AFRL-AFOSR-VA-TR-2019-0131

Realizing the promise of proof-based verifiable computation

Michael Walfish
NEW YORK UNIVERSITY

04/24/2019
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

DISTRIBUTION A: Distribution approved for public release.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06-05-2019	Final Performance	15 Jul 2015 to 14 Jan 2019

4. TITLE AND SUBTITLE
Realizing the promise of proof-based verifiable computation

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-15-1-0302

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Michael Walfish, Andrew Blumberg

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NEW YORK UNIVERSITY
70 WASHINGTON SQUARE S
NEW YORK, NY 10012-1019 US

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTA2

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2019-0131

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This project designed and built systems that allow people to get useful work out of computers and components without having to trust, that is assume, that those components work properly. We have answered questions such as: how can we integrate custom hardware into a system, if the hardware manufacturer may be adversarial? How can a client outsource a computation to servers, if the servers are not guaranteed to return the right answer? How can we use peripheral devices with our computers, when those devices may be malicious, given that commodity computers are designed to trust physically connected hardware?

15. SUBJECT TERMS
Probabilistic Checkable Proofs, Verifiable Computation, Trusted Distributed Computing, Verifiable Cloud Computing, Computational Complexity

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			NGUYEN, TRISTAN
Unclassified	Unclassified	Unclassified	UU		19b. TELEPHONE NUMBER *(Include area code)* 703-696-7796

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Final report for "Realizing the promise of proof-based verifiable computation" (FA9550-15-1-0302)

PIs: Michael Walfish, Courant Institute, New York University (NYU), and Andrew J. Blumberg, Department of Mathematics, The University of Texas at Austin

Summary and overview of accomplishments

This project's top-level goal is to design and build systems that allow people to get useful work out of computers and components without having to trust, that is assume, that those components work properly. For example, how can we integrate custom hardware into a system, if the hardware manufacturer may be adversarial? How can a client outsource a computation to servers, if the servers are not guaranteed to return the right answer? How can we use peripheral devices with our computers, when those devices may be malicious, given that commodity computers are designed to trust physically connected hardware?

A central focus is systems in which one entity (a client or *verifier*) sends a program to another party (a server or *prover*), along with input. The prover returns the purported output, along with some auxiliary information (a certificate of correctness, or interactive answers to queries) that should allow the verifier to *check* that the prover computed correctly.

The grant funded the following accomplishments and "firsts":

- We initiated the study of "Verifiable ASICs" as an alternative to trusted foundries and produced a prototype, Zebra [17], that represents a milestone for proof-based verifiable outsourcing: the first hardware implementation of a probabilistic proof protocol, and the first time that the prover's cost was explicitly factored in, when considering whether verifiable outsourcing is worthwhile.

- The aforementioned paper received an award of "Distinguished student paper" from the IEEE Security & Privacy (Oakland) conference.

- We developed and published (at CCS, a leading venue in computer security) Giraffe [18], the first system for verifiable outsourcing that takes into account the actual cost of outsourcing—the verifier's running time, the cost of precomputation, the overhead of the prover—and demonstrates that the sum of these costs can be lower than simply running the computation on the verifier. This is a landmark in proof-based verifiable outsourcing.

- We formulated *The Efficient Server Audit Problem*, an abstract solution to the problem (called SSCO), and a concrete instantiation (called Orochi) [14]. This is the first work that, while operating in the verifiable outsourcing framework, comprehensively verifies execution of a concurrent server and comprehensively verifies the execution of legacy programs (such as web applications).

- The aforementioned publication received the award of Best Paper at SOSP 2017; this is arguably the highest recognition for a newly published systems paper.

- We developed a novel formulation of geometric distance and point-cloud matching computations as an SDP-based optimization problem, which achieves the best performance in the literature and is amenable to outsourcing [16].

- We developed a method for dimensionality reduction adapted to the space of phylogenetic trees which is amenable to outsourcing and supports inference algorithms on very large data sets [9, 21].

- We applied this foundational work on geometric inference in tree space to glioblastoma data, and described our results in two publications in Nature Genetics [12, 20].

