AFRL-RY-WP-TR-2020-0162

A SECURITY FRAMEWORK FOR LOGIC LOCKING THROUGH LOCAL AND GLOBAL STRUCTURAL ANALYSIS

Christopher Taylor
The Ohio State University

JUNE 2020
Final Report

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
June 2020	Thesis	6 May 2020 –6 May 2020

4. TITLE AND SUBTITLE
A SECURITY FRAMEWORK FOR LOGIC LOCKING THROUGH LOCAL AND GLOBAL STRUCTURAL ANALYSIS

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Christopher Taylor

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Ohio State University
Columbus, OH 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYDT

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2020-0162

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number 88ABW-2020-1654, Clearance Date 6 May 2020. © 2020 Christopher Taylor. Presented in partial fulfillment of the requirements for the Degree Doctor of Philosophy in Electrical and Computer Engineering in the Graduate School of The Ohio State University. This work was funded in whole or in part by Department of the Air Force. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. Report contains color.

14. ABSTRACT

With the globalization of the semiconductor industry and increased reliance on intellectual property (IP) blocks in integrated circuit (IC) design; malicious modifications, IP theft, and cloning have started to pose a significant economic and security threat. To mitigate this risk, logic locking (LL) techniques have been proposed to obscure the chip functionality and increase the difficulty to insert a trigger-based change via a hardware trojan. This is accomplished through the introduction of localized key gates, which corrupt the IC's function unless the correct key is supplied. The effectiveness of any LL technique, however, depends on the target design, the extent of locking, and where the locking elements are placed. Current attacks on LL focus primarily on Boolean satisability problem (SAT) solvers, which require the use of a fully operational chip (oracle) and rely solely on the input and output data through functional testing. To the authors' best knowledge, no current attacks exploit the design's underlying structure, vast amount of repetition, or circuit reuse. In this work, we propose a systematic method, borrowed from the network analysis domain, to analyze and exploit the local and global structure of circuits. The methods presented in this work demonstrates that LL minimally effects the underlying structure, allowing for circuit identification and key bit prediction without the need of an oracle. Moreover, this work also presents a framework in which to capture the security level of LL based on the amount of information leakage through our analysis techniques. Additionally, the framework can be expanded to incorporate other attack methods to create an overall security assessment of any implemented LL. To this end, the analyses and theory introduced in this work demonstrate the need for new comprehensive LL techniques, and proposes the method in which to validate their security.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Pompei Orlando
Unclassified	Unclassified	Unclassified	SAR	134	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

A Security Framework for Logic Locking Through Local and Global Structural Analysis

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Christopher Taylor, B.S.E.E, M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2020

Dissertation Committee:

Dr. Waleed Khalil, Advisor

Dr. Hesham El Gamal

Dr. Xinmiao Zhang

Dr. Radu Teodorescu

Abstract

With the globalization of the semiconductor industry and increased reliance on intellectual property (IP) blocks in integrated circuit (IC) design; malicious modifications, IP theft, and cloning have started to pose a significant economic and security threat. To mitigate this risk, logic locking (LL) techniques have been proposed to obscure the chip functionality and increase the difficulty to insert a trigger-based change via a hardware trojan. This is accomplished through the introduction of localized key gates, which corrupt the IC's function unless the correct key is supplied. The effectiveness of any LL technique, however, depends on the target design, the extent of locking, and where the locking elements are placed. Current attacks on LL focus primarily on Boolean satisfiability problem (SAT) solvers, which require the use of a fully operational chip (oracle) and rely solely on the input and output data through functional testing. To the authors' best knowledge, no current attacks exploit the design's underlying structure, vast amount of repetition, or circuit reuse. In this work, we propose a systematic method, borrowed from the network analysis domain, to analyze and exploit the local and global structure of circuits. The methods presented in this work demonstrates that LL minimally effects the underlying structure, allowing for circuit identification and key bit prediction without the need of an oracle. Moreover, this work also presents a framework in which to capture the security level of LL based on the amount of information leakage through our analysis techniques.

Additionally, the framework can be expanded to incorporate other attack methods to create an overall security assessment of any implemented LL. To this end, the analyses and theory introduced in this work demonstrate the need for new comprehensive LL techniques, and proposes the method in which to validate their security.

This is dedicated to my wife, for your support and willingness to upend our lives in pursuit of this opportunity.

Acknowledgments

First, I would also like to thank my committee members Dr. Hesham El Gamal, Dr. Xinmiao Zhang, and Dr. Radu Teodorescu for their contributions to my candidacy and final defense examinations. In addition, special thanks goes to Dr. El Gamal as his insights into information theory and the many meetings we have had over the years has truly defined and expanded this work. Furthermore, I would like to thank Dr. Swarup Bhunia, for his contributions and advice the past few years.

To my lab mates, while at times you may not have contributed to my productivity, you have truly made my experience at Ohio State some of the best years of my life.

I would extend a special thanks to Dr. Brian Dupaix, who provided the foundation of my PhD work. The long conversations on hardware security we have had the past few years have finally come to fruition.

I would also like to thank the Air Force Research Labs (AFRL) for their fellowship program funding my time to pursue this opportunity.

Lastly, I would like to express my deepest gratitude to my advisor Dr. Waleed Khalil, who's teaching and guidance contributed not only to this work, but to my understanding of Digital, Analog, and RF circuits as well.

Vita

2012	B.S.E.E., Wright State University
2013-2014	Graduate Teaching Assistant, Wright State University
2014	M.S., Wright State University
2015-2016	Adjunct Professor, Wright State University
2019	M.S., The Ohio State University
2014-present	Associate Research Electronics Engineer, Air Force Research Laboratory

Publications

Research Publications

C. Taylor, S. Bhunia, B. Dupaix, W. Khalil "A Structural Analysis of Logic Locking". *GOMACTech, Mar. 2020.*

S.Ellicot, **C. Taylor**, W. Khalil "Anti-SAT Removal Attack Using Structural Analysis with Community Detection Algorithms". *GOMACTech, Mar. 2020.*

R. Fragasse, R. Tantawy, S. Smith, T. Specht, Z. Taghipour, P. Van Hooser, **C. Taylor**, et al. "Advancing Uncooled Infrared Imagers Using An Open-Circuit Voltage Pixel". *GOMACTech, Mar. 2020.*

R. Fragasse, R. Tantawy, S. Smith, T. Specht, Z. Taghipour, P. Van Hooser, **C. Taylor**, et al. "Advancing Uncooled Infrared Imagers Using An Open-Circuit Voltage Pixel". *Latin American Symposium on Circuits and Systems, Feb. 2020.*

C. Taylor, B. Dupaix, W. Khalil "Towards Measurement of Hardware Complexity". *GOMACTech, Mar. 2018.*

C. Taylor, P.L. Orlando, B. Dupaix "Automated Register Mapping Via Logic Sequence Analysis". *GOMACTech, Mar. 2017.*

Fields of Study

Major Field: Electrical and Computer Engineering

Specialization: Analog and Rf Electronics

Table of Contents

Page

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction

1.1 Motivation

The Integrated Circuit (IC) market has witnessed a significant shift in recent decades as the cost to stand up and maintain a foundry has skyrocketed [43]. This has directly led to horizontal integration of the semiconductor industry, accelerated growth, and gave rise to fabless semiconductor companies. Additionally, the United States suffered a dramatic drop of onshore IC production capabilities while other countries, such as China, have rapidly increased their capacity, as shown in Fig. 1.1. Nevertheless, this shift has increased the reliance on third party foundries, which has opened the door to a number of vulnerabilities in many facets of the fabrication process. These vulnerabilities have created a need to ensure the trustworthiness of fabricated ICs and the protection of Intellectual Property (IP). Consequently, a foundry can easily reverse engineer and steal IP or over produce a chip to sell for profit [44]. Moreover, the foundry can also insert a Hardware Trojan (HT) or make any other desired modification with relative ease [2].

The unfortunate reality is that to maintain access to state of the art fabrication facilities there are few options and the use of third party facilities is almost a necessity. For the Department of Defense (DoD), this is especially troubling as they are required

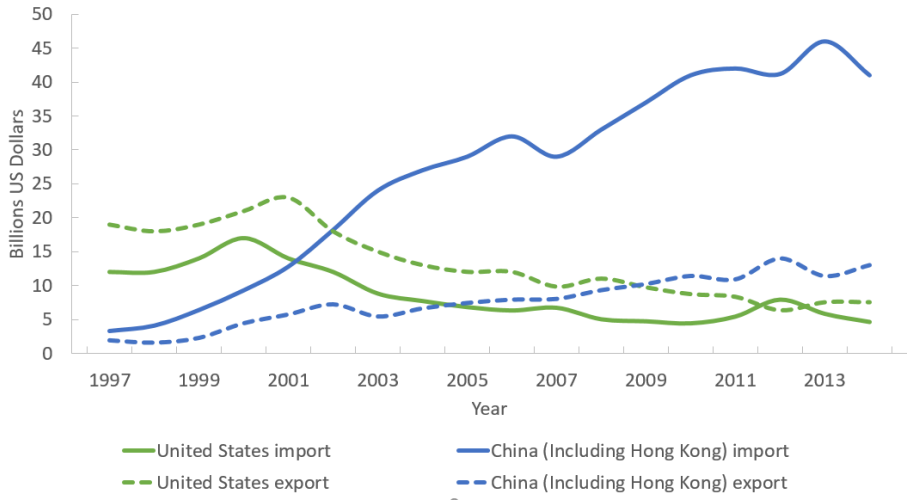


Figure 1.1: Import/Export IC market valuation from 1997 to 2014 for the U.S.A and China [38]

to maintain systems for upwards of 50 years, and threats, such as the suspected Syrian radar kill switch [4], are of real concern. The DoD also often resorts to lifetime buys of ICs as purchasing replacement parts during the extended system lifespan is nearly impossible. The longevity and sensitivity of DoD applications make quality, reliability and trustworthiness of the ICs that are installed in systems of the utmost importance. Alternative options are few as the DoD does not drive enough of the IC market to keep production on shore or stand up their own fabrication facilities, so they fall victim to commercial market forces. This problem is exacerbated by the increased reliance on ICs in autonomous vehicles, Internet of Things (IOT), recent headlines such as the Spectre and Meltdown vulnerabilities in processors [24], and malicious chips found in motherboards [34]. With the current state of the IC market and the desire to use advanced processing nodes, the security of our ICs has become an important issue

and has lead to significant increases in trusted and assured microelectronics research and spending.

1.2 Background into Hardware Security

Hardware security is a relatively new research field as it has only gained significant traction in the last decade. Hardware was once thought safe, due to vertical production and the difficulty of stealing IP or modifying a physical IC after it has been fabricated and deployed. The old adage of "gates, guards, and guns" was entirely sufficient and a straight forward way to protect ICs and worked well until recently. However, the idea of the hardware being secure only holds true if the entire manufacturing process is trusted. Moreover, the issues hardware engineers are facing today could be compared to software developers, if software developers were unable to compile their own code. To parallel the IC industry, code would need to be sent to a third party company overseas to be compiled. Trusting that the returned compiled code is free from modifications and the IP hasn't been stolen would be a hard pill to swallow for most software companies, but is the current reality for hardware companies and the DoD.

