could not be used to support integration with other VoIP infrastructure, or even other kinds of clients such as Android clients.
8) **Easily Deployable**: The end-to-end encrypted VoIP teleconferencing capability is easy to deploy, at least as easy as an iOS app.
9) **Usable**: The end-to-end encrypted VoIP teleconferencing capability is intuitive and easy to use as it build on the prior usability of the Mumble GUI.
10) **Extensible**: The end-to-end encrypted VoIP teleconferencing capability is relatively easy to modify to add additional and more advanced functionality at a later date, inclusive of QoS management, encrypted text message passing, amongst other capabilities.

Beyond our VoIP functionality, our HE implementation is part of a long-term community vision to support a general, practical and secure computing capability through a layered services architecture. Part of our vision is to provide software interfaces in our design for our highly optimized implementations of the basic FHE operations (KeyGen, Encrypt, EvalAdd, EvalMult, Decrypt) for both general and specif c applications.

Although we only utilize limited-depth Somewhat and Partially Homomorphic Encryption capabilities, our encryption system design is a scaled-down version of a Fully Homomorphic Encryption (FHE) scheme design. When used in conjunction of a variation of a not previously implemented bootstrapping scheme [45] simplif ed for power-of-2 rings, our design offers the possibility for a much more general VoIP teleconferencing capability that incorporates signal detection and noise f ltering operations on the encrypted VoIP channels. This more general design would enable protection against some of the more practical attacks that could be made by an adversary such as noise injection attacks where an adversary inserts noise into a VoIP teleconferencing session to reduce the ability of participants to hear one another. Using more general FHE capabilities, we could enable the untrusted cloud host to securely f lter the encrypted VoIP signals before or after mixing to reduce the impacts of insertion attacks.

A further aspect of our layered architecture vision is an ability to mix-and-match a computing substrate at the server for much larger scalability and throughput. Although not an immediate focus of the results reported here, our FHE design ports to other, high-performance and low-cost parallel computing environments such as FPGAs [18] and GPUs [46] operating as “FHE co-processors”. If ported to a dedicated

FPGA co-processor, the runtime of our underlying SHE/FHE implementation can be greatly improved upon as compared to the runtime of the corresponding interpreted CPU-only implementation which we discuss herein.

Taken together, we see our design and experimentation with our end-to-end encrypted VoIP teleconferencing capability as being a highly practical and extensible implementation that protects VoIP teleconferencing users against data leakage through a very simple but highly secure design. Our primary path forward is to add more functionality, protection against other kinds of attacks and increasingly leverage the inherent parallelism of our design at multiple levels of our implementation.

ACKNOWLEDGEMENT

The authors wish to acknowledge the helpful feedback and guidance of Prof. Chris Peikert in preparing the material discussed in this paper.

The authors wish to acknowledge implementation advice received from Dave Archer and Thomas DuBuisson.

REFERENCES

[1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (srtp)," 2004.

[2] P. Zimmermann, A. Johnston, and J. Callas, "Zrtp: Media path key agreement for secure rtp," *draft-zimmermann-avt-zrtp-04 (work in progress)*, 2007.

[3] R. Zhang, X. Wang, R. Farley, X. Yang, and X. Jiang, "On the feasibility of launching the man-in-the-middle attacks on voip from remote attackers," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 61–69. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533069

[4] A. Ornaghi and M. Valleri, "Man in the middle attacks," in *Blackhat Conference Europe*, 2003.

[5] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. N. Gansterer, "Security and usability aspects of man-in-the-middle attacks on zrtp." *J. UCS*, vol. 14, no. 5, pp. 673–692, 2008.

[6] S. Capkun, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 1, pp. 52–64, 2003.

[7] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Rfc 3261: Sip: session initiation protocol," 2003.

[8] K. Rohloff and D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on NTRU," in *Proceedings of the 2nd Workshop on Applied Homomorphic Cryptography (WAHC)*, 2014.

[9] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, J. P. Buhler, Ed. Springer Berlin Heidelberg, 1998, vol. 1423, pp. 267–288. [Online]. Available: http://dx.doi.org/10.1007/BFb0054868

[10] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-f y multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC*, 2012, pp. 1219–1234.

[11] C. Gentry, S. Halevi, V. Lyubashevsky, C. Peikert, J. Silverman, and N. Smart, 2011, personal communication.

[12] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology - EUROCRYPT 2010*, ser. Lecture Notes in Computer Science, H. Gilbert, Ed. Springer Berlin / Heidelberg, 2010, vol. 6110, pp. 1–23, 10.1007/978-3-642-13190-5. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13190-5

[13] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 7073. Springer, 2011, pp. 1–20.

[14] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *CT-RSA*, 2011, pp. 319–339.

[15] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post Quantum Cryptography*. Springer, February 2009, pp. 147–191.

[16] MATLAB, *R2012b*. Natick, Massachusetts: The MathWorks Inc., 2012.

[17] D. B. Cousins, K. Rohloff, C. Peikert, and R. Schantz, "SIPHER: Scalable implementation of primitives for homomorphic encryption - FPGA implementation using Simulink," in *Fifteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, ser. HPEC '11, 2011.

[18] ——, "An update on scalable implementation of primitives for homomorphic encryption - FPGA implementation using simulink," in *Sixteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, ser. HPEC '12, 2012.

[19] *Ookla*, 2014. [Online]. Available: https://www.ookla.com/

[20] *Amazon AWS Instance Types*, 2014. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[21] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[22] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in CryptologyCRYPTO93*. Springer, 1994, pp. 232–249.

[23] A. D. Keromytis, "A survey of voice over ip security research," in *Information Systems Security*. Springer, 2009, pp. 1–17.

[24] ——, "Voice-over-ip security: Research and practice," *Security & Privacy, IEEE*, vol. 8, no. 2, pp. 76–78, 2010.

[25] V. Kumar, M. Korpi, and S. Sengodan, *IP Telephony with H.323: Architectures for Unified Networks and Integrated Services*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[26] Z. Anwar, W. Yurcik, R. E. Johnson, M. Haf z, and R. H. Campbell, "Multiple design patterns for voice over ip (voip) security," in *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*. IEEE, 2006, pp. 8–pp.