- We developed very fast methods based on topological data analysis and metric geometry for recombination-rate inference and introgression detection in genomic data [7, 11]. These methods allow new kinds of biological inference and apply to much larger data sets than were previously tractable.
- We developed a statistical framework for testing whether sample distributions on arbitrary metric measure spaces are different [5, 8]. Our method is faster than the standard alternative (the "energy test") and has more power.
- We developed a method for denoising single-cell genomic data which achieves the best performance in the literature (measured on cell type inference tasks) and is amenable to outsourcing [2].
- We initiated the study of the abstract homotopy theory underlying topological data analysis [6]. Our work introduces a notion of "approximate weak equivalence" and relates it to the main structural theorems about persistent homology.
- We have developed and published (at SIGCOMM, the premier venue in computer networking) the Pretzel system [10], which hides emails from intermediate handlers (such as Webmail providers) yet still allows those handlers to perform functions over the email (such as spam filtering).
- We have developed Cinch, which defends against untrusted peripherals (for example, storage devices that masquerade as keyboards). Cinch is one of the first works to provide a systematic response to the threat of malicious peripherals on commodity OSes. Cinch appeared at Usenix Security [1].
- We have released a simplified version of our project's codebase for verifiable outsourcing [13]. This is the only released "front-end" for verifiable computation that integrates cleanly with various backends.
- We released the code that implements Orochi, on github.
- We released the code that implements the denoising algorithm for single-cell data, on github.
- PI Walfish was invited to lead the Bar-Ilan Cryptography Winter School tutorials on verifiable computation in January, 2016. He presented 4.5 hours of tutorials over several days, together with extensive written materials [3]. Videos are on Youtube (for example, [19]).
- PI Walfish co-organized a DIMACS workshop on secure computation, which brought together theorists and applied researchers for two days in July 2017, and was successful.
- PI Walfish, as a co-organizer, submitted a proposal for a semester-long program to run at the Simons Institute in the fall of 2019. The proposal was accepted.
- PI Blumberg (joint with R. Rabadan) wrote a book on topological data analysis for genomic data that incorporates some of the work on geometric inference described above.

Activities

Verifiable ASICs [17]. The motivation for this work is that a manufacturer of custom hardware (an ASIC) can undermine the intended execution of that hardware; high-assurance execution thus requires controlling the manufacturing chain. However, a trusted platform might be orders of magnitude worse in performance or price than an advanced, untrusted platform.

Our project explores an alternative: using verifiable computation (VC), an untrusted ASIC computes proofs of correct execution, which are verified by a trusted processor or ASIC. One of the technical challenges is that, in the present setting, the prover and verifier together must impose less overhead than the baseline alternative of running the given computation directly on the trusted platform.

We respond to this challenge by designing and implementing physically realizable, area-efficient, high throughput ASICs (for a prover and verifier), in fully synthesizable Verilog.

The system, called *Zebra*, is based on the CMT interactive proof protocol (Cormode et al., ITCS 2012); instantiating Zebra has required a blend of new observations about CMT, careful hardware design, and attention to architectural challenges. We have carefully measured and evaluated Zebra; for a class of real computations, it indeed poses less overhead than executing directly on the trusted platform.

Zebra is a milestone in verifiable computation, for two reasons. (1) It is the first *hardware* design and implementation of a probabilistic proof system. (2) It is the first work to identify a setting in which one can simultaneously capture the "cost" of the prover and verifier together, and to give an implementation of the prover and verifier for which this quantity is less expensive than having the verifier compute on its own. Zebra is also a milestone in the field of hardware Trojans: it works in a much stronger threat model than prior defenses.

Full accounting for verifiable outsourcing [18]. Systems for verifiable outsourcing incur costs for a *prover*, a *verifier*, and precomputation; outsourcing makes sense when the combination of these costs is cheaper than not outsourcing. Yet, when prior works impose quantitative thresholds to analyze whether outsourcing is justified, they generally ignore prover costs. The Verifiable ASICs framework is the other way around: it pays for the prover but its cost calculations assume away precomputation.

In this work, we build a new system, called *Giraffe* [18] which is intended for the VA setting, and which can "win", *even when accounting for all three costs*. Giraffe is the first system to reach this landmark.