1.2.1 Hardware Trojans

In anticipation of the current IC fabrication environment and the particular susceptibility to the DoD, the Defense Advanced Research Projects Agency (DARPA) launched the Trust in Integrated Circuits program in 2006 to provide a mechanism to independently verify IC integrity [13]. The Trust program's main concern was Hardware Trojans (HT) and the ability to detect them. In general, HTs are similar in concept to their software counterparts in that they are hidden and malicious pieces

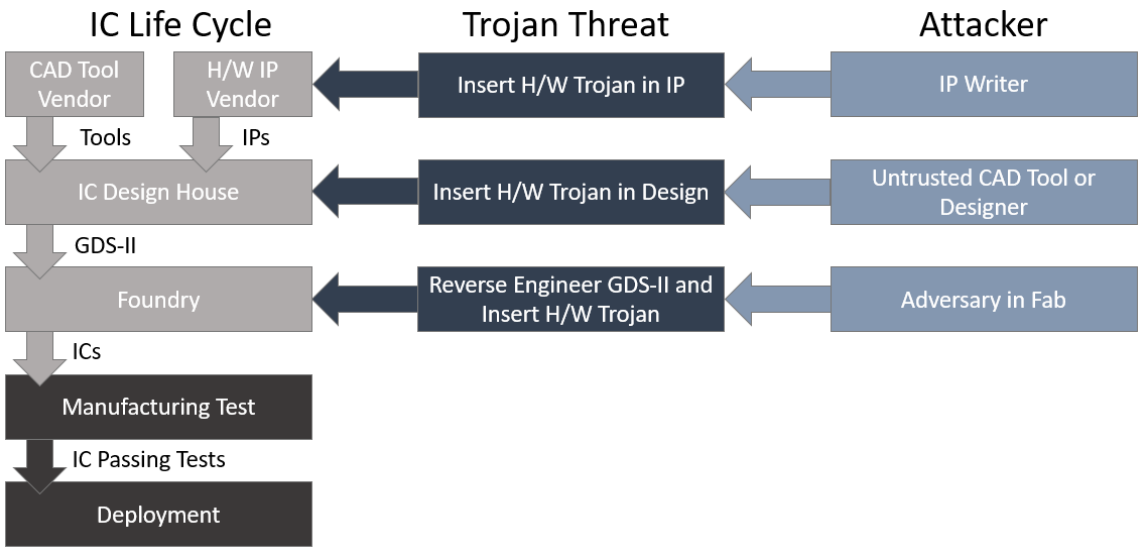


Figure 1.2: Hardware Trojan Attack Vectors [9]

of hardware that intend to steal information or disrupt normal operation. There are several vectors an adversary can use to insert a HT into an IC, as shown in Fig. 1.2. Ultimately, a variety of reasons exist to insert Trojans into an IC, such as to gain a competitive edge in the market, disrupt major national infrastructure, or leak secret information from inside the chip.

HTs are described by their payloads and triggers [9], where the trigger is the mechanism in which to activate the HT and the payload is the effect the HT has on the chip. While triggers can manifest in a number of ways, the most common are rare node values or rare sequences. In either case, the core idea is the HT only activates under a very specific input vector or sequence of input vectors. For example, the rare vector can either be triggered randomly during normal chip operation or the malicious actor might have access to implement the triggers while the chip is in use. Typical testing methods would never activate the triggers, leaving the HT undetectable. In

the case of the ISCAS'85 c880 benchmark circuit, which only has 451 gates, there are 10^9 possible HT trigger conditions [9]. Another common trigger is referred to as a time bomb. A large counter is inserted, only activating the HT after a predetermined amount of time. Initial burn in and testing would never run a chip for a long enough period to activate the HT, and hence it would go unnoticed. Once a HT is triggered, it delivers its payload. Payloads also vary and the owner of the IC may not even know a HT has been activated depending on the type of payload. A HT payload may alter the output giving a slightly corrupted answer. Other types of payloads can leak secret information such as the key to an AES encryption core onto an output pin. Another type may elevate the permissions of the user allowing them unrestricted access to the chip or system. The amount of triggers and payloads is only limited by the imagination of the malicious actor who places them, and due to the sheer size of modern ICs, finding and preventing Trojans is like trying to find a needle in a stack of needles.

1.2.2 IP Theft

Another threat of third party IC fabrication is IP theft. Consequently, any company who sends their designs out for fabrication runs a significant risk of those designs being stolen and reproduced without permissions. With the increase in reliance of offshore fabrication facilities, this risk of IP theft has only increased, in large part due to traditionally lax IP laws in many other parts of the world. As an example, the easiest and most straight forward type of IP theft is overproduction. Specifically, the fabrication facility overproduces the chips ordered, selling the additional chips for a tidy profit. Another type of IP theft is done through reverse engineering, which

is more complex and time consuming. In this case, a malicious actor at the fabrication facility who has access to the raw design can reverse engineer and steal the IP. Afterwards, the IP can either be used in other chips or sold to other companies. Additionally, IP theft also poses a national security threat for certain government and military application, as the integrity of a critical systems can become compromised. While it is the easiest attack vector, the fab is not the only entity capably of reverse engineering a design. Chips can also be reverse engineered after fabrication from the chip itself. For example, the Trust program used a chip delayering and image capture process from a fabricated chip to reconstruct a netlist in which to search for Trojans. Additionally, the same methods could be applied towards reverse engineering. Moreover, extracting IP is also not unheard of, as companies exist solely to reverse engineer an IC and look for IP in cases of patent infringement [3].

1.3 Logic Obfuscation

To protect from IP theft, malicious modifications, and HTs, the concept of logic obfuscation, also known as Logic Locking (LL), gate camouflaging, logic encryption, and function locking has been proposed. LL is intended to hide the functionality of a circuit and can be defined by the type of circuit it modifies e.g. sequential or combinational, or by the abstraction level where it is implemented, e.g. behavioral, structural, or physical, as illustrated in Fig. 1.3 [5]. Sequential obfuscation attempts to alter the Finite State Machine (FSM) [12], while combinational obfuscation, as the name implies, alters the combinational logic in a circuit [36]. In terms of abstraction level, behavioral obfuscations modify the HDL prior to synthesis and are found in some sequential obfuscation types [30]. Structural obfuscation is implemented on

the gate level netlist which is generated from the synthesis process. Finally, physical obfuscation is based on modifications to the physical representation of the circuit, such as the split manufacturing technique [32]. The most common obfuscation type and the subject of this work is combinational LL at the structural level. Combinational LL can further be divided into several sub-categories depending on the method of concealment and the means in which to program it.

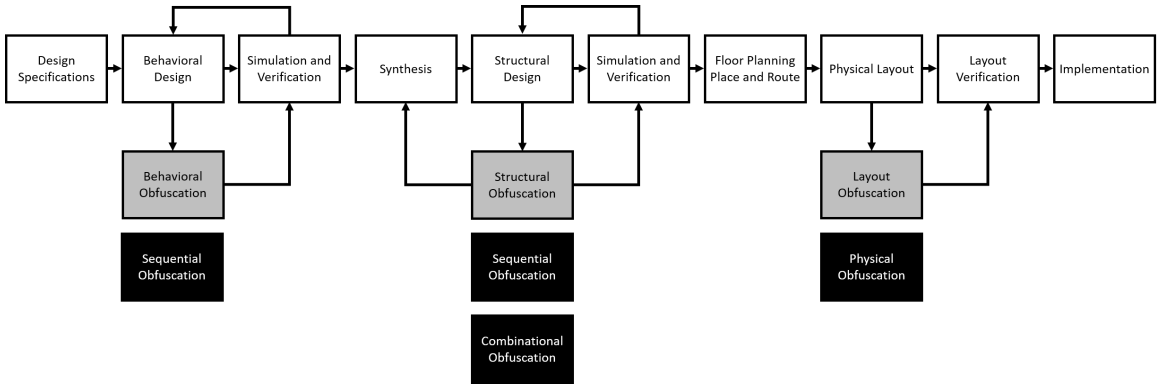


Figure 1.3: Logic locking description methods where IC design steps are shown in white and logic locking types based on their abstraction level of implementation (gray) and the type of modification LL is executing (black) [5].

1.3.1 Preprogrammed Known Gate

Known gate LL inserts a preprogramed logic cell at various places in a design to obfuscate its function. Additionally, known gate LL does not conceal functionality from the foundry, but in fact relies on the fabrication process to make the obfuscated gates work correctly. Furthermore, a plethora of known gate obfuscation types have been proposed and one such example from [16], shown in Fig. 1.4, uses High Voltage Threshold (HVT) and Low Voltage Threshold (LVT) transistors, to make identical

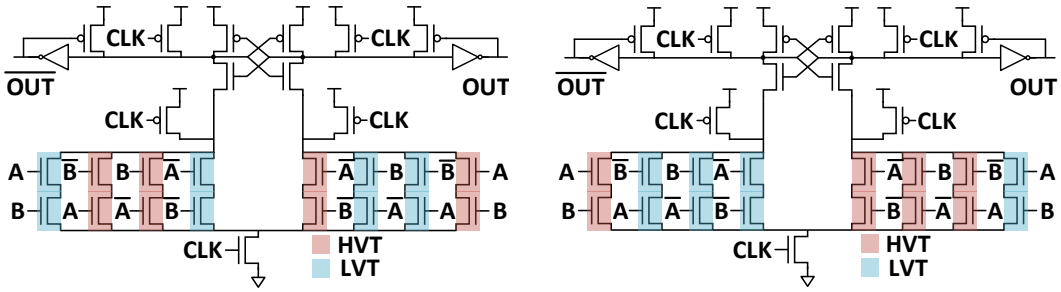


Figure 1.4: Threshold voltage defined logic family: (a) OR function configuration; (b) AND function configuration. [16]

looking gates perform different functions. These particular gates also have the added benefit of being side channel resistant as all functions use the same amount of power.

The commonality of all known gate LL types is that the obfuscated elements do not resemble a typical standard cell. Consequently, during the reverse engineering process, any gate that has been obfuscated is easily recognized and obvious to an attacker. With this in mind, the quality of an obfuscated gate is usually measured on how many different functions it can perform, its performance/overhead, or its side channel resistance. To provide side channel resistance for example, all functions of the gate must use the same power or have the same delay. Known gate of obfuscation also has the added benefit of not requiring additional inputs or a separate chip to be programmed.

1.3.2 Preprogrammed Unknown Gate

Unknown gate LL modifies the entire standard cell library to be optimized for LL and as a result makes all gates appear identical or sets of gates appear identical. Fig. 1.5 shows a regular standard cell NAND and NOR gate which are visually

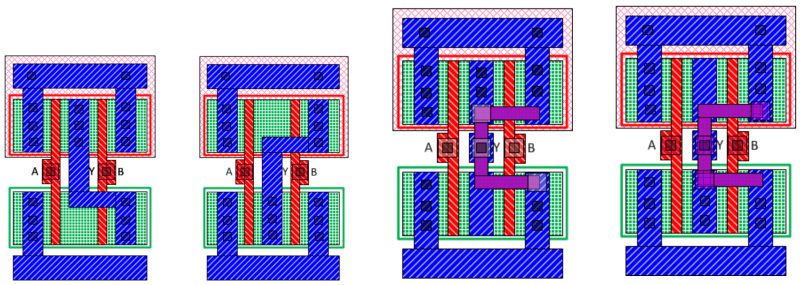
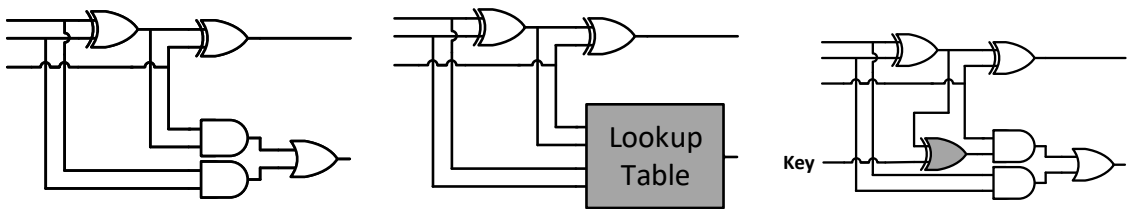


Figure 1.5: Standard cell layout of (a) NAND and (b) NOR and camouflaged gates (c) NAND and (d) NOR [33]

different, and a camouflaged standard cell NAND and NOR which are almost visually identical. Furthermore, through the use of dummy vias, the true function of the gates can be masked from an attacker. Moreover, unknown gate LL, by reason of the standard cell library consisting of only camouflaged gates, results in all gates being camouflaged. From a security standpoint this vastly increases the difficulty of deobfuscating a design, however this comes at a price. The large overhead of the obfuscated elements, compared to a typical standard cell, results in severe penalties to size, power, and performance of the chip. Therefore, most unknown gate obfuscation schemes recommend a 5-15% obfuscation amount, to reduce these penalties. As evident from the figure, the camouflaged gate variants are significantly larger than their standard cell counterpart. Also, due to the additional wiring and size, the camouflaged gates are slower and require more power to compensate. In order to mitigate some of these penalties, unknown gate obfuscation typically performs a very limited subset of functions. As a result, there is more obfuscation but the gates themselves are less complex.