[27] J. F. Ransome, C. CISM, J. Rittinghouse *et al.*, *Voice over Internet Protocol (VoIP) Security*. Digital Press, 2005.

[28] D. C. Sicker and T. Lookabaugh, "Voip security: Not an afterthought," *Queue*, vol. 2, no. 6, pp. 56–64, Sep. 2004. [Online]. Available: http://doi.acm.org/10.1145/1028893.1028898

[29] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2009, aAI3382729.

[30] ——, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[31] J. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Stam, Ed. Springer Berlin Heidelberg, 2013, vol. 8308, pp. 45–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45239-0_4

[32] J. Cheon, J.-S. Coron, J. Kim, M. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully homomorphic encryption over the integers," in *Advances in Cryptology EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, T. Johansson and P. Nguyen, Eds. Springer Berlin Heidelberg, 2013, vol. 7881, pp. 315–335. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38348-9_20

[33] Y. Doroz, Y. Hu, and B. Sunar, "Homomorphic aes evaluation using ntru," Cryptology ePrint Archive, Report 2014/039, 2014, http://eprint.iacr.org/.

[34] C. Gentry and S. Halevi, "Implementing Gentry's fully homomorphic encryption scheme," in *Advances in Cryptology EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. Paterson, Ed. Springer Berlin / Heidelberg, 2011, vol. 6632, pp. 129–148.

[35] ——, "HElib," https://github.com/shaih/HElib, 2014.

[36] C. Gentry, S. Halevi, and N. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology CRYPTO 2012*, ser. Lecture

Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds. Springer Berlin / Heidelberg, 2012, vol. 7417, pp. 850–867.

[37] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, ser. CCSW '11. New York, NY, USA: ACM, 2011, pp. 113–124. [Online]. Available: http://doi.acm.org/10.1145/2046660.2046682

[38] H. Perl, M. Brenner, and M. Smith, "Poster: an implementation of the fully homomorphic Smart-Vercauteren cryptosystem," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 837–840. [Online]. Available: http://doi.acm.org/10.1145/2093476.2093506

[39] D. Wu and J. Haven, "Using homomorphic encryption for large scale statistical analysis," 2012.

[40] J. Alperin-Sheriff and C. Peikert, "Practical bootstrapping in quasilinear time," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 1–20.

[41] L. Liu, M. Fukumoto, and S. Saiki, "An improved mu-law proportionate nlms algorithm," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3797–3800.

[42] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*. Springer, 1999, pp. 223–238.

[43] J. Launchbury, D. Archer, T. DuBuisson, and E. Mertens, "Application-scale secure multiparty computation," in *Programming Languages and Systems*, ser. Lecture Notes in Computer Science, Z. Shao, Ed. Springer Berlin Heidelberg, 2014, vol. 8410, pp. 8–26. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54833-8_2

[44] S. Goldwasser, "Multi party computations: past and present," in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*. ACM, 1997, pp. 1–6.

[45] J. Alperin-Sheriff and C. Peikert, "Practical bootstrapping in quasilinear time," in *Advances in Cryptology CRYPTO 2013*, ser. Lecture Notes in Computer Science, R. Canetti and J. Garay, Eds. Springer Berlin Heidelberg, 2013, vol. 8042, pp. 1–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40041-4_1

[46] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Accelerating fully homomorphic encryption on GPUs," in *Proceedings of the IEEE High Performance Extreme Computing Conference*, 2012.

A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU*

Kurt Rohloff David Bruce Cousins
Raytheon BBN Technologies
10 Moulton St.
Cambridge, MA, USA
{krohloff,dcousins}@bbn.com

February 16, 2014

Abstract

In this paper we report on our work to design, implement and evaluate a Fully Homomorphic Encryption (FHE) scheme. Our FHE scheme is an NTRU-like cryptosystem, with additional support for efficient key switching and modulus reduction operations to reduce the frequency of bootstrapping operations. Ciphertexts in our scheme are represented as matrices of 64-bit integers. The basis of our design is a layered software services stack to provide high-level FHE operations supported by lower-level lattice-based primitive implementations running on a computing substrate. We implement and evaluate our FHE scheme to run on a commodity CPU-based computing environment. We implemented our FHE scheme to run in a compiled C environment and use parallelism to take advantage of multi-core processors. We provide experimental results which show that our FHE implementation provides at least an order of magnitude improvement in runtime as compared to recent publicly known evaluation results of other FHE software implementations.

1 Introduction

Recent breakthroughs in Homomorphic Encryption have shown that it is theoretically possible to securely run arbitrary computations over encrypted data without decrypting the data [10, 11]. There has been recent work on designing and implementing variations of Somewhat Homomorphic Encryption (SHE) and Fully Homomorphic Encryption (FHE) schemes [2, 6, 9, 12, 13, 15, 18, 23, 24, 28]. These implementations have become increasingly practical with published results on both the runtime of isolated EvalAdd and EvalMult operations for some implementation [12, 23, 24] and evaluations of composite functions like AES [9, 15, 28].

Current approaches to design FHE schemes rely on bootstrapping to arbitrarily increase the size of computation supported by an underlying SHE scheme. Many current implementations of SHE and FHE schemes rely on the the manipulation of very large integers so that the schemes are both secure and capable of supporting the evaluation of sufficiently large circuits. Prior SHE and FHE implementation designs [12, 15, 23, 24], for the most part, rely on single-threaded execution on commodity CPU-type hardware, partially due to the difficulty of or lack of native support for multi-threaded execution with underlying software libraries [20, 25]. This, in addition to the inherent computational cost of secure computing using known SHE and FHE schemes, prevented the practical use of SHE and FHE.

*Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited.)