Giraffe has two high-level aspects: a back-end (a probabilistic proof protocol) and a front-end (which transforms from high-level programs to the representation that the back-end works over). Giraffe's back-end is based on an existing probabilistic proof protocol [15, §7], which is geared to the data parallel context. Giraffe improves this protocol, and achieves asymptotic optimality for the prover: the prover runs in time *linear* in the number of steps in the computation. The back-end work for Giraffe includes further protocol improvements and a *design template* that automatically instantiates physically realizable, efficient, high-throughput ASIC designs for the prover and verifier. The design template employs several new hardware structures; these structures are geared to the data flows and parallelize easily.

The front-end work consists of program analysis techniques that adapt programs to the form required by the back-end. One of them is *squashing*, in which a serial program is represented as a data parallel arithmetic circuit (by treating intermediate values that are transferred from loop to loop as "inputs" to the circuit). Another is *slicing*, which takes as input an abstract cost model and a program, automatically identifies amenable *subregions* of the program to outsource—even when the program as a whole is not amenable to outsourcing—and generates glue code to sew the outsourced pieces into the rest of the program.

The Efficient Server Audit Problem [14]. The high-level goal of this project is to answer: how can we tell if a server is executing a program consistent with the program's source code? The setup is related to verifiable outsourcing; here, there is an emphasis on solutions that scale to real servers and programs.

The abstract problem is as follows. The actors and components are depicted in Figure 1. A *principal* chooses or develops a *program*, and deploys that program on a powerful but untrusted *executor*. Clients (the outside world) issue *requests* (inputs) to the executor, and receive *responses* (outputs). A response is

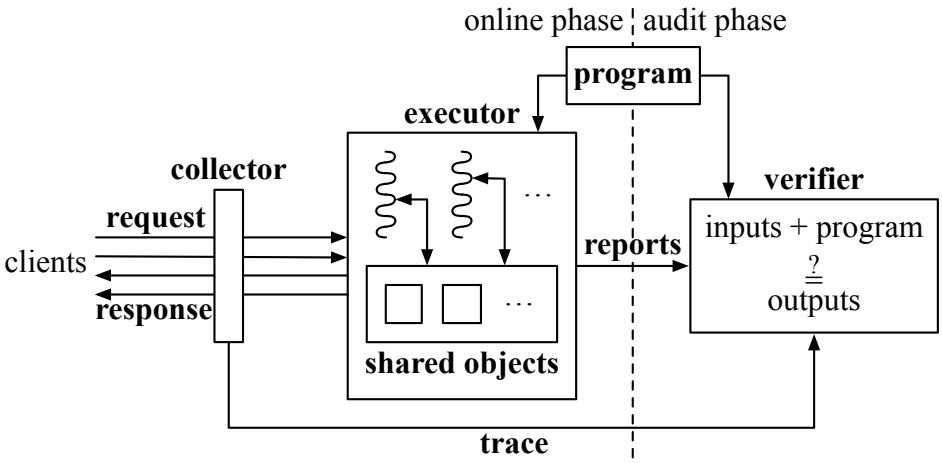

FIGURE 1—The Efficient Server Audit Problem. The objects abstract shared state (databases, key-value stores, memory, etc.). The technical problem is to design the verifier and the reports to enable the verifier, given a trace and a program, to efficiently validate (or invalidate) the contents of responses.

supposed to be the output of the program, when the corresponding request is the input. But the executor is untrusted, so the response could be anything.

A *collector* captures an ordered list, or *trace*, of requests and responses. We assume that the collector does its job accurately, meaning that the trace exactly records the requests and the (possibly wrong) responses that actually flow into and out of the executor.

The executor maintains *reports* whose purpose is to assist an audit; like the responses, the reports are untrusted.

Periodically, the principal conducts an audit; we often refer to the audit procedure as a *verifier*. The verifier gets a trace (from the accurate collector) and reports (from the untrusted executor). The verifier needs to determine whether executing the program on each input in the trace truly produces the respective output in the trace.

Two features of our setting makes this determination challenging. First, the verifier is much weaker than the executor, so it cannot simply re-execute all of the requests.

The second challenge arises from concurrency: the executor is permitted to handle multiple requests at the same time (for example, by assigning each to a separate thread), and the invoked program is permitted to issue *operations* to *objects*. An object abstracts state shared among executions, for example a database, key-value store, or memory cells (if shared). In this setting, there is a kind of explosion: given a trace—in particular, given the ordering of requests and responses in the trace, and given the contents of requests—the number of valid possibilities for the contents of responses could be immense. This is because an executor's responses depend on the contents of shared objects; as usual in concurrent systems, those contents depend on the operation order, which depends on the executor's internal scheduling choices.