(a) (b) (b)

Figure 1.6: LL examples: (a) Unobfuscated circuit; (b) LUT Obfuscation; (c) XOR Obfuscation.

1.3.3 Key Based

Key Based LL is programmed after chip fabrication. Through the use of configurable gates, the chip is locked unless the correct key is supplied, where an incorrect key or no key will result in a non-functioning chip. Therefore, the resultant chip is protected from reverse engineering and overproduction from the foundry as the key is not needed for fabrication. Additionally, various methods have been proposed, with two examples shown in Fig. 1.6. In [22], circuit functionality is hidden from the foundry by replacing some gates with programmable look-up tables (LUTs), similar to structures found in FPGA architectures, which must be configured prior to correct chip operation, as shown in Fig. 1.6(b). In this research, we focus our analysis on secret-key XOR gate LL, shown in Fig. 1.6(c), as it is the predominately used architecture for obfuscation [7][33][20], and has the most easily available benchmarks. XOR-based locking works by inserting key-controlled XOR and XNOR gates, with and without an inversion stage, into various nets in a given design, as illustrated in Fig. 1.7. Accordingly, a wrong key inverts the bit on the corresponding net, resulting in a faulty operation of the chip.

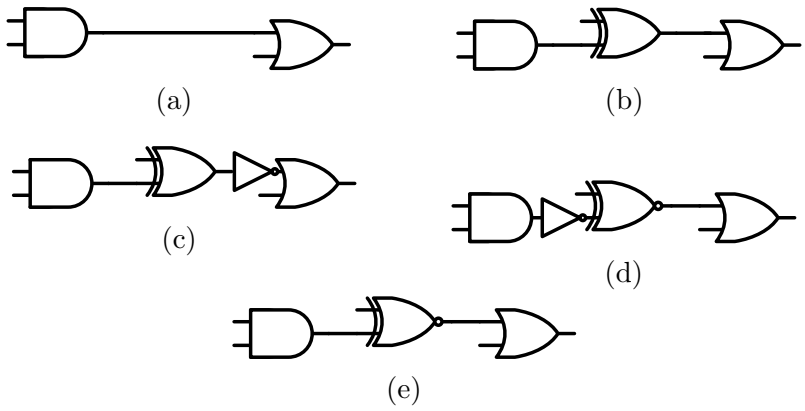


Figure 1.7: XOR based logic locking examples: (a) Unlocked; (b) XOR with key = '0'; (c) XOR with key = '1'; (d) XNOR with key = '0'; (e) XNOR with key = '1'.

1.4 Logic Deobfuscation

Logic locking, in theory, renders IP unusable, and increases the difficulty and risk of inserting a trigger based HT. However, the effectiveness of any LL technique depends on the design chosen, the relative amount of camouflage inserted, key length, the specific location of the inserted cells, and the metric used to measure success. Currently, the most common metric, based on functional attacks, is the measure of time required to break obfuscation. Additionally, regardless of the implemented technique, the evaluation procedure is to apply a deobfuscation method to an obfuscated sample circuit, such as the ISCAS'85 [18] and ISCAS'89 [10] benchmarks.

1.4.1 Functional Based Attacks

Functional attacks are by and large the most researched types of attacks; hence, consequently, they have produced the most influence on LL techniques. In the original model for functional attacks, an adversary is assumed to have access to an obfuscated

netlist, a fully functioning chip, referred to as an oracle, and the knowledge of which gates have been camouflaged [33]. To reveal the functionality of the obfuscated elements in the netlist, an attacker applies test vectors to the inputs of the oracle and observes its outputs. The locked elements are then configured to match the obfuscated netlist's function to the oracle's, based on these observed outputs. Early on, no intelligence was used to select the input vectors, therefore, the effort for this brute force attack was the chosen metric for security [33]. Later on, satisfiability (SAT) Solvers were leveraged to determine the ideal input vectors that efficiently decrypt the camouflaged gates in a minimal amount of time [15]. In the following years, various SAT solvers and SAT protection methods have been proposed, such as an incremental SAT solver in [25] and a partial SAT solver in [41]. Other proposed functional attacks, such as key sensitization attacks, use automatic test pattern generation to propagate the key value to the output, much like fault detection [46].

SAT solvers work by generating Differentiating Input Patterns (DIPs) that eliminate the highest number of possible keys or gate types and then test these patterns on the oracle. An example if this is shown in Table 1.1. From the table there are four locked XOR based key-gates. Additionally, the key-gates interfere with one another and must be solved at the same time. Therefore, there are 2^4 possible key value combinations. A SAT solver determines that the first input vector to test should be '0000', which will result in the elimination of eight key values if the oracle outputs a '1', and the other eight key values are eliminated if the oracle outputs a '0'. As shown in the Table 1.1, the oracle outputs a '0' and the key values with an incorrect output are labeled with an 'X'. With eight possible keys left, the SAT solver then determines the next best input vector is '0001' which would eliminate half of the remaining keys

Table 1.1: Differentiating input patterns (DIP) on two obfuscated gates.

Key Combination	Differentiating Input Pattern (DIP)			
	0000	0001	0100	0101
Oracle	0	0	1	1
0000	1 (X)			
0001	1 (X)			
0010	1 (X)			
0011	1 (X)			
0100	0	0	0 (X)	
0101	0	0	1	1
0110	1 (X)			
0111	0	1 (X)		
1000	0	1 (X)		
1001	1 (X)			
1010	0	0	1	0 (X)
1011	1 (X)			
1100	0	1 (X)		
1101	0	0	0 (X)	
1110	0	1 (X)		
1111	1 (X)			

regardless of the oracle output. This process continues until there is only one possible key combination left. Compared to a brute force approach the oracle would require up to sixteen inputs patterns to determine the key value. The SAT solver, however, is able to reduce the number of input patterns down to four.

In recent years, various SAT solvers and SAT protection methods have been proposed, such as an incremental SAT solver in [25], where after each iteration the Conjunctive Normal Form (CNF) formula becomes more restrictive to reduce solving time. In [41], a partial SAT solver was used to deobfuscate portions of a design, reducing the encryption effectiveness. Other proposed functional attacks, such as key sensitization attacks, use automatic test pattern generation to propagate the key

value to the output, much like fault detection [46]. To protect from SAT attacks, a specialized anti-sat block was proposed in [45] to increase the CNF formula complexity beyond the capabilities of a SAT solver. This is an evolution of the AES core proposed in [46], which connects a fixed key AES between the key inputs and the key-gates, essentially removing the correlation of key input to key value on the chip thus preventing a SAT solver from generating DIPs.

1.4.2 Structural Based Attacks

While much of the focus has been on functional testing, a new class of attacks have emerged to analyze LL without the need for an oracle. These attacks are innately structural, exploiting the underlying formation of a circuit and the deterministic nature of the synthesis process. One such example is the Structural Analysis using machIne Learning (SAIL) attack, which employs a machine learning algorithm to recover the original, pre-synthesized structure immediately around the key-gate [11]. The original structure can then be used to determine the inserted change and uncover the key. Another attack, referred to as a desynth attack, extracts a sub circuit around key-gates and re-synthesizes them with all the possible key combinations [29]. The structure that is least changed after re-synthesis is identified as the most probable key.

1.5 Evaluation Method

In most cases, obfuscation methods are evaluated based on the time to break them using a functional test. However, evaluation results vary considerably as the methods and equipment used directly affect the solution speed. For example, a technique may be deemed unbreakable based on one decryption time estimate, only to later be

disproved by using a different deobfuscation method. The metric can also be misleading, as the architecture of the test circuit can be naturally SAT resistant, aiding an otherwise breakable LL technique. Consequently, the lack of an accurate metric makes it hard for a defender to know when sufficient obfuscation has been applied. Additionally, most current metrics assume access to an oracle, which is not always a valid hypothesis. Foundries, for instance, would only have access to an obfuscated netlist, rendering functional attacks unfeasible. However, removing the precondition of an oracle does not guarantee protection against deofuscation. More specifically, the attacker can extract information from the underlying formation of a design, given the frequent repetition, design reuse, and inclusion of third-party IP (3PIP). Unfortunately, conventional models for functional attacks, do not consider these threat vectors. Alternatively, other attack methods, such as SAIL and Desynth, only analyze the localized structure and forgo global circuit information. While a successful attack can be launched on the localized structure, slight circuit modifications at the key-gates will have a large impact on the technique's ability to break LL. Additionally, localized attacks do not consider design reuse and repetition, leading to another deficiency in their attack methodology. Contrarily, global structural features should be less effected by small circuit changes, while also being able to exploit repetition. Therefore, the global structure creates a new attack surface, exposing LL to additional vulnerabilities.

1.6 Holistic Approach to Logic Locking

The intention of this work is to develop a metric in which to asses the security level of logic locking operations. In order to completely capture the entire vulnerability

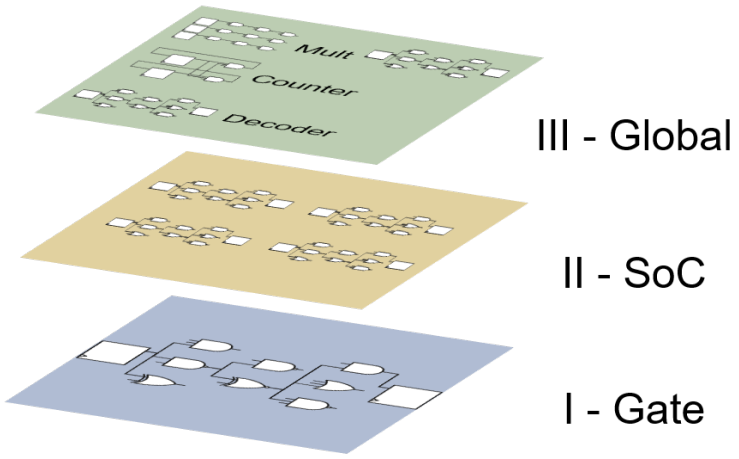


Figure 1.8: Design abstraction levels.

space, a holistic approach must be used that encompasses the multiple abstraction levels of a circuit design, which are shown in Fig. 1.8. Generally speaking, each level opens a new vulnerability to a logic locked circuit, since each level contains information about the design which can be exploited to solve the locking.

Level I: Gate

Abstraction level I is the gate level approach to logic locking security. This is the lowest level of abstraction and observes a logic locked circuit as a sea of gates. At this level, SAT solvers operate by attempting to analyze the entire circuit as an equation ignoring the larger picture. Additionally, low level gate repetition and the structural attacks, SAIL and Desynth, also operate at this level. Consequently current metrics for security are also based purely on this layer of abstraction. However, these metrics are still limited in scope and largely ignore vulnerabilities at higher levels.

Level II: SoC

Abstraction level II considers the logic locked circuit, the entire SoC it resides in, and the many modules it is made up of. At the gate level, a locked circuit may be safe as it is resilient to SAT attacks and there isn't enough information otherwise to solve it. While an individual module in a design maybe resilient to SAT attacks, such as the substitution box (SBOX) from an AES encryption core, however, these types of modules are not unique in the overall design. Moreover, there could be tens, or in many cases hundreds, of the same instances that appear in the overall circuit. During deobfuscation, if all instances of the same module are correlated then the attack space is reduced significantly. Additionally, to obfuscate these modules, the same elements in all instances must be obfuscated identically. Otherwise, if an attacker can correlate obfuscated instances as the same type of module then they are able to deobfuscate them by correlating the locked and unlocked sections. However, even locking these modules identically still results in reduced security as now only one of the modules must be unlocked. Furthermore, abstraction level II also incorporates scan chain access and overall access to the underlying design.