In this paper we report on our work to design, implement and evaluate a scalable Fully Homomorphic Encryption (FHE) scheme which addresses the limitations for secure arbitrary computation. Our implementation uses a variation of a not previously implemented bootstrapping scheme [1] simplified for power-of-2 rings. We also use a "double-CRT" representation of ciphertexts which was also discussed in [15]. With this double-CRT representation, we can select parameters so that ciphertexts are secure when represented as matrices of 64-bit integers, but still support the secure execution of programs on commodity computing device without expending unnecessary computational overhead manipulating large multi-hundred-bit or even multi-thousand-bit integers.

We implement in software specialized lattice primitives such as Ring Addition, Ring Multiplication and the Chinese Remainder Transform (CRT). We use our primitive implementations to construct the FHE operations of Key Generation (KeyGen), Encryption (Enc), Decryption (Dec), Evaluation Addition (EvalAdd), Evaluation Multiplication (EvalMult) and Bootstrapping (Boot). We use supporting Modulus Reduction (ModReduce), Ring Reduction (RingReduce) and Key Switching (KeySwitch) operations to augment the EvalMult operation and support larger depth computations without bootstrapping or decreasing the security of our scheme.

We implemented this scheme to run in a compiled C environment and use parallelism to take advantage of multi-core processors. Taken together, our implementation of these concepts points the way to a practical implementation of FHE with a more efficient (and less frequent) use of the bootstrapping operation. We evaluate the performance of our software library as a set of compiled executables in a commodity CPU-based multi-core Linux environment. The evaluated performance of our library compares favorably with evaluations of the reported experimental CPU-based evaluation results of other recent SHE and FHE schemes implemented in software such as in [12, 23, 24].

This paper is organized as follows. In Section 2 we discuss how we represent ciphertexts in our implementation. In Section 3 we define our NTRU-based FHE scheme. In Section 4 we discuss parameter selection for our NTRU-based scheme to provide practical secure computing on commodity computing hardware. In Section 5 we discuss our experimental results from our FHE scheme implemented in Matlab. We conclude the paper with a discussion of our insights and next steps in Section 6. Data tables experimental runtime results can be seen in Appendix A.

2 Double-CRT Ciphertext Representation

Previous SHE/FHE designs and implementations use two primary parameters to tune the security provided and the supported depth of homomorphic computation (without resorting to bootstrapping): the ring dimension n and the ciphertext modulus q. With these parameters, fresh ciphertexts are typically represented as n-element integer arrays, where each array element consists of at least $\log_2(q)$ bits. In previous implementations the ring dimension n typically ranged from 512 (2^9) to 16384 (2^{14}) and beyond, while several hundred to several thousand bits was typically required to represent q. In the previous implementations that use this "large-q" approach, the practicality challenge derives from the difficulty of supporting both a large ring dimension n (which provides comparatively better security) and a large q (which increases the depth of computation supported).

The requirement of a very large q is potentially problematic, because the number of clock cycles to support mod-q operations using naive "big integer" arithmetic grows at least linearly (and often quadratically) with the number of bits used to represent q for even the simplest operations, e.g., modular addition and multiplication. We use a variation of the double-CRT approach discussed in [15] to circumvent this problem using the standard technique of a "residue number system" (based on the Chinese remainder theorem over the integers) to represent ciphertexts as t length-n integer vectors of mod-q_i values instead of a single integer vector mod q where $q = q_1 * \cdots * q_t$ for pairwise coprime moduli q_i. For our ciphertext representation we use t length-n integer vectors of mod-q_i values represented as a $n \times t$ integer matrix. With our double-CRT approach, the number of moduli (t) grows to support the secure execution of larger programs, but more bits are not required to represent the moduli $q_1, \cdots, q_t$. Our implementation supports the secure execution of depth $t-1$ programs with t moduli.

The double-CRT representation is an extension of the Chinese Remainder Transform (CRT) [19]

representation used in prior SHE and FHE implementations. Chinese remainder transforms are used to convert ciphertexts from the natural "power basis" representation to the double-CRT representation. This conversion can mathematically be represented as a multiplication by square $n \times n$ matrices, but admits a fast, highly parallel evaluation procedure that is closely related to the Cooley-Tukey Fast Fourier Transform (and others.)

As we discuss more in Section 4 below, each of the moduli $q_1, \cdots, q_t$ can be represented as 64-bit integers and still support the secure execution of non-trivial programs. These 64-bit representations greatly improve the practicality of our approach to SHE and FHE. By using 64-bit modular operations to manipulate ciphertexts, keys, etc., we support faster low-level execution of the SHE operations on commodity 64-bit (or even 32-bit) processors.

An advantage of our double-CRT NTRU approach is that the FHE operations can be highly parallelized. Similar to the standard CRT representation, by using a double-CRT representation, the EvalAdd, EvalMult operations and key sub-operations in Bootstrapping, Modulus Reduction, Ring Switching and Key Switching can become t naively parallelized operations. This greatly simplifies the secure execution of programs using our FHE implementation as compared to other, non-CRT representations of ciphertexts.

3 Cryptosystem

In this section we describe the somewhat homomorphic cryptosystem we use that is very similar to the NTRU system [16], though it was not until recently that its homomorphic properties were noticed independently by López-Alt et al. [18] and Gentry et al. [14].

For ease of implementation and design simplicity, we limit our description to power-of-2 cyclotomic rings. For ring dimension n which is a power of 2, define the ring $R = \mathbb{Z}[x]/(x^n + 1)$ (i.e., integer polynomials modulo $x^n + 1$). For a positive integer q, define the quotient ring $R_q = R/qR$ (i.e., integer polynomials modulo $x^n + 1$, with coefficients from $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$).

3.1 Basic NTRU-Type System

In this subsection we provide a mathematical description of a somewhat homomorphic NTRU-based scheme. The message space is R_p for some integer $p \geq 2$, and most arithmetic operations are performed modulo some $q \gg p$ that is relatively prime with p. Fast addition and multiplication in R_q can be performed by using the mod-q Chinese Remainder Transform (CRT) representation of elements. The basic operations of the scheme are as follows:

- Gen: choose a short $f \in R$ such that $f = 1 \bmod p$ and f is invertible modulo q, and a short $g \in R$. Output $pk = h = g \cdot f^{-1} \bmod q$ and $sk = f$.