Somehow, the reports, though unreliable, will have to help the verifier efficiently tell the difference between valid and invalid traces.

To address the above problems, we introduce an abstract solution, called SSCO. SSCO assumes that there is similarity among the executions, in particular that there are a relatively small number of control flow paths induced by requests.

SSCO introduces several novel techniques. The first is is *SIMD-on-demand execution*. The verifier re-executes all requests, in an accelerated way. For a group of requests that the executor asserts have the same control flow, the verifier executes a "superposition": instructions with identical operands across requests are performed once, whereas instructions with different operands are executed individually and merged into the superposed execution.

A second technique is *simulate-and-check*. To motivate this technique, we ask: how can the verifier re-execute *reads* of persistent or shared state? Because it re-executes requests out of order, it cannot physically

re-invoke operations on such state, but neither can it trust reports that are allegedly the originally read values. Instead, the executor (purportedly) logs each operation's operands; during re-execution, the verifier simulates reads, based on the most recent write entry in the logs, and checks the logged writes opportunistically.

A third technique is *consistent ordering verification*. The prior technique (simulate-and-check) makes sense only if alleged operations can be ordered consistent with observed requests and responses. To this end, the verifier builds a directed graph with a node for every external observation or alleged operation, and checks whether the graph is acyclic. This step incorporates an efficient algorithm for converting a trace into a time precedence graph. This algorithm may have broader applicability.

In a system called Orochi, we instantiate SSCO for PHP-based web applications; this involves modifying the PHP interpreter and adapting simulate-and-check to databases (the verifier replays a transaction log into a versioned database, checks that the queries generated during re-execution match the initially replayed ones, and exploits deduplication opportunities).

For several Web applications, our implementation of Orochi achieves between 5.6–10.9$\times$ speedup versus verification by naive replay, at the cost of roughly 10% overhead during normal execution. An additional contribution in Orochi is a rigorous proof of correctness.

Verifying geometric inference problems. The high-level goal of this thrust is to verify geometric inference problems in a variety of domains. We want to develop algorithms for such problems that are (a) efficient enough to be applicable to real data at scale, and (b) amenable to high-performance special-purpose verifiable outsourcing.

The metric space of phylogenetic trees [4] provides a geometric context for studying evolution. Unfortunately, the geometry exhibits perverse behavior in high dimensions (i.e., large numbers of leaves). We developed a novel dimensionality reduction technique for spaces of phylogenetic trees that decomposes a single large tree into a forest of trees with a small number of leaves. This procedure was applied to genomic data coming from brain cancer (glioblastomas), resulting in two publications in Nature Genetics [12, 20]. Subsequently, PI Blumberg's student G. Grindstaff and her collaborator M. Owen studied algorithms for the inverse problem (recovering the original tree from the projection), completely characterizing the space of solutions to the problem [9].

Motivated by consideration of point clouds in tree space, we studied the problem of testing whether two sample distributions on a metric measure space are different [5]. There are standard non-parametric approaches in Euclidean space; we have developed a method that works in arbitrary metric measure spaces and is computationally tractable. Experimental evaluation suggests that this method is substantially more powerful than the "energy test" family of comparison techniques. PI Blumberg's student G. Grindstaff has characterized the group of isometries for the space of phylogenetic trees, which is a key input to the application of this algorithm [8]. We also made progress on problems associated to statistical inference problems in non-Euclidean metric measure spaces, notably associated to density estimation.

Another natural question that arises from the tree dimensionality reduction procedure is a metric comparison of point clouds in arbitrary metric spaces. We developed an semidefinite programming relaxation for computing the Gromov-Hausdorff distance between two point clouds [16]. Although the resulting algorithm has the best performance in the literature, it is nonetheless not truly practical for data sets larger than a few hundred points. Subsequently, we have developed a non-convex Lagrangian approximation scheme which substantially improves performance.

Applications to genomic inference. Genomics provides an important testbed for our efficient geometric inference algorithms. We have developed an extremely fast verifiable algorithm for estimating recombination rates based on computing simple geometric features of sequencing data. Our selection of topological features is based on a coalescent theory analysis. This algorithm allows analysis of data sets that were previously too large to study. In related work which adapts ideas from our tree dimensionality reduction procedure, we introduced a new algorithm for detecting and estimating introgression rates in gene tree data. The algorithm provides new kinds of estimates about introgression rates and also allows estimation in very large data sets.