Level III: Global

Abstraction level III goes beyond a single SoC and looks to establish the vulnerabilities associated with information gained elsewhere. One area of attack is based on the use of common design elements and IP in circuits, as well as IP reuse. Additionally a common library of components could be created and subsequently used to analyze a locked SoC, to help identify function. Moreover, other information such as

who constructed the design and what tools were used is additional information that an attacker can utilize to gain an advantage.

1.7 Summary

Considering the vast amount of information that LL leaves behind and the knowledge an attacker can gain from multiple levels of abstraction, this work attempts to develop a metric in which to asses the security of LL. This is accomplished through three major contributions. In Chapter 2, we detail a new structural attack method that exploits the repetition of an IC and utilizes the information to predict key bits. In Chapter 3, we describe a systematic method to describe a circuit by its structure which enables the comparison of circuits through structure to circumvent the effects of LL. Additionally, we combine the attacks into a security framework in Chapter 4, that is agnostic of attack types and can be used to combine the effects of multiple attacks to assess the probability of an adversary to break a locking technique. Lastly, we summarize the work and address future research in Chapter 5.

Chapter 2: GRAPPLLE

2.1 Core Concept of SAT Attacks and SAT Defenses

The primary attack methodology for any LL technique has been SAT based. While there is a plethora of anti-SAT algorithms, procedures, and key placement arrangements, they all focus on manipulating the controllability and observability of the circuit. Considering that the core requirement to SAT attacks is the capacity to elicit a change through the use of primary inputs (control) and then be able to monitor that change on a primary output (observe), the concept of anti-Sat is then finding ways to restrict the controllability or observability of these changes.

The original methodology for logic locking did not consider this concept and produced locked circuits similar to Fig. 2.1a. Consequently, the key gates are isolated, which gives an attacker nearly direct access to control the inserted key-gates and observe their outputs. Considering the original protection model assumed brute force was the only viable attack, these circuits were deemed safe as long as there was a sufficient key length and output corruption, regardless of how the obfuscated gates were connected. In an attempt to increase the security of logic locking, an interference scheme was created, such as in Fig. 2.1b, [33]. This concept required solving several key-gates at the same time, as one key output would interfere with another.

Therefore, in essence, the researches reduced the controllability and observability into the circuit, which in turn increased the difficulty to break the function locking key. However, the introduction of SAT solvers increased the efficiency to control the keys and in turn reduced the number of input vectors needed to solve them. The cat and mouse game that followed introduced new ways to decrease the controllability or observability into the system while attackers built better SAT solvers to overcome these issues [25], [33], [37], [31], [46], and [45].

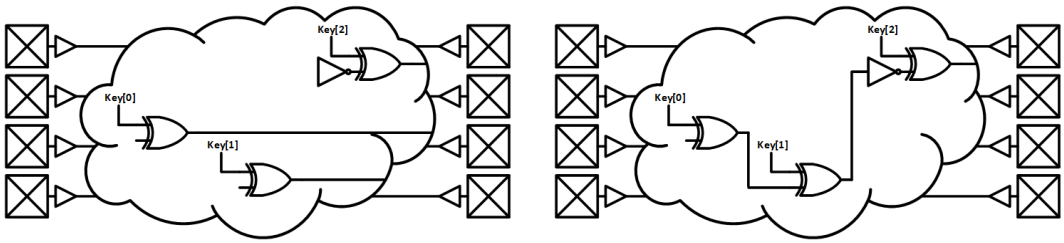


Figure 2.1: Generic implementation of logic locking; (a) LL implemented with isolated keys allowing functional testing to solve one key at a time for easy breaking: (b) LL implemented with keys interfering with one another. This forces a functional test to solve all the keys as one block which increases the time to break.

2.2 Benchmark effects on controllability and observability.

The most common set of test articles for combinational logic locking consists of the ISCAS'85 benchmark circuits. These test articles are relatively small, ranging from a few hundred to a few thousand gates, and contain no flip flops. Therefore, the logic depth from input to output is short, making it easier for attackers to break through functional testing. Additionally, the small size of the test articles enables SAT solvers to be more effective and doesn't require the circuits to be broken into

smaller pieces. However, when the community began looking at larger test articles, it was apparent that the ability to observe outputs was severally restricted unless scan chain access was possible. Consequently, this led researchers down the path of scan chain obfuscation, restricting its access, or essentially locking the scan chain, and in turn, forcing attackers to again only utilize the primary inputs and outputs of a chip. As a result, this greatly increased the difficulty of breaking a function lock as the sheer amount of logic depth required is difficult for SAT solvers to analyze. While there has been some research on ways to break function locking without access to the scan chain, such as in [28], in general, the larger the IC, the more difficult it is for SAT attacks and functional testing to succeed. Additionally, ICs are limited by the number of inputs/outputs (IO) pads and therefore limited in the control and observe points. Given this, new non-functional attack methods may prove to be more fruitful.

2.3 Increased Observability

While SAT based attacks have proven useful, they have not been able to capture and exploit the sheer amount of repetition, design reuse, and 3PIP used in today's modern SoCs. While SAT attacks require the obfuscated netlist, they largely ignore the information it contains and only use it to decide how to generate input vectors to eliminate possible key values. However, there is quite a bit of information which can be gathered through the observation of the netlist. Certain features can reveal clues and indicate how the circuit is operating to help predict the key values. Nevertheless, even with all the information readily available in a netlist, for SAT based attacks, a random netlist is treated exactly the same as a highly repetitive one. It is important to note that the synthesis process, which generates the netlist, is deterministic. This

deterministic nature and the repetition and predictability that ensue is the downfall of many function locking techniques. Since these features can be observed directly at the netlist, and not only at an output, the amount of observable points is increased by orders of magnitude as each gate is an observable entity. However, while the structural and repetitive information is present, it does not mean it is easy to exploit and will requires methods to analyze it.

2.3.1 Basic Example of Repetition

To demonstrate how to exploit design repetition, a simple example is shown in Fig. 2.2. The figure shows a 2-bit adder comprised of two Full Adder (FA) circuits and the design is locked with a 1-bit key on the least significant bit. While this is an easy lock to solve using a SAT solver, it would still require an oracle and one input vector to solve the key. Therefore, without an oracle, the SAT solver can not actually determine the correct key value as there would be no way for it to test a DIP. However, through visual analysis of the circuit, there is an obvious repetition due to the two identical sub-circuits, except for the inserted key-gate. Hence, the lack of a key on the second FA circuit reveals the answer to the first. While this example may seem simple, repetition is extremely common when using a synthesis process and any arithmetic, arrays, and generate statements all produce very localized repetition that can be exploited.

Consequently, the threat model for SAT assumes the only control points are the circuit inputs and the only observable points are the outputs. However, through localized repetition observation, the observability into the system is increased. Therefore, current protection methods that generate interference between either the circuit input

and the key-gate or between the key-gate and the output are insufficient. Moreover, interference methods must not only obstruct the IO but also the sub-circuit surrounding a key-gate. Additionally, a method in which to generate and analyze the repetition must be created.

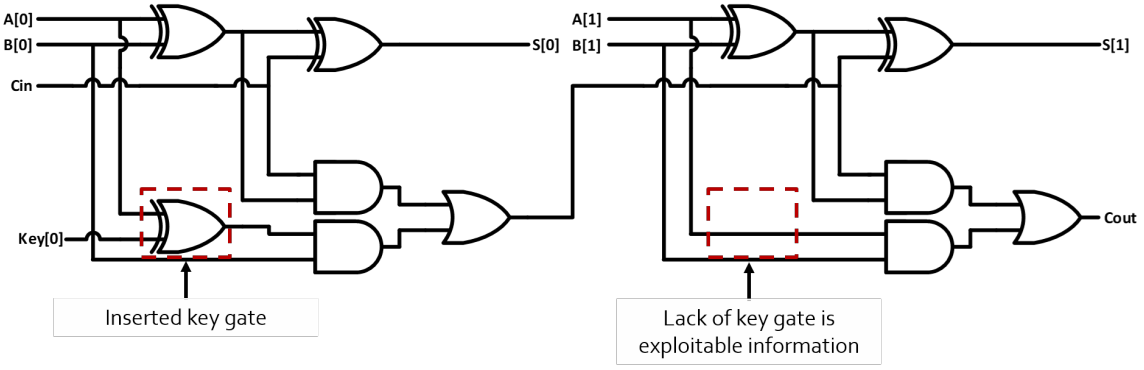


Figure 2.2: 2-bit adder with key-gate on first full adder but not on second. The lack of a key-gate on the second full adder indicates the key for the first.

2.3.2 Ideal Sub-Circuit Extraction and Key Value Prediction

As a perfect example, a sub-circuit could be extracted for every key-gate, where the sub-circuit refers to the key-gate and the gates attached to it within a predefined radius. Additionally, every non key-gate would also generate its own sub-circuit with no overlap between the non-key and key-gate sub-circuits. Then, from the key-gate sub-circuits, all the possible permutations of smaller sub-circuits would be generated and compared to every possible sub-circuit in the remaining design. Lastly, locating the best and largest match through exhaustive simulations would reveal a sub-circuit without an obfuscated element to be used as a reference to solve a key bit.

As an illustration, a locked circuit contains the extracted sub-circuit surrounding key-gate K0, as shown in Fig. 2.3a. Since an XOR based obfuscation technique was used, it is known that the key is either a '0', indicating no other modifications were made, or a '1' indicating an inverter was placed just prior to, or just after the key-gate. The sub-circuit can be further simplified by removing the key-gate and combining nets $N0$ and $N1$ into net N as shown in Fig. 2.3b. Net N is now the anchor point in which the key-gate effects the rest of the circuit and if the key is '1', net N becomes $\overline{N}$.

To describe the sub-circuit, it is divided into two pieces where the first piece has N as the final output, while the second has N as an input, such that N is represented by (2.1). Assuming the entire remaining circuit is analyzed, the most similar sub-circuit found is represented by Fig. 2.3c and described in (2.2). While the output equation of N is the same between the two, there is an inversion on the input equation to net N. This indicates that an inverter was placed near the key-gate and the key bit is '1'.

$$\begin{aligned} N &= (A * B) + (C * D) \\ Out &= N * E \end{aligned} \tag{2.1}$$

$$\begin{aligned} N &= \overline{(A * B) + (C * D)} \\ Out &= N * E \end{aligned} \tag{2.2}$$

2.3.3 Reality of Circuit Comparison

While the two examples of repetition are clearly defined and matched, the reality of extracting perfectly sized sub-circuits and locating a perfect match is not realistic. The variability of sizes and connections renders the matching process difficult. Additionally, there is no method to determine which nodes correspond between sub-circuits

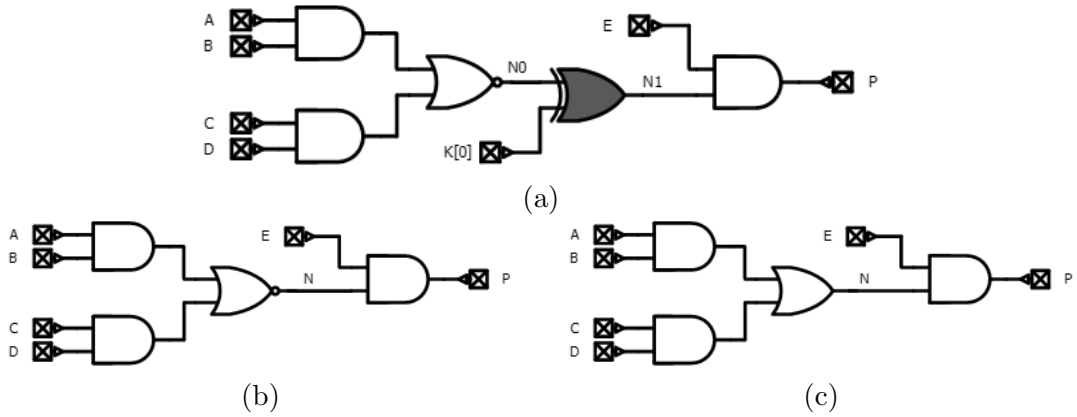


Figure 2.3: Extracted sub-circuits: (a) Sub-circuit of key K[0] with key-gate shown in gray; (b) K[0] with key-gate removed; (c) closest matching sub-circuit.