  Note that f is invertible modulo q if and only if each of its mod-q CRT coefficients is nonzero. The CRT coefficients of f^{-1} (modulo q) are just the mod-q inverses of those of f.

  Concretely, the short elements f and g can be chosen from discrete Gaussians. E.g., we can let $f = p \cdot f' + 1$ for some Gaussian-distributed f'. Note that such an f will have expectation (center) 1. Using a zero-centered f can have some advantages, and may be chosen using a more sophisticated sampling algorithm.

- Enc($pk = h, \mu \in R_p$): choose a short $r \in R$ and a short $m \in R$ such that $m = \mu \bmod p$. Output $c = p \cdot r \cdot h + m \bmod q$.

  Concretely, m can naively be chosen as $m = p \cdot m' + \mu$ for a Gaussian-distributed m', but again, such an m is not zero-centered. It is typically better to choose m as a zero-centered random variable congruent to μ modulo p.

- Dec($sk = f, c \in R_q$): compute $\bar{b} = f \cdot c \bmod q$, and lift it to the integer polynomial $b \in R$ with coefficients in $[-q/2, q/2)$. Output $\mu = b \bmod p$.

The homomorphic operations are defined as follows:

- $\mathsf{EvalAdd}(c_0, c_1)$: output $c = c_0 + c_1 \bmod q$.
- $\mathsf{EvalMult}(c_0, c_1)$: output $c = c_0 \cdot c_1 \bmod q$.

With the use of EvalMult, the decryption procedure needs to be modified. Define the "degree" of ciphertexts as follows: a freshly generated ciphertext has degree 1, and the degree of $c = \mathsf{EvalMult}(c_0, c_1)$ is the sum of the degrees of c_0 and c_1. Then decryption of a ciphertext c of degree at most d is the same as above, except that we instead compute $\bar{b} = f^d \cdot c \bmod q$.

3.2 Key Switching

Key switching converts a ciphertext of degree at most d, encrypted under a secret key f_1, into a degree-1 ciphertext c_2 encrypted under a secret key f_2 (which may or may not be the same as f_1). This requires publishing a "hint"

$$a_{1\to 2} = m \cdot f_1^d \cdot f_2^{-1} \bmod q,$$

for a short $m \in R$ congruent to 1 modulo p. (Concretely, we can choose $m = p \cdot e + 1$ for a Gaussian-distributed e, though a zero-centered m is better.)

- $\mathsf{KeySwitch}(c_1, a_{1\to 2})$: output $c_2 = a_{1\to 2} \cdot c_1 \bmod q$.

Note that $a_{1\to 2}$, c_1, c_2 can all be stored and operated upon in CRT form, so key switching is very efficient: the hint is just one ring element, and the procedure involves just one coordinate-wise multiplication of the CRT vectors. This compares quite favorably to key-switching procedures for other cryptosystems, which typically require decomposing a ciphertext into several short ring elements and performing several ring multiplications.

3.3 Ring Reduction

Ring reduction maps a ciphertext from ring n to smaller ring $n' = n/2^a$, where typically $a = 1$. Although we describe a ring reduction operation for power-of-2 rings, more general ring switching approaches exist and can be obtained from simple generalizations of the approach we describe here.

The basic ring switching operation is a Decompose algorithm, which maps a dimension n ring to dimension n' elements. Decompose(c) works as follows:

- Let $c = (c_0, ..., c_{n-1})$ be in the power basis and let $w = n/n'$.
- We output ciphertexts c'_i for each $i = 0, ..., w-1$ where $c'_i = (c_i, c_{w+i}, c_{2w+i}, ..., c_{(m'-1)w+i})$. I.e., c'_i just consists of those entries of c whose indices are $i \bmod w$.

Before applying Decompose we first key-switch the ciphertext to one which can be decrypted by a "sparse" secret key sk, whose only nonzero entries in the power basis are at indices equal to $0 \bmod w$. We perform the ring-switching on a ciphertext c, by performing key-switching on c to get cp (encrypted under sk), then call Decompose(cp) to get the $/c'_i/$. The ciphertext c should only have plaintext data only in its indices $0 \bmod w$. Otherwise, this data is lost during the ring reduction operation.

3.4 Modulus Reduction

Modulus reduction, initially proposed in [3], converts a ciphertext from modulus q to a smaller modulus (q/q'), where q' divides q (and so is also relatively prime with p), while also reducing the underlying noise by about a q' factor.

The basic description is as follows: given a ciphertext $c \in R_q$, we add to it a small integer multiple of p that is congruent to $-c \bmod q'$. This ensures that the underlying noise remains small, the plaintext remains unchanged, and the resulting ciphertext is divisible by q'. Then we can divide both the ciphertext and modulus by q', which reduces the underlying noise term by a q' factor as well.

Note that the final step (of dividing by q') implicitly multiplies the underlying message by $(q')^{-1} \bmod p$. We can either keep track of these extra factors as part of the ciphertext and correct for them as the final step of decryption, or we can just ensure that $q' = 1 \bmod p$, so that division by q' does not affect the underlying message.

The following formal procedure uses the fixed (ciphertext-independent) value $v = (q')^{-1} \bmod p$, which can be computed in advance and stored.

- ModReduce(c, q, q'):
  1. compute a short $d \in R$ such that $d = c \bmod q'$.
  2. compute a short $\Delta \in R$ such that $\Delta = (vq' - 1) \cdot d \bmod (pq')$. E.g., all of Δ's integer coefficients can be in the range $[-pq'/2, pq'/2)$.
  3. let $d' = c + \Delta \bmod q$. By construction, d' is divisible by q'.
  4. output $(d'/q') \in R_{(q/q')}$.