This algorithm has been applied to resolve questions about ancient lineages in butterflies.

We have developed a novel technique for denoising single-cell data using random matrix theory [2]. Single cell data (using current technology) is a context in which traditional dimensionality reduction techniques work very badly; the ambient dimension and number of data points are commensurate and the data tends to be very sparse. Our approach uses predictions from random matrix theory about the distribution of singular values and the impact of sparsity to identify a low-rank signal basis. Our new technique is very fast and achieves the best performance in the literature on known cell-type identification tasks. Additionally, we have applied it to discover new types of human blood cells.

Computing over encrypted email [10]. Emails today are often encrypted, but only between mail servers—the vast majority of emails are exposed in plaintext to the mail servers that handle them. While better than no encryption, this arrangement leaves open the possibility of attacks, privacy violations, and other disclosures. Publicly, email providers have stated that default end-to-end encryption would conflict with essential functions (spam filtering, etc.), because the latter requires analyzing email text.

We have built a system, *Pretzel*, which demonstrates that there is no conflict.

In Pretzel, senders encrypt email using an end-to-end encryption scheme, and the intended recipients decrypt and obtain email contents. Then, the email provider and each recipient engage in a *secure two-party computation* (2PC), a family of cryptographic protocols in which one or both parties learn the output of an agreed-upon function, without revealing the inputs to each other. For example, a provider supplies its spam filter, a user supplies an email, and both parties learn whether the email is spam while protecting the details of the filter and the content of the email.

The challenge in Pretzel comes from the 2PC component. There is a tension between expressive power (the best 2PC schemes can handle any function and even hide it from one of the two parties) and cost (those schemes remain exorbitant, despite progress in lowering the costs). Therefore, in designing Pretzel, we decided to make certain compromises to gain even the possibility of plausible performance: baking in specific algorithms, requiring both the algorithms' logic and the model features to be exposed (model parameters are hidden), and incurring per-function design work.

The work of Pretzel is refining and adapting a 2PC protocol that is geared to linear operations, as well as attending to a number of systems issues, both to keep costs down and to defend against adversarial parties. Our experimental evaluation of a prototype of Pretzel demonstrates that email can be encrypted end-to-end *and* providers can compute over it, at tolerable cost: clients must devote some storage and processing, and provider overhead is roughly $5\times$ versus the s

Defending against malicious peripherals [1]. The motivation for this work is that inexpensive and powerful peripherals, which attach to plug-and-play buses, have made attacks on host computers by malicious peripherals easy to mount. Making matters worse, commodity operating systems lack coherent defenses, and users are often unaware of the scope of the problem.

The *Cinch* system is a pragmatic response to this threat. Cinch uses virtualization to attach peripheral devices to a logically separate, untrusted machine, and includes an interposition layer between the untrusted machine and the protected one. This layer regulates interaction with devices and enforces security policies that are easily configured and extended by users. Cinch integrates with existing OSes, enforces policies that thwart real world attacks, and has low overhead.

Cinch is, we believe, one of the first works to attempt a systematic response to the threat of malicious peripherals on commodity OSes (as opposed to clean slate operating systems developed in the defense context).

References

[1] S. Angel, R. S. Wahby, M. Howald, J. B. Leners, M. Spilo, Z. Sun, A. J. Blumberg, and M. Walfish. Defending against malicious peripherals with Cinch. In *USENIX Security Symposium*, Aug. 2016.

[2] L. Aparicio, M. Bordyuh, A. J. Blumberg, and R. Rabadan. Quasi-universality in single-cell sequencing data, 2018. biorxiv, https://doi.org/10.1101/426239.

[3] Bar-Ilan Winter School on Cryptography. http://crypto.biu.ac.il/6th-biu-winter-school, Jan. 2016.

[4] L. J. Billera, S. P. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

[5] A. J. Blumberg, P. Bhaumik, and S. G. Walker. Testing to distinguish measures on metric spaces, Feb. 2018. arXiv:1802.01152, https://arxiv.org/abs/1802.01152.