requiring permutations of the node correspondence to fully evaluate the similarity. For example, in the previous circuits the inputs were labeled identically but input A could have been switched with C. Consequently, the node correspondence between sub-circuits is not clear and would require every permutation of input ordering to truly determine the similarity. To demonstrate the challenge this inflicts, another sample sub-circuit of only the input driving gates is shown in Fig. 2.4. It should be noted the the output net P would be attached to a key-gate. In the previous example the binary equations were used to compare the sub-circuits, but the most straightforward way to make this comparison would be to perform exhaustive simulation on both sub-circuits and compare the results. For Fig. 2.4 this would require 256 input vectors. However, this assumes it is known which inputs correspond with each other. Conversely, the A inputs could correspond to the C inputs or to the D inputs and so on. Therefore, every permutation would have to be computed and then simulated in order to determine the similarity between the two sub-circuits. Given that $A[0]$ and

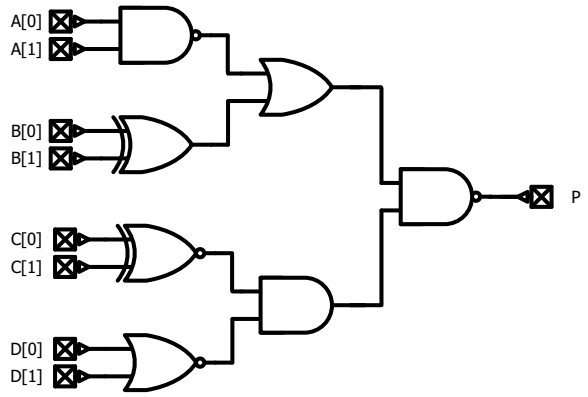


Figure 2.4: Three level depth sub-circuit

$A[1]$ are symmetrical, this ordering does not matter, hence there are 24 possible permutations of input order. Under these circumstances, an exhaustive simulation now requires 6144 input vectors to cover all possible permutations of the inputs, and this only includes the input driving net. Additionally, the complexity also grows quickly when considering the number of comparisons checks that must be performed. As illustrated in Fig. 2.5, if a logic locked circuit has a 128-bit key, each key is checked against 1000 other nets, and the output circuit has the same complexity as the input (6144 input vectors) then to evaluate the locking method, about 1.5 billion comparisons must be performed. Granted, this is a small circuit, so any moderately complex design would require orders of magnitude more comparisons. Therefore, a method must be developed to compute the similarity of a sub-circuit with no permutations and less comparisons.

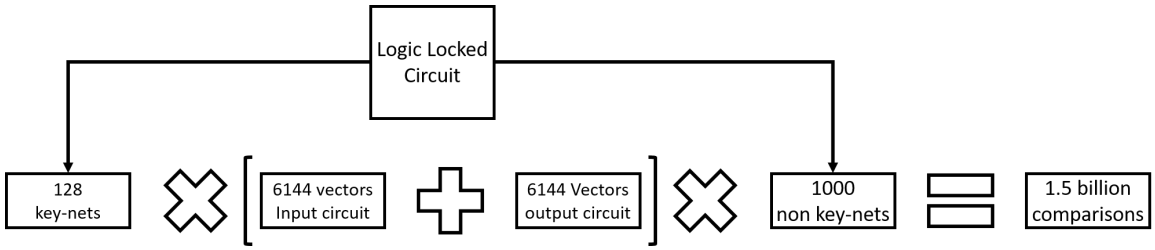


Figure 2.5: Small logic locked circuit with 128-bit keys and the required number of comparisons needed to analyze through simulations.

2.4 Sub-Circuit similarity through graphical representation.

Determining the similarity of two sub-circuits through an exhaustive set of input vectors and permutations is not feasible for any mildly complex circuit. Hence, a more efficient method in which to determine the similarity must be explored. For that reason, we devised GRAPh Probability for Logic Locking Evaluation (GRAPPLLE). Returning to the full adder example, is was apparent through visual inspection that the full adders were the same circuit. However, manual visual inspection of every sub-circuit is not feasible. With this in mind, if the sub-circuit around the key-gate is generated from a predefined pattern and all other sub-circuits use the same pattern, then the matching process is simplified as the sub-circuits are the same shape. To this end, GRAPPLLE generates a sub-circuit consisting of any gate within two hops around each node or gate in the circuit, as illustrated in Fig. 2.6, which shows G1 and its sub-circuit. Then, to determine a key value, the gate type of the node driving a key-gate is predicted based on its sub-circuit and the gate type of the node with the closest matching sub-circuit.

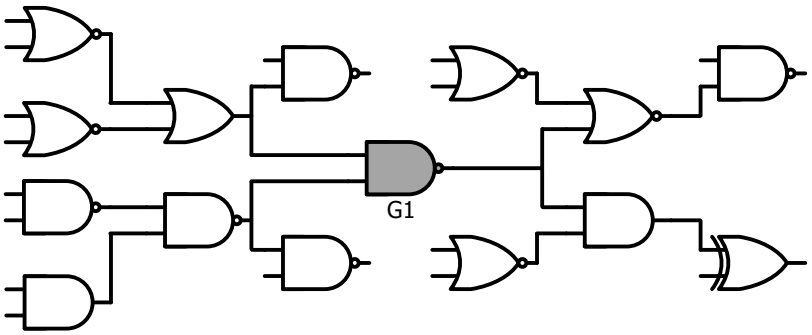


Figure 2.6: Gate G1 (shown in gray) and the gates in G1's sub-circuit (shown in white)

As a simple example, an AND gate G2 is driving key $K0$, as shown in Fig. 2.7, and in the same circuit resides gate G1 from Fig. 2.6. Considering the sub-circuits between G1 and G2 are identical, it stands to reason that G1's gate type should be the same as G2's. However, G1 is a NAND while G2 is an AND. Therefore the key can be guessed with some probability to be a '1', signifying the XOR key-gate is inverting that net. Additionally, the probability of this prediction is then directly related to the similarity of the sub-circuits which the match is based upon. In light of the exact sub-circuit match between G1 and G2, the prediction for G2 is highly probable.

2.5 Sub-Circuit Comparison

The predefined two hop sub-circuits are used as signatures which describe each node in a circuit. These signatures are further broken down into sub-signatures consisting of gate sequences based on the directions of travel used to generate the sequence, where the four sub-signatures are Forward Forward (FF), Forward Reverse (FR), Reverse Reverse (RR), and Reverse Forward (RF). Furthermore, the sequences

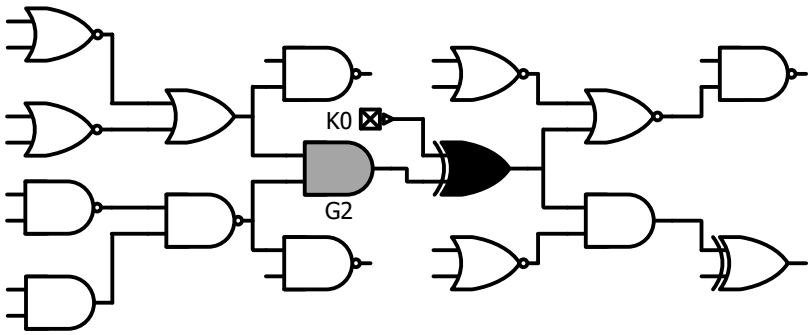


Figure 2.7: Gate G2 (shown in gray), the gates in G2's sub-circuit (shown in white) and the key-gate (shown in black).

which are generated for each sub-signature consist of all the possible gate paths that follow the sub-signature's direction. For example, the FF sub-signature contains all the possible gate sequences that are two forward hops from the source gate and the RF sub-signature contains all the possible sequences that are one reverse hop and one forward hop. The sequences are recorded as in (2.3)

$$Seq = (Gate_1, Gate_2, x) \tag{2.3}$$

where Seq is a sequence from gate $G1$, $Gate_1$ corresponds to a gate in the first logical depth from $G1$, $Gate_2$ corresponds to a gate attached to $Gate_1$ and is two hops from $G1$, and x is the number of times this exact sequence has occurred in this sub-signature. For the first occurrence $x = 0$, the second occurrence $x = 1$, and so on. The data is stored in this manner to allow for recording the type and frequency of each sequence, while also preventing duplicate sequences to exist in a given sub-signature. As a result of removing duplicate entries, the signatures are stored as hashed data types which greatly speeds up comparison times. Conversely, storing the data in a list or array would require checking a sequence against every element

in a list to verify it exists, while checking a hash tells the program exactly where to look for a particular element. To demonstrate this process, Table 2.1 shows all the sequences in the signatures from Fig. 2.6.

Table 2.1: GRAPPLLE sub-signature for gate G1 from Fig. 2.6

Sub-Signature Type	**Sequences**
Forward Forward (FF)	NOR2, NAND2, 0 AND2, XOR2, 0
Forward Revers (RE)	NOR2, NOR2, 0 AND2, NOR2, 0
Reverse Reverse (RE)	OR2, NOR2, 0 OR2, NOR2, 1 NAND2, NAND2, 0 NAND2, AND2, 0
Reverse Forward (RF)	NOR2, NAND2, 0 NAND2, NAND2, 0

2.5.1 Sub-Signature Comparison

Considering that the signatures are broken up into four sub-signatures (i.e. FF, FR, RR, and RF), these sub-signatures can be compared separately and then combined to asses the overall similarity of two sub-circuits. Hence, to compare the signatures of gate $G1$ and $G2$, first the FF sub-signature from gate $G1$ is compared to the FF sub-signature of gate $G2$, then the FR sub-signatures are compared and so on and so forth. Additionally, considering the sub-signatures are comprised of unique sequence elements, the more matching sequences between two sub-signatures indicates the more similar they are. However, instead of using a direct ratio of matching sequences to total sequences, the Jaccard index [21] is proposed, as in (2.4)

$$J_D(i,j) = \frac{|A \cap B|}{|A \cup B|} \tag{2.4}$$

where A is a sub-signature from node i, B is the same sub-signature type from node j, and D is the direction of the sub-signature such as FF, FR etc. The Jaccard index is used in order to take size disparities into account. Thus, this prevents very large sub-signatures from automatically matching perfectly with small sub-signatures due to the sheer amount of sequences they contains. Finally, to asses the overall similarity between node i and j (2.5) is used

$$MS(i,j) = J_{FF} * J_{FR} * J_{RR} * J_{RF} \tag{2.5}$$

where $MS(i,j)$ is the match score which is between 0 and 1; 0 indicating the nodes are not at all similar and 1 indicates the nodes are identical.

2.6 Netlist Creation

Only assessing the similarity between sub-circuits of two nodes does not predict key values. Therefore, the next step in the process is to determine key values by utilizing the sub-circuits around the key-gates, and comparing them to other sub-circuits in the same design. Hence, after generating the signatures for every node, the nodes are then binned into two categories. The first category is nodes that directly drive key-gates which will be referred to as key-nodes. The second category consists of all the other nodes in the circuit and is referred to as match-nodes. The main goal of this process is to asses if the key-gate is inverting its input or allowing it to pass through unmodified. This is accomplished for each key bit by comparing its key-node's signature to every match-node's signature. The gate type of the match-node with the

highest match-score is used to predict the gate type of the key-node. However, the inversion process is more complicated than a single gate inversion, such as an AND becoming a NAND. Therefore, an inversion may alter the signature instead of the key-node type which hinders the matching process.