Following [15], the above is most efficient to implement when $q = q_1 \cdots q_t$ is the product of several small, pairwise relatively prime moduli; when q' is one of those moduli (say, $q' = q_t$ without loss of generality); and when c is represented in "double-CRT" form, i.e., each of c's mod-q CRT coefficients is itself represented in (integer) CRT form as a vector of mod-q_i values, one for each i. Then the above steps can be computed as follows:

1. Computing d is done by inverting the mod-q_t CRT on the vector of mod-q_t components of c (leaving the other mod-q_i components unused), and interpreting the resulting coefficients as integers in $[-q_t/2, q_t/2)$.
2. Computing Δ is done by multiplying the coefficients of d by the fixed scalar $(vq_t - 1)$ modulo pq_t.
3. Adding Δ to c is done by computing the double-CRT representation of Δ (i.e., applying each mod-q_i CRT to Δ), and adding it entry-wise to c's double-CRT representation.

   Note that the mod-q_t CRTs of Δ and c are just the negations of each other (by construction), so their sum is the all-zeros vector. Therefore, there is no need to explicitly compute the mod-q_t CRT of Δ.
4. Computing d'/q_t is done by dropping the mod-q_t components in the double-CRT representation of d' (which are all zero anyway), and multiplying every mod-q_i component by the fixed scalar $q_t^{-1} \bmod q_i$. (These scalars can be computed in advance and stored.)

3.5 Composed EvalMult

We use the Key Switching, Ring Reduction and Modulus Reduction operations as supporting functions with EvalMult to improve noise management and enable more computation between calls to the Bootstrapping operation. Taken together, we form a composite operation, which we call ComposedEvalMult, from the sequential execution of an EvalMult, Key Switching and Modulus Reduction operation.

Ring Reduction is called during some ComposedEvalMult operations, depending on the level of security provided by a ciphertext resulting from the result of the Ring Reduction operation. As Modulus Reduction operations are performed the security provided by a ciphertexts increases (as described in 4.) Ring Reduction correspondingly reduces the level of security provided by a ciphertext. We implemented our FHE library such that a minimum level of security δ' is provided at all times, and this level of δ' is a parameter selectable by the library user. If a call to a Ring Reduction operation will result in a level of security $\delta \leq \delta'$, then the RingReduction is performed in the ComposedEvalMult operation.

Our conception is that due to the ModReduction and RingReduction component of ComposedEvalMult, it is feasible to coordinate the choice of the original ciphertext width t and the scheduling of ComposedEvalMult operations so that the final ciphertext resulting from secure circuit evaluation and which needs to be decrypted is only one column wide with respect to a single modulus q_1 and provides

a level of security at least as great as the original ciphertexts resulting from the encryption operation. More explicitly, if we need to support a depth $t-1$ computation, the initial encryptions should only be t columns wide to ensure that the final ciphertext is 1 column wide. Whereas the runtime of Encryption, EvalAdd, ComposedEvalMul depend on the ring dimension and depth of computation supported, the Decryption operation would hence depend only on the final ring dimension after all ring switching has been completed. If we need to decrypt a ciphertext that has multiple columns we our double-CRT representation, we could perform multiple ModReduction operations to reduce this $t > 1$ ciphertext until we are left with a single mod-q_1 column.

3.6 Bootstrapping

The basis of our bootstrapping approach comes from a new approach to homomorphic rounding. This approach to bootstrapping is described in detail in [1]. We provide a high-level overview of this operation here, simplified for our restriction to power-of-2 rings. This operation has the following steps:

1. *Round the ciphertext*: For each entry v for residue i, we output $round(v * q/q_i)$, where the inner expression is rational, and "round" means taking the nearest integer. Generally $q = 2^\ell$ is chosen experimentally, but as small as possible.
2. *Convert the plaintext modulus*: This is no-op under our simplifying assumptions.
3. *Lift the ciphertext and plaintext moduli*: This is also a no-op under our simplifying assumptions.
4. *Scale the ciphertext*: We scale up the ciphertext by a Q/q' factor (rounding to nearest integers in the power basis), and embed into dimension N (new ring dimension) as well. The plaintext modulus is still q'.
5. *Compute the homomorphic trace*: The following steps are performed iteratively $\log_2(N)$ times:
   (a) "Lift" the ciphertext modulus to $2Q$, which has the effect of making the plaintext modulus 2q.
   (b) Apply the automorphism from [1], with appropriate key switching to put the result into the same key as the original ciphertext in the iteration.
   (c) Sum the original and resulting ciphertexts.
   (d) Divide the ciphertexts by 2.
6. *Perform a homomorphic rounding*: This operation is described in Appendix B of [1].

4 Parameter Selection

The selection of n and $q_1, \ldots, q_t$ depends heavily on the plaintext modulus p, the depth of computation that needs to be supported, and the desired security level. We capture the primary concerns influencing the selection of a ring dimension n and the moduli $q_1, \ldots, q_t$ at a high level as follows:

- The necessary ring arithmetic should be easily supported on the computation substrate – i.e., that mod-q_i operations (for $i \in \{1, \ldots, t\}$) require few clock cycles.
- The moduli $q_1, \ldots, q_t$ are sufficiently large to enable sufficient noise shrinkage via modulus reduction.
- The ring dimension n and noise parameters are sufficiently large so the scheme provides adequate security.
- The ring dimension n is not so large that it becomes overly time-consuming and memory-intensive to manipulate the ciphertexts.
- The plaintext modulus p and any noise added to the ciphertext during encryption is sufficiently small that we can evaluate reasonably sized circuits with correct decryption.

Table 1: Dependence of bit lengths of moduli q_i, as a function of ring dimension for $p = 2$.

Ring dimension n	512	1024	2048	4096	8192	16384
Bit length $\log_2(q_i)$	44	45	47	48	50	51

We choose to add discrete Gaussian noise to the fresh ciphertexts where $r = 6$ represents the selected probability distribution parameter. We have found theoretically that the smallest modulus q_1 needs to satisfy the expression

$$q_1 > 4pr\sqrt{n}w \tag{1}$$

in order to ensure successful decryption, where the parameter $w \approx 6$ represents an "assurance" measure for correct decryption (essentially, the probability of decryption failure is bounded by the probability that a normally distributed variable is more than $w\sqrt{2\pi}$ standard deviations from its mean), and $p \cdot r$ is the Gaussian parameter of the noise used in fresh ciphertexts. (Hence r is the Gaussian parameter of the underlying NTRU-like problem.)