[6] A. J. Blumberg and M. Lesnick. Universality of the homotopy interleaving distance, 2017. arXiv:1705.01690, https://arxiv.org/abs/1705.01690.

[7] N. B. Edelman, P. Frandsen, M. Miyagi, B. Clavijo, J. Davey, R. Dikow, G. García-Accinelli, N. Patterson, D. Neafsey, R. Challis, S. Kumar, G. Moreira, C. Salazar, B. Counterman, R. Papa, A. Whibley, K. Dasmahapatra, M. Kronforst, M. Joron, C. D. Jiggins, W. O. McMillan, A. J. Blumberg, J. Wakeley, D. Jaffe, , and J. Mallet. Genomic architecture and introgression shape a butterfly radiation, 2018. biorxiv, http://doi.org/10.1101/466292.

[8] G. Grindstaff. The isometry group of phylogenetic tree space is S_n, 2019. arxiv:1901.02982, https://arxiv.org/abs/1901.02982.

[9] G. Grindstaff and M. Owen. Geometric comparison of phylogenetic trees with different leaf sets, 2018. arxiv:1807.04235, https://arxiv.org/abs/1807.04235.

[10] T. Gupta, H. Fingler, L. Alvisi, and M. Walfish. Pretzel: Email encryption and provider-supplied functions are compatible. In *ACM SIGCOMM*, Aug. 2017.

[11] D. P. Humphreys, M. R. McGuirl, M. Miyagi, and A. J. Blumberg. Fast estimation of recombination rates using topological data analysis. *Genetics*, 211(4):1191–1204, 2019.

[12] J.-K. Lee, J. Wang, J. K. Sa, E. Ladewig, H.-O. Lee, I.-H. Lee, H. J. Kang, D. S. Rosenbloom, P. G. Camara, Z. Liu, P. van Nieuwenhuizen, S. W. Jung, S. W. Choi, J. Kim, A. Chen, K.-T. Kim, S. Shin, Y. J. Seo, J.-M. Oh, Y. J. Shin, C.-K. Park, D.-S. Kong, H. J. Seol, A. Blumberg, J.-I. Lee, A. Iavarone, W.-Y. Park, R. Rabadan, and D.-H. Nam. Spatiotemporal genomic architecture informs precision oncology in glioblastoma. *Nature Genetics*, 49:594–599, 2017.

[13] A system for verifying outsourced computations. Simplified release of the main Pepper codebase. https://github.com/pepper-project/pequin.

[14] C. Tan, L. Yu, J. B. Leners, and M. Walfish. The efficient server audit problem, de-duplicated execution, and the web. In *ACM Symposium on Operating Systems Principles (SOSP)*, Nov. 2017.

[15] J. Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, Aug. 2013.

[16] S. Villar, A. S. Bandeira, A. J. Blumberg, and R. Ward. A polynomial-time relaxation of the Gromov-Hausdorff distance, 2016. arXiv:1610.05214, https://arxiv.org/abs/1610.05214.

[17] R. S. Wahby, M. Howald, S. Garg, abhi shelat, and M. Walfish. Verifiable ASICs. In *IEEE Symposium on Security & Privacy (Oakland)*, May 2016.

[18] R. S. Wahby, Y. Ji, A. J. Blumberg, abhi shelat, J. Thaler, M. Walfish, and T. Wies. Full accounting for verifiable outsourcing. In *ACM Conference on Communication and Computing Security (CCS)*, Oct. 2017.

[19] M. Walfish. Introduction and overview of verifiable computation. https://www.youtube.com/watch?v=qiusq9R8Wws.

[20] J. Wang, E. Cazzato, E. Ladewig, V. Frattini, D. I. S. Rosenbloom, S. Zairis, F. Abate, Z. Liu, O. Elliott, Y.-J. Shin, J.-K. Lee, I.-H. Lee, W.-Y. Park, M. Eoli, A. J. Blumberg, A. Lasorella, D.-H. Nam, G. Finocchiaro, A. Iavarone, and R. Rabadan. Clonal evolution of glioblastoma under therapy. *Nature Genetics*, doi:10.1038/ng.3590, June 2016.

[21] S. Zairis, H. Khiabanian, A. Blumberg, and R. Rabadan. Genomic data analysis in tree spaces, 2016. arXiv:1607.07503, https://arxiv.org/abs/1607.07503.