In general, to implement XOR-based LL, inverters must also be used to allow both key values. A '0' key indicates no additional inverter is inserted, while a '1' key indicates an inverter is placed just before or after the XOR key-gate. Fig. 2.8 shows the progression from the original circuit in Fig. 2.8a, to inserting an XOR gate and inverter in Fig. 2.8b. The presence of an inverter immediately preceding or following the key-gate would be an obvious indication of the correct key value. To hide the added inverter, the design is re-synthesized and the inverter or the inversion bubble is moved around depending on the design optimization, as shown in Fig. 2.8c-e. Note that the re-synthesis process could leave the inverter nearby, move it to another net, invert a nearby gate, or remove it entirely. Therefore, due to the different possible ways the inverter can move, the inversion may not be contained by the key-node but instead contained inside of the signature. Additionally, the inversion may even change the signature substantially, resulting in no similar sub-circuits. Moreover, circuits can be composed of complex gates such as AOI (AND-OR-INVERT), which creates another degree of complexity in the signature matching process. Therefore, in order to simplify gate types and to overcome the obstacles of netlist inversion, two new netlists are generated from the original design.

The proposed process to generate the new locked netlists starts with striping process design kit (PDK) and standard cell library gate information and converting the designs into a generic representation. Then, this generic netlist is re-synthesized

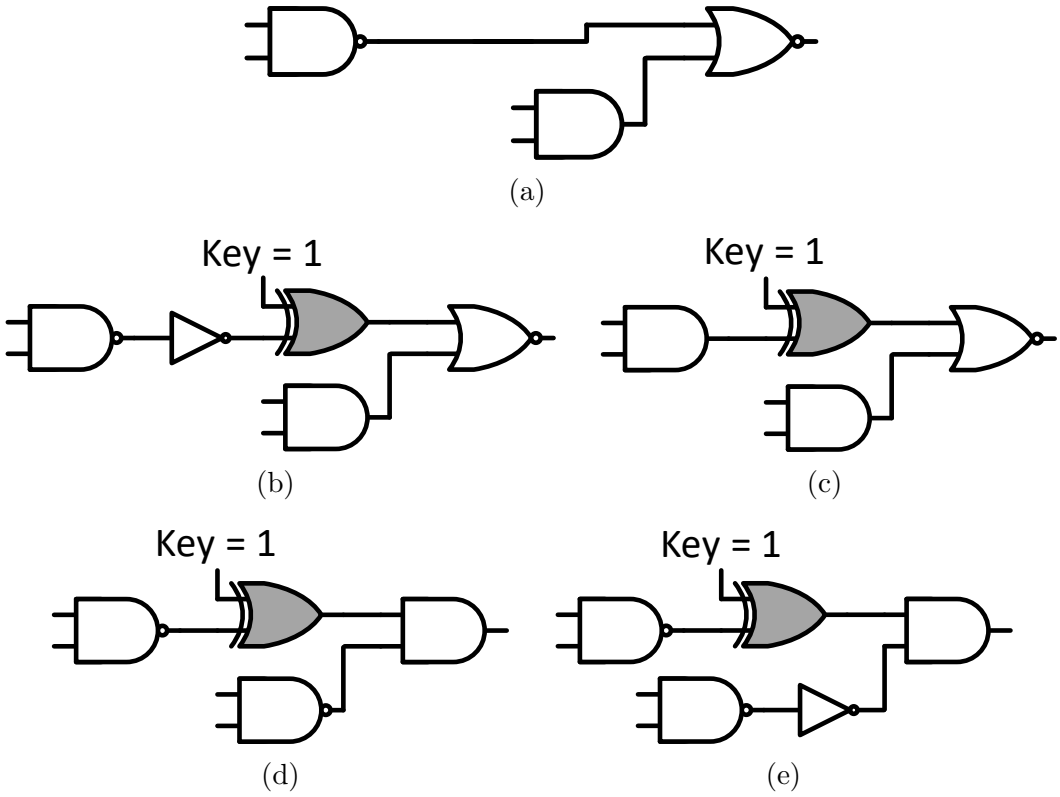


Figure 2.8: XOR gate insertion example, with original circuit (white), inserted key-gate (gray): (a) Original unlocked circuit; (b) XOR and inverter inserted with key = '1'; (c) After re-synthesis example 1; (d) After re-synthesis example 2; (e) After re-synthesis example 3.

into a simplified standard cell library that only contains, NAND, AND, OR, NOR, XOR, and XNOR gates, which creates the first new netlist. This netlist is referred to as the Non-Inverted (NI) netlist. This re-synthesis process overcomes the complex gate issues but does not overcome the inversion issue. Therefore, to solve the inversion issue, the generic netlist is again re-synthesized, but first an inverter is placed after every single key-gate. This creates the second new netlist which is referred to as the Inverted (IN) netlist. The purpose of inserting inverters and re-synthesizing the design is to remove the guess work in how the signatures are effected by inverters

from the key insertion process. If the key was a '1', the additional inverter will now switch the key back to a '0', essentially returning the signature to its original form.

2.7 Key Prediction Flow

The GRAPPLLE flow is shown in Fig. 2.9. First, the NI and IN netlists are synthesized. Then, signatures are generated for every match-node and key-node between the IN and NI netlists. Next, every key-node pair is compared to every match-node, where the key-node with the highest match score is the first bit to be solved. The key-gate type of the highest matching key-node and the netlist from which the key-node presides determines how the netlists are modified. For example, Fig. 2.10a and 2.10b show the K0 key-gate from the NI and IN netlists respectively. The sub-circuit from the NI netlist, shown in Fig. 2.10a, has the highest match score, the key-gate type is an XOR, and the predicted gate type and the actual key-node's gate type are AND gates. In this scenario, the key is determined to be a '0'. Therefore in the un-synthesized NI netlist, the XOR gate is removed and its input and output are tied together, as shown in Fig. 2.10c. Additionally, in the un-synthesised IN netlists, the extra inverter and the XOR gate are removed, shown in Fig. 2.10d. However, in some instances, the predicted gate type may not match the actual gate type of the key-node. For example the key-node may be an AND gate while the predicted gate type is a NAND gate, as illustrated in Fig. 2.11. In this situation, since the AND (Fig. 2.11a) is simply an inverted NAND (Fig. 2.11b) then the predicted key value is a '1' so the XOR gate performs the inverting function. To further illustrate this example, the original unsynthesized key-gate is shown in Fig. 2.11c. There are

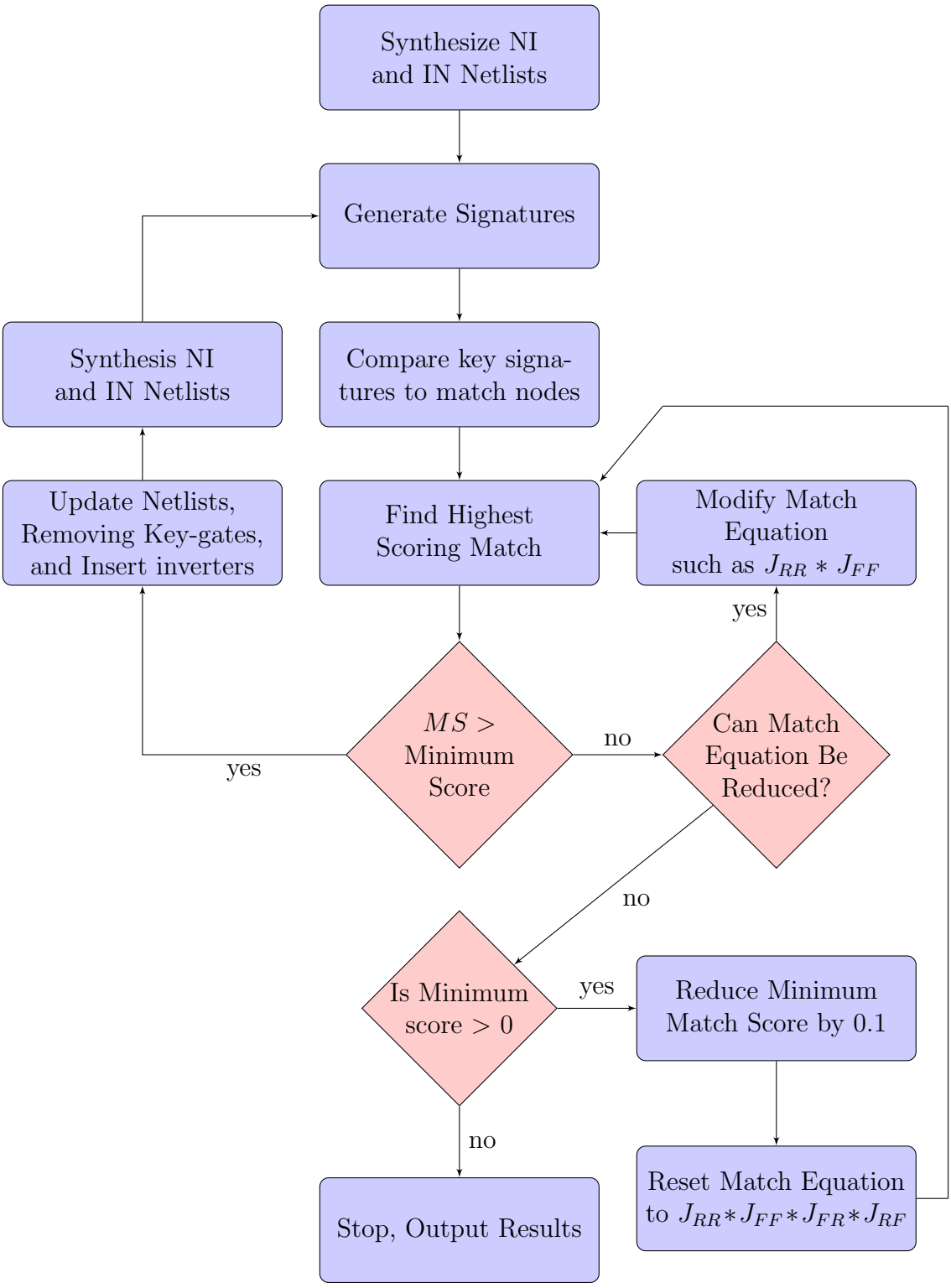


Figure 2.9: GRAPPLLE algorithm flow.

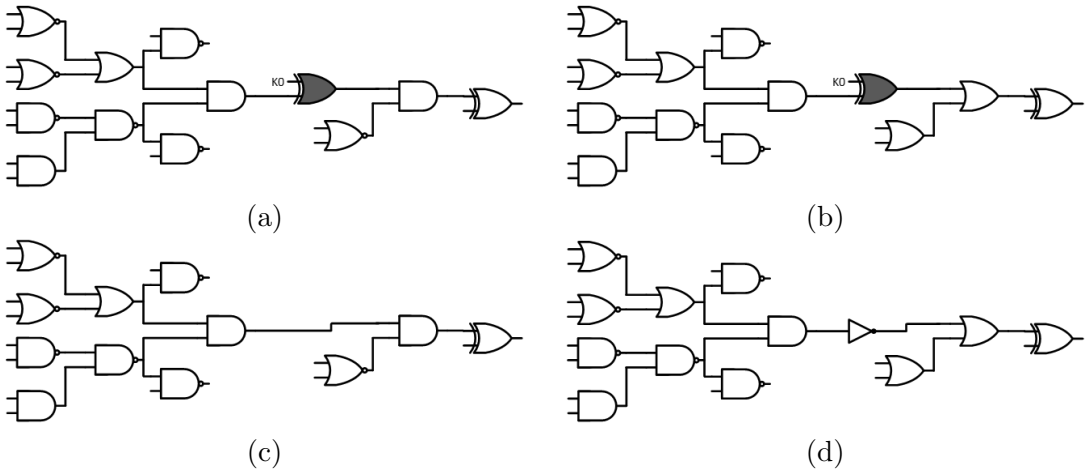


Figure 2.10: Match prediction example: (a) K0 sub-circuit from NI netlist with key-gate shown in gray; (b) K0 sub-circuit from IN netlist with with key-gate shown in gray; (c) NI netlist with K0 key-gate removed; (d) IN netlist with K0 removed and inverter added to fix the inversion.