After selecting q_1, we select the remaining $q_i \in \{q_2, \ldots, q_t\}$ such that

$$q_i > 4p^2r^5n^{1.5}w^5, \tag{2}$$

which ensures that modulus reduction by a factor of q_i sufficiently reduces the noise after a ComposedEvalMult operation. For implementation simplicity, we set q_1 to be the smallest feasible solution to $q_1 > 4p^2r^5n^{1.5}w^5$. Consequently all q_i are represented by $\log_2(q_t)$ bits, leading to simpler implementations.

Table 1 shows how many bits are required to represent $q_1, \ldots, q_t$ for varying ring dimensions for $p = 2$. Note that all $q_1, \ldots, q_t$ can be represented in less than 64 bits.

Following [5, 17, 22, 26], we use the standard "root Hermite factor" δ as the primary measure of concrete security for a set of parameters. The most recent experimental evidence [5] suggests that $\delta = 1.007$ would require roughly 2^{40} core-years on recent Intel Xeon processors to break. Using the estimates from [17, 22], we found that in order to achieve a security level δ for a depth of computation $d = t - 1$ using the t moduli $q_1, \ldots, q_t$, we need to ensure that

$$n \geq \lg(q_1 \cdots q_t)/(4\lg(\delta)). \tag{3}$$

Table 2 shows how δ varies as a function of the ring dimension and depth of computation supported. Based on our analysis, if we impose the requirement that $\delta \leq 1.007$, then we would need to use ring dimension $n = 16324$ to support depth $d = 13$ computations.

Table 2: Security level δ, as a function of depth of computation supported and ring dimension for $p = 2$.

Dim. \ Depth	1	3	5	7	9	11	13	15	17	19
512	1.015	1.045	1.077	1.109	1.143	1.178	1.213	1.250	1.288	1.327
1024	1.007	1.023	1.038	1.054	1.070	1.087	1.104	1.121	1.138	1.155
2048	1.004	1.012	1.020	1.028	1.036	1.044	1.053	1.061	1.069	1.078
4096	1.002	1.006	1.010	1.014	1.018	1.022	1.026	1.030	1.035	1.039
8192	1.0011	1.003	1.005	1.007	1.009	1.011	1.013	1.016	1.018	1.020
16384	1.0005	1.0016	1.003	1.003	1.005	1.006	1.007	1.008	1.009	1.010

5 Evaluation Experiments

We implemented our scheme in the Mathworks Matlab environment and used the Matlab coder toolkit [21] to generate an ANSI C representation of our implementation. We subsequently hand-modified our

auto-generated ANSI C to incorporate the pthreads library [4] to leverage parallelism. We compiled this ANSI C using gcc to run as an executable in a Linux environment. We believe that additional performance improvements could be obtained by implementing our FHE scheme natively in C.

We chose to implement our scheme in Matlab because it provides an interpreted computation environment for rapid prototyping with native support for vector and matrix manipulation which simplifies implementation development. We found the Matlab syntax to be a natural fit for writing software to support the primitive lattice operations needed for our double-CRT NTRU-based SHE design.

We wrote our Matlab implementation of our double-CRT NTRU SHE scheme using the Matlab fixed-point toolbox. The Matlab fixed-point toolbox also provides a path toward generated HDL implementations of our design that can be deployed for practical use on highly parallel computing hardware such as FPGAs. Part of our vision for the use of our SHE design is to develop an FPGA implementation of FHE [7, 8].

We ran our compiled implementation on a 64core server with 2.1GHz Intel Xeon processors and 1TB of RAM in a CentOS environment. Although we had access to many resources, we used at most 10 GB of memory and 20 cores during the evaluation of our software implementation.

We collected data on the runtime of the Encryption, EvalAdd, ComposedEvalMult, Decryption and Bootstrapping operations over selections of depth of computation supported and ring dimension. We ran 100 iterations of this collection procedure for each combination of t and ring dimension. We used different randomly selected key sets, plaintexts and encryption noise on every iteration to mitigate minor variations in performance that may arise due to these experimental random variables on every iteration. Tables of the raw mean runtime results can be seen in Tables 3 through 7 in Appendix A.

We collected data on the runtime of the Encryption, EvalAdd and ComposedEvalMult operations for settings of $t \in \{2, 4, 6, ..., 20\}$ and for ring dimensions $n \in \{512, 1024, 2048, 4096, 8192, 16384\}$. We collected data on the runtime of the Decryption operation of final ciphertexts, for computations with fresh (input) ciphertexts with ring dimensions $n \in \{512, 1024, 2048, 4096, 8192, 16384\}$ and depth of computation $t - 1$ for $t \in \{2, 4, 6, ..., 20\}$. Note that due to ring switching, decryption runtime is dependent only on the dimension of the final ciphertext, which is a function of the initial ciphertext and depth of computation. We collected data on the runtime of the Bootstrapping operation for settings of the "maximum" ring dimensions $n \in \{512, 1024, 2048, 4096, 8192, 16384\}$ ciphertexts are expressed in where the resulting ciphertext supports a depth one computation before another bootstrapping operations is required. As discussed in [1], the depth of computation required for bootstrapping is logarithmic in the ring dimension. We are currently exploring practical trade-offs associated with the impacts on the scheduling of bootstrapping to enable more computation between bootstrapping calls.

Our experimental results shows that run times grow linearly with ring dimension n and the ciphertext width t where $t-1$ is the depth of computation supported before bootstrapping or decryption could still be performed and have a high probability of recovering a correctly decrypted ciphertext. This makes intuitive sense because as we double either the ring dimension or the ciphertext width, we roughly double the amount of computation that needs to be performed with every Encryption, EvalAdd and ComposedEvalMult operation. Similar results hold for Decryption (Table 6) which shows a linear dependence of runtime on ring dimension, but under the assumption that decryption occurs after $t - 1$ ModReduction operations, including ModReduction operations bundled in ComposedEvalMult operations. Our initial results show that Bootstrapping runtime is similarly linear with respect to the maximum ring dimension. As compared to the results reported in [12, 23, 24], our FHE software implementation provides order-of-magnitude improvements in the runtime of the FHE operations.