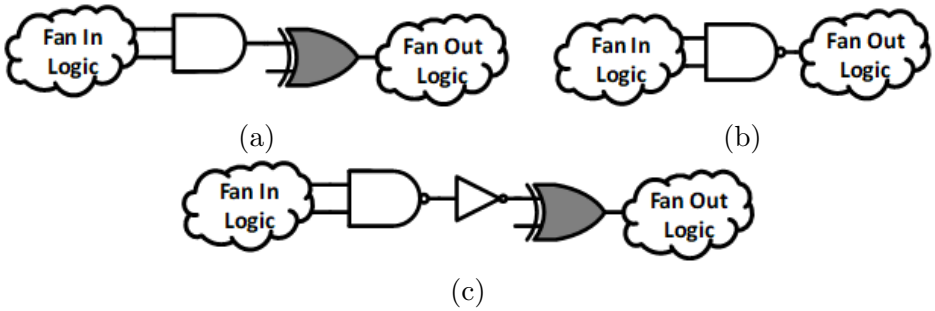


Figure 2.11: Example of gate inversion resulting in a predicted key value of '1': (a) Key-node (shown in white) and key-gate (shown in gray); (b) Highest match score sub-circuit with NAND type match-node; (c) Original unsynthesized obfuscated sub-circuit with inserted key-gate (shown in gray), the original NAND gate, and the inserted inverter.

36 combinations of key-node gate types and match-node gate types, and their corresponding key bit predictions can be found in Table 2.2. Once the key values are predicted, both the NI and IN netlists are updated. The update process involves

removing the key-gate and inserting the necessary inversion depending on the key value. To integrate the inversion and remove any effects the key-gate may have had on other key values, both the NI and IN netlists are re-synthesized and the prediction process starts again. However, if at some point there is not a high enough match score then the match score equation is modified to reduce the amount of required sub-signatures, as in (2.6), (2.7), (2.8), and (2.9).

$$MS(i,j) = J_{FF} * J_{RR} \tag{2.6}$$

$$MS(i,j) = J_{FR} * J_{RF} \tag{2.7}$$

$$MS(i,j) = J_{FF} * J_{RF} \tag{2.8}$$

$$MS(i,j) = J_{FR} * J_{RR} \tag{2.9}$$

If still no sufficient matches can be found at this stage, then the minimum score is reduced and the process starts all over. Additionally, if no more matches can be found, then no more key bits can be determined and the process is ended. It should be noted that large portions of this flow are manual. While the signature generation and matching portions are done through a series of python scripts, the process of deciding and subsequently updating the netlists for re-synthesis is done manually.

2.8 Equivalent Circuit Creation

As from the Table 2.2, there are obvious results such as a when a NAND gate is predicted to be an AND gate signifies an inversion has occurred and thus the key is '1'. However, the less obvious results occur with a NAND key-node but a predicted OR gate. In most situations, this is due to equivalent circuits being represented with

Table 2.2: Key Node Key Prediction

Key Node	Predicted Gate	Predicted Key			
		Netlist / Key Gate Type			
		NI/ XOR	NI/ XNOR	IN/ XOR	IN/ XNOR
NAND	NAND	0	1	1	0
	AND	1	0	0	1
	NOR	1	0	0	1
	OR	0	1	1	0
	XNOR	N/A	N/A	N/A	N/A
	XOR	N/A	N/A	N/A	N/A
AND	NAND	1	0	0	1
	AND	0	1	1	0
	NOR	0	1	1	0
	OR	1	0	0	1
	XNOR	N/A	N/A	N/A	N/A
	XOR	N/A	N/A	N/A	N/A
NOR	NAND	1	0	0	1
	AND	0	1	1	0
	NOR	0	1	1	0
	OR	1	0	0	1
	XNOR	N/A	N/A	N/A	N/A
	XOR	N/A	N/A	N/A	N/A
OR	NAND	0	1	1	0
	AND	1	0	0	1
	NOR	1	0	0	1
	OR	0	1	1	0
	XNOR	N/A	N/A	N/A	N/A
	XOR	N/A	N/A	N/A	N/A
XNOR	NAND	N/A	N/A	N/A	N/A
	AND	N/A	N/A	N/A	N/A
	NOR	N/A	N/A	N/A	N/A
	OR	N/A	N/A	N/A	N/A
	XNOR	0	1	1	0
	XOR	1	0	0	1
XOR	NAND	N/A	N/A	N/A	N/A
	AND	N/A	N/A	N/A	N/A
	NOR	N/A	N/A	N/A	N/A
	OR	N/A	N/A	N/A	N/A
	XNOR	1	0	0	1
	XOR	0	1	1	0

different gate types even though they produce the exact same function. An example of this is shown in Fig. 2.12, where 2.12a and 2.12b perform the exact same function but use different types of gates. Consequently, this reality not only causes problems with gate prediction but can also result in equivalent signatures not matching.

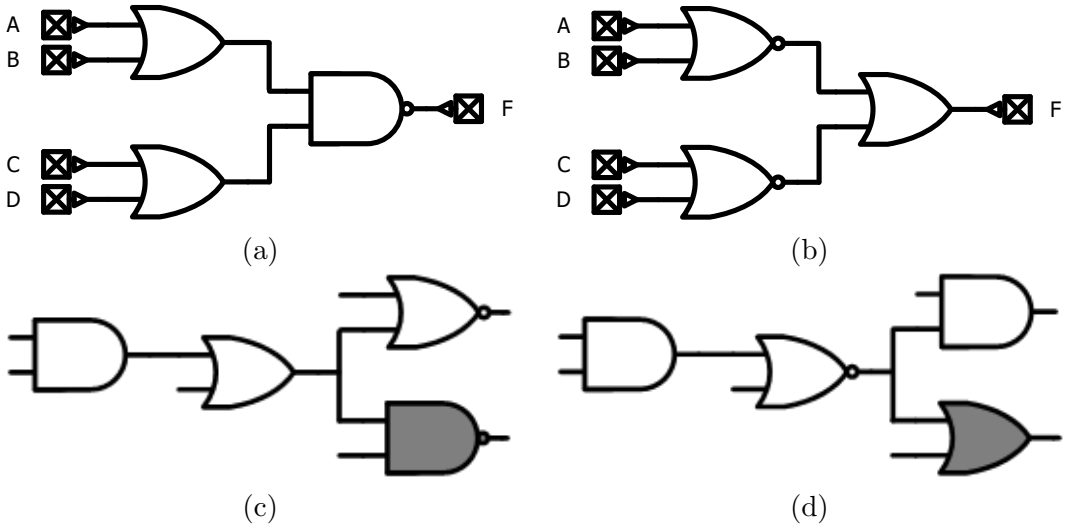


Figure 2.12: Equivalent circuits configured using different gate types: (a) Original circuit; (b) Applying DeMorgan's rule to the original circuit derives this functionally identical circuit; (c) Partial RR and RF sequences from the original circuit where starting gate is shown in gray; (d) Partial RR and RF sequences from DeMorgan circuit where the starting gate is shown in gray.

To account for this, all of the signatures go through an equivalence check and alteration process. The principal idea is to account for equivalent signatures due to De Morgan's law but without needing to analyze and invert the entire sub-circuit. Instead, each sequence from each sub-signature is processed separately resulting in a much simpler and straight forward conversion. Additionally, the process can also be pre-generated during the signature creation process to speed up matching. This

process takes each sequence and determines an equivalent sequence from a limited set. However, at no point is the source node changed, hence, an OR gate is equivalent to a NAND gate as in Table 2.2. As an example, Figs. 2.12c and 2.12d represents the partial RR and RF sequences of Figs. 2.12a 2.12b respectively. To account for the possibility of equivalent signatures the source node, marked in gray, is left unmodified while the sequences are changed based on the first reverse gate encountered. If the first reverse gate is an OR then the sequence are left unmodified, which is also shown in Table 2.3 for Fig. 2.12c. However, if the first gate in the reverse sequence is a NOR then to account for the equivalent signatures the NOR gate is converted to an OR, which causes the inversion bubble to push forward. Since the source node is unmodified it is left unchanged but the RF sequence is modified, as shown in Table 2.3 for Fig. 2.12d. As shown in the table, the original RF sequences do not match but after performing the equivalence modification they do. A similar process is done for all sequences to standardize the types of signatures and enable a more comprehensive matching process.

Table 2.3: Sequence Conversions to DeMorgan Equivalents

Sequences	Fig. 2.12c	Fig. 2.12d
Original RR Sequence	(OR, AND)	(NOR, AND)
Original RF Sequence	(OR, NOR)	(NOR, AND)
DeMorgan RR Sequence	(OR, AND)	(OR, AND)
DeMorgan RF Sequence	(OR, NOR)	(OR, NOR)

Table 2.4: GRAPPLE results on several ISCAS'85 benchmarks

Test Article	Logic Locking Type	Correct Keys	Wrong Keys	Unknown Keys
c5315	Random	47	12	5
c3540	Random	22	7	3
c6288	Random	57	6	1
c5315	Random	27	3	2

2.9 Results

A few samples of locked ISCAS'85 benchmarks were taken from Trust HUB [6], and the results of applying GRAPPLLE to the benchmarks are shown in table 2.4. From the four tested circuits, there were a total of 192 keys where 154 were predicted correctly, 28 were predicted incorrectly and 11 were unknown. This means that GRAPPLLE was able to correctly predict 80% of the keys, while only incorrectly predicting 18%. In several cases there were keys that could not be determined, which occurred for a variety of reasons. In some cases, the IN and NI key signatures had the exact same match scores, which made determining the correct bit impossible. Another common occurrence is an XOR/XNOR gate being predicted as a NAND, NOR, AND, or OR gate or vice versa. Considering there is no direct equivalent from these gates, there was not a straightforward way to predict the key value. To correct for this inability to compare XOR gate types, we propose a new matching method.

2.10 Improving GRAPPLLE

While graph pattern matching has promising results, there are some limitations of a gate signature based approach. In particular, relying only on gate types reduces the information fidelity of the signatures by removing any similarity between different

Table 2.5: GRAPPLE versus simulation results (a) NAND XOR simulations, (b) 2-input 2-depth gate sequences, (c) 2-input 2-depth gate simulation.

A	B	NAND	XOR
0	0	1	0
0	1	1	1
1	0	1	1
1	1	0	0

(a)

Output	Sequence
F1	OR, NAND
	OR, NOR
F2	OR, NAND
	OR, NAND

(c)

Inputs	F1	F2
0000	0	0
0001	0	0
0010	0	0
0011	0	0
0100	1	1
0101	0	0
0100	0	0
0111	1	1
1000	1	1
1001	0	0
1010	0	0
1011	1	1
1100	1	0
1101	0	0
1110	0	0
1111	1	0

(b)

gate types. For instance, a simple 2-input NAND and a 2-input XOR gate, according to GRAPPLLE, are not equivalent in any way, which results in any sequence with an XOR gate having zero similarity to any sequence with a NAND gate in the same position. However, the exhaustive simulation results in Table 2.5a illustrate that 50% of input to output vectors between a NAND and XOR gate are the same. Consequently, GRAPPLE tends to underestimate the signature similarity because it can not capture partial gate equivalence. Furthermore, this principle can be expanded to larger sub-circuits as shown in Fig. 2.13 and functionally represented in Table 2.5b, where the results would yield a Jaccard Index of 0.68, indicating a high similarity between the two circuits. However, GRAPPLLE, using the RR sequences in Table

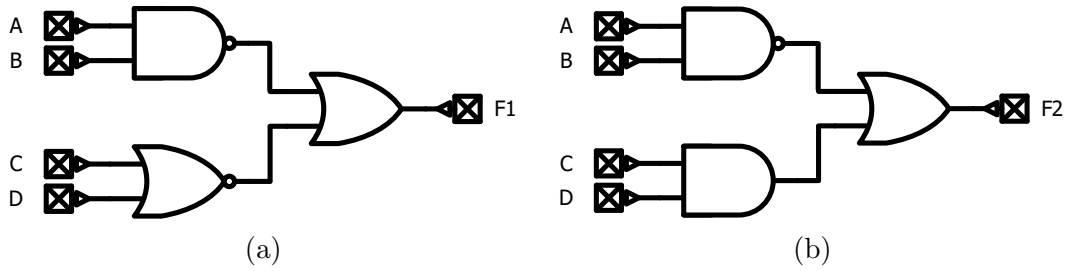


Figure 2.13: Sub-circuits that produce similar outputs: (a) F1; (b) F2.