6 Discussion and Looking Forward

Our FHE implementation is part of our long-term vision to support a general, practical and secure computing capability through a layered services architecture. Part of our vision is to provide software interfaces in our design for our highly optimized implementations of the basic FHE operations (KeyGen, Encrypt, EvalAdd, EvalMult, Decrypt) for users to construct general applications that require secure computation on encrypted data with automated calls to supporting operations such as Ring Switching,

Key Switching, Modulus Reduction and Bootstrapping. Inherent to this architecture vision is our FHE implementation of lattice-based computational primitives which form a lower layer of our envisioned architecture. We use these primitives such as ring addition, ring multiplication, modulus operations and the Chinese Remainder Transforms to run on commodity computing devices such as CPUs and FPGAs. We designed this modular approach to the implementation of the SHE operations and the underlying core primitives which allows us to 1) augment these operations with additional operations such as a bootstrapping operation (which enables FHE), or 2) replace the implementations of a subset of the operations or primitives as implementation advances are made.

A further aspect of our layered architecture vision is our ability to mix-and-match a computing substrate at lower levels of our architecture. Although not an immediate focus of the results reported here, the double-CRT representation, coupled with the 64-bit integer representation, simplifies parallelization of our FHE scheme for easier porting to other, high-performance and low-cost parallel computing environments such as FPGAs [7, 8] and possibly even GPUs [27]. If ported to a dedicated FPGA co-processor, the runtime of our underlying SHE/FHE implementation can be greatly improved upon as compared to the runtime of the corresponding interpreted CPU-only implementation which we discuss herein.

Taken together, we see our design and experimentation with our NTRU-based FHE scheme as a stepping-stone to a practical implementation of FHE through our layered architecture vision. Our primary path forward is to increasingly leverage the inherent parallelism of our design at multiple levels of our implementation. At a low level we are working to port our lattice-based primitives to operate on commodity FPGAs. This higher level parallelism offers the possibility of more practical SHE and FHE on both multi-core CPUs or multiple parallel FPGAs operating as "FHE co-processors".

Acknowledgement

The authors wish to acknowledge the helpful feedback and guidance of Prof. Chris Peikert in preparing the material discussed in this paper.

References

[1] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin Heidelberg, 2013.

[2] JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer Berlin Heidelberg, 2013.

[3] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[4] David Butenhof. *Programming with POSIX (R) threads*. Addison-Wesley Professional, 1997.

[5] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[6] JungHee Cheon, Jean-Sbastien Coron, Jinsu Kim, MoonSung Lee, Tancrde Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer Berlin Heidelberg, 2013.

[7] David Bruce Cousins, Kurt Rohloff, Chris Peikert, and Rick Schantz. SIPHER: Scalable implementation of primitives for homomorphic encryption - FPGA implementation using Simulink. In *Fifteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, HPEC '11, 2011.

[8] David Bruce Cousins, Kurt Rohloff, Chris Peikert, and Rick Schantz. An update on scalable implementation of primitives for homomorphic encryption - FPGA implementation using simulink. In *Sixteenth Annual Workshop on High Performance Embedded Computing (HPEC)*, HPEC '12, 2012.

[9] Yarkin Doroz, Yin Hu, and Berk Sunar. Homomorphic aes evaluation using ntru. Cryptology ePrint Archive, Report 2014/039, 2014. `http://eprint.iacr.org/`.

[10] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009. AAI3382729.

[11] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[12] Craig Gentry and Shai Halevi. Implementing Gentry's fully homomorphic encryption scheme. In Kenneth Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin / Heidelberg, 2011.

[13] Craig Gentry and Shai Halevi. HElib. `https://github.com/shaih/HElib`, 2014.

[14] Craig Gentry, Shai Halevi, Vadim Lyubashevsky, Chris Peikert, Joseph Silverman, and Nigel Smart, 2011. Personal communication.

[15] Craig Gentry, Shai Halevi, and Nigel Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer Berlin / Heidelberg, 2012.

[16] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Berlin Heidelberg, 1998.

[17] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339, 2011.

[18] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

[19] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-13190-5.

[20] MAGMA. V2.18-11, 2012. Available at `http://magma.maths.usyd.edu.au/magma/`.

[21] MATLAB. *R2012b*. The MathWorks Inc., Natick, Massachusetts, 2012.

[22] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post Quantum Cryptography*, pages 147–191. Springer, February 2009.

[23] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

[24] Henning Perl, Michael Brenner, and Matthew Smith. Poster: an implementation of the fully homomorphic Smart-Vercauteren cryptosystem. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 837–840, New York, NY, USA, 2011. ACM.

[25] Victor Shoup. *NTL: A Library for doing Number Theory*. Courant Institute, New York University, New York, NY, 2012. Available at `http://shoup.net/ntl/`.

[26] Joop van de Pol. Quantifying the security of lattice-based cryptosystems in practice. In *Mathematical and Statistical Aspects of Cryptography*, 2012.

[27] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. Accelerating fully homomorphic encryption on GPUs. In *Proceedings of the IEEE High Performance Extreme Computing Conference*, 2012.

[28] David Wu and Jacob Haven. Using homomorphic encryption for large scale statistical analysis. 2012.

A Experimental Results

Table 3: Encryption Runtime (ms) vs. Depth of Computation Supported and Ring Dimension for $p = 2$.

Dim. \ Depth	1	3	5	7	9	11	13	15	17	19
512	2.32	2.83	2.86	3.27	3.39	3.25	4.38	4.64	5.35	5.66
1024	3.87	5.33	5.17	5.98	5.68	5.63	6.94	8.40	9.04	9.20
2048	6.26	6.48	7.01	7.47	7.94	8.78	12.70	13.03	13.05	14.52
4096	12.08	12.27	13.04	14.87	17.38	17.65	20.73	17.46	21.57	22.13
8192	24.53	25.18	26.13	29.07	30.81	32.15	34.43	32.46	36.16	37.90
16384	52.30	55.02	58.05	59.71	60.29	61.98	63.44	64.99	69.96	72.89

Table 4: EvalAdd Runtime (ms) vs. Depth of Computation Supported and Ring Dimension for $p = 2$.

Dim. \ Depth	1	3	5	7	9	11	13	15	17	19
512	0.21	0.32	0.42	0.54	0.64	0.73	1.26	2.11	2.90	3.12
1024	0.30	1.04	0.47	0.57	0.72	0.74	1.40	2.72	2.85	2.93
2048	0.37	0.45	0.55	0.67	0.80	1.00	1.97	3.00	3.04	3.24
4096	0.56	0.65	0.74	0.91	1.92	2.07	2.25	2.43	3.73	3.54
8192	0.89	1.01	1.20	1.36	2.46	2.70	3.69	3.23	5.05	5.44
16384	1.58	1.82	2.12	2.39	3.99	4.19	4.27	4.77	7.16	7.29

Table 5: ComposedEvalMult Runtime (ms) vs. Depth of Computation and Ring Dim. for $p = 2$.

Dim. \ Depth	1	3	5	7	9	11	13	15	17	19
512	16.03	22.73	23.32	22.65	22.87	22.96	24.35	25.24	25.37	25.78
1024	29.15	37.85	39.05	39.11	38.79	39.24	39.49	39.59	39.52	39.68
2048	49.17	66.31	66.77	67.41	67.15	68.38	68.22	69.27	69.45	71.09
4096	99.56	140.42	140.71	141.42	141.26	142.75	143.52	145.51	144.61	148.31
8192	196.83	279.37	280.42	284.40	283.98	285.69	289.59	286.55	292.69	295.69
16384	463.92	623.19	622.74	628.87	630.43	633.37	639.52	642.80	651.20	659.88

Table 6: Decryption Runtime (ms) vs. Depth of Computation Supported and Initial Ring Dim. for $p = 2$.

Dim. \ Depth	1	3	5	7	9	11	13	15	17	19
512	0.40	0.26	0.13	0.14	0.10	0.10	0.06	0.06	0.06	0.06
1024	0.87	0.38	0.18	0.11	0.11	0.11	0.11	0.11	0.05	0.05
2048	1.92	0.84	0.38	0.38	0.22	0.22	0.22	0.22	0.12	0.12
4096	3.36	1.70	0.84	0.86	0.37	0.39	0.38	0.22	0.22	0.21
8192	7.22	3.43	1.67	1.72	0.85	0.87	0.86	0.87	0.39	0.40
16384	15.36	7.18	3.37	3.37	1.67	1.67	1.67	1.73	0.87	0.85

Table 7: Bootstrapping Runtime (s) vs. Ring Dimension for $p = 2$.

Ring Dimension	512	1024	2048	4096	8192	16384
Runtime (s)	5.8	13	26	60	125	275

9 LIST OF ABBREVIATONS AND ACRONYMS

2G	2nd generation Wireless Telecommunications
3G	3rd generation Wireless Telecommunications
4G LTE	4th generation Long-Term Evolution Wireless Telecommunications
AES	Advanced Encryption Standard
AND	logical And operation
ANSI C	American National Standards Institute Standard for the C programming language.
ASCII	American Standard Code for Information Interchange is a character-encoding scheme.
AWS	Amazon Web Services (Cloud Computing Service)
AXI	Advanced eXtensible Interface 4th generation. An open standard hardware interconnection bus used in FPGA designs.
AXI4	Advanced eXtensible Interface. An open standard hardware interconnection bus used in FPGA designs.
BRAM	Block RAM (in an FPGA)
CEM	composed Eval Mult
CPU	Central Processing Unit
CRT	Chinese Remainder Transform
DDoS	Distributed Denial of Service Computer Network Attack
DDR3	Double data rate type three synchronous dynamic random-access memory Direct
DMA	Memory Access (controller)
DSL	Digital subscriber line computer communications over telephone lines
EC2	Amazon Elastic Compute Cloud (Cloud Computing Service)
EKWS	encrypted keyword search
FFT	Fast Fourier Transform
FHE	Fully Homomorphic Encryption
FHEPU	Fully Homomorphic Encryption Processing Unit
FPGA	Field Programmable Gate Array
GB	Gigabytes
gcc	GNU C Compiler
Gen	Generation
GMP	GNU Multiple Precision Arithmetic Library
GNU	Open source software consorti: Gnu's Not Unix
GPGPU	General Purpose Graphical Processing unit
GPL	General Public License
GPU	Graphical Processing Unit
HDL	Hardware Design language (i.e. Verilog or VHDL)
HE	Homomorphic Encryption

HELib	Homomorphic Encryption Library (originally) from Shai Halevi and Victor Shoup
I/O	Input / Output
ICRT	Inverse Chinese Remainder Transform
iOS	iPhone Operating System
IP	In an FPGA context: Intellectual property, in a network context: Internet Protocol
kbs	kilo-bits per second
KHz	kilohertz
KWS	keyword search
LTV	López-Alt, Tromer and Vaikuntanathan crypto scheme
Mb/sec	megabit per second
MOD or mod	modulo operation
MPC	Multi Party Computation
mSec	milliSecond
NAND	logical Nand Function (Not(And))
NOT	logical Not Function
NTRU	NTRU is a patented and open source public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. Also “Number Theoretics R Us”
NTT	Number Theoretic Transform
OR	logical Or operation
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
ROM	Read Only Memory
SGMII	serial gigabit media-independent interface (Gigabit Ethernet Physical Layer)
SHE	Somewhat Homomorphic Encryption
SIPHER	Scalable Implementation of Primitives for Homomorphic EncRyption
SMC	Secure Multiparty Computation
TCP/IP	Transmission Control Protocol/Internet Protocol
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VOIP / VoIP	voice over Internet Protocol
XOR	logical exclusive Or operation