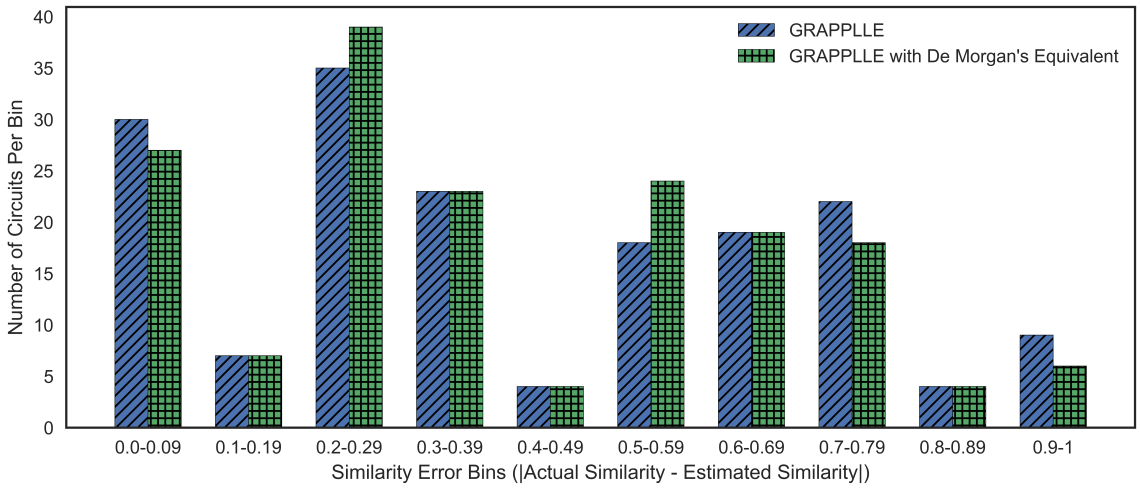


Figure 2.14: Information loss due to similarity error. Data is generated from circuits with two logical depth and the error is calculated as the difference between similarity through simulation and the estimation technique. The x-axis is the error amount divided into 10 bins and the y-axis is the number of circuits in each bin.

2.5c, yields a Jaccard Index of 0.33, indicating a medium to low similarity. Moreover, this phenomenon is not limited to a few sequences. To illustrate, Fig. 2.14 shows the information loss over a set of circuits composed of three 2-input gates where the first two gates drive the third, as previously shown in Fig. 2.13. The set includes every permutation of NAND, AND, NOR, OR, XOR, and XNOR gates for a total of

216 circuits. From the figure, a score of zero indicates no error between GRAPPLLE and simulation results, while a score of one indicates 100% error using GRAPPLLE. Additionally, the figure also indicates that there is only a minor improvement when using the De Morgan equivalent transformations, as the results are nearly identical. Therefore, to increase the accuracy of GRAPPLLE, the information loss needs to be remedied, but this must be accompanied without hindering the efficiency of the analysis process.

2.11 Limitations on Simulations

An exhaustive set of simulations will yield the most accurate representation of the sub-circuit functions. However, actually generating simulations and comparing the results between sub-circuits can prove to be prohibitively difficult for a few reasons. For one, the sample circuits thus far are purposefully laid out in identical structures, in that they are made of four total inputs which drive two 2-input gates, then those gates drive a single 2-input gate. This enables gates to be aligned which in turn allows the IO to be perfectly mapped between the sub-circuits. However, such as in Fig. 2.15, there is no indication which input from one sub-circuit matches an input to another sub-circuit and therefore, any increase in complexity yields a node correspondence problem between the sub-circuits being compared. This is illustrated by Figs. 2.15a and 2.15b having mismatched input labels. Second, the sub-circuits will not always be the same shape or have the same number of inputs and gate counts, which is demonstrated by the additional gate, shown in gray, in Fig. 2.15b. For this reason, exhaustive simulations are difficult due to the IO vector size disparity. Ultimately, an improved approach should yield a higher degree of fidelity into the sub-circuits, while

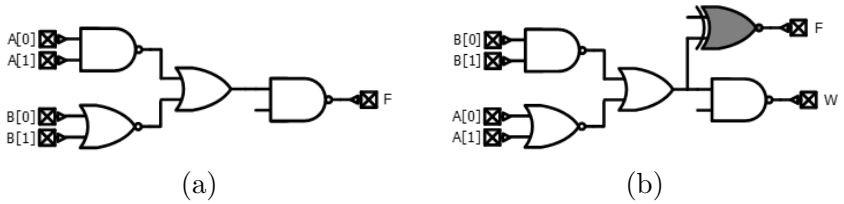


Figure 2.15: Similar sub-circuits: (a) Original sub-circuit; (b) Similar sub-circuit with mismatched input pins, an extra gate in the fan-out logic (shown in gray), and output pin switched.

also not requiring solutions to node correspondence or needing identically structured circuits.

2.12 Symmetric Simulation

The proposed method to improve the previously discussed deficiencies in GRAPPLLE is a symmetric simulation, which uses a hybrid approach between the sequence matching and an exhaustive simulation. Similar to sequence matching, the sub-circuits are broken into separate sequences, where sub-circuit 1, shown in Fig. 2.16a, breaks into two sequences, as shown in Fig. 2.16b and 2.16c.

In addition to breaking each sub-circuit into sequences, each sequence will have a corresponding driver logic. The driver logic serves as the logic to secondary inputs that are not along the main sequence path. In this example, the driving logic is represented by single gate but it can be expanded further back to include more fan-in logic. For the purpose of maintaining an input agnostic approach, instead of matching the sequences directly, they are both simulated. It is not possible to simulate a sequence alone, as there is a second input on the 2nd logic depth of each sequence. To account for this, the simulations require primary inputs $P1$ and $P2$, which correspond to inputs that

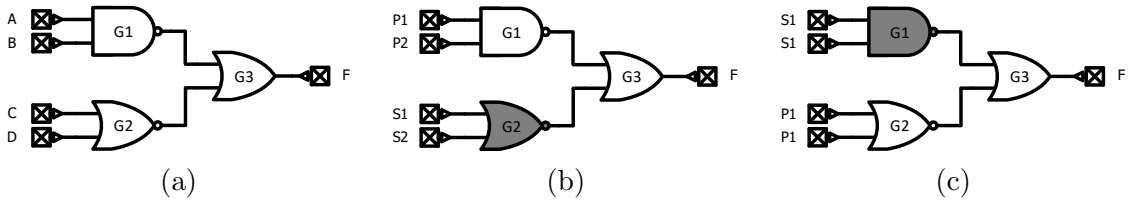


Figure 2.16: Sub-circuit 1: (a) Original circuit; (b) First sequence in white with secondary driver in gray; (c) Second sequence in white with secondary driver in gray.

control the main sequence. The driver logic controls that second input at the 2nd logic depth and is controlled by the secondary inputs $S1$ and $S2$. Separating the sub-circuits into sequences, and differentiating driver logic removes the necessity of computing permutations of the inputs. In the example from Fig. 2.16, the driver circuit is only one gate as shown in gray. For each sequence, a signature is generated through an exhaustive input and output vector set where each input vector for each sequence generates a sub-signature as (2.10).

$$SigSym = (P1, P2, S1, S2, F, X) \tag{2.10}$$

where $P1$ and $P2$ are the primary inputs, $S1$ and $S2$ are secondary inputs, F is the output, and X is the number of times that specific input/output vector has occurred. Given, that signatures are generated for both sequences of the sub-circuit, the possibility of duplicate signatures exists. Altogether, the final signatures have 32 vectors or twice the original number compared to a standard simulation. Each of the two sequences has four inputs $P1, P2, S1, S2$ which when simulated exhaustively generates 16 vectors per sequence for 32 total. However, the growth rate of signatures is slower than exhaustive simulations so a 2-depth sub-circuit is the only case where there are more signatures than simulation vectors. As an example, increasing the

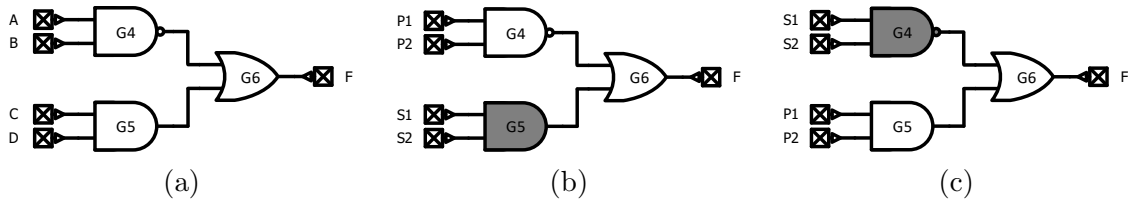


Figure 2.17: Sub-circuit 2: (a) Original circuit; (b) First sequence in white with secondary driver in gray and (c) second sequence in white with secondary driver in gray.

logic depth to three would yield 8 inputs, 256 inputs vectors, and 24 permutations for a standard simulation. Therefore, a total of $256 * 24 = 6,144$ IO vectors need to be generated. A symmetric simulation, would yield 4 sequences with 6 input vectors each for a total of only 256 IO vectors. Additionally, no permutations are required so the number of matches needed for 2-depth circuits is the same between GRAPPLLE and simulations. As an example, the sequences from sub-circuit 1 in Fig. 2.16 and sub-circuit 2 in Fig. 2.17 produce the vector table, shown in Table 2.6. The table demonstrates the 32 IO vectors for each sub-circuit, where an IO vector consists of the Inputs $P1, P2, P3$, and $P4$, as well as F and X from one of the sequences (A or B). In the final analysis, comparing the 32 IO vectors from sub-circuit 1 to the 32 IO vectors to sub-circuit 2, the symmetric simulation Jaccard Index matches the original standard simulation Jaccard Index of 0.77.

In order to compare the symmetric simulation approach versus sequence matching, an exhaustive set of benchmark circuits composed of three 2-input gates with a logical depth of two, structured as in Fig. 2.16 and Fig. 2.17, was analyzed and is shown in Fig. 2.18. As demonstrated in the figure, the similarity error is substantially decreased using the symmetric simulation approach where the maximum error,

Table 2.6: Symmetric Simulation IO Vectors

Inputs				Sub-Circuit 1 Fig. (2.16)				Sub-Circuit 2 (Fig. 2.17)			
				Seq A		Seq B		Seq A		Seq B	
P1	P2	S1	S2	F	X	F	X	F	X	F	X
0	0	0	0	1	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1	1	0	1	1
0	0	1	0	1	0	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	0	0	0
0	1	0	0	1	0	1	1	1	0	1	1
0	1	0	1	1	0	1	1	1	0	1	1
0	1	1	0	1	0	1	1	1	0	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	0	1	1
1	0	1	0	1	0	1	1	1	0	1	1
1	0	1	1	1	0	0	0	1	0	0	0
1	1	0	0	1	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0	0	0	1	0
1	1	1	0	0	0	1	0	0	0	1	0
1	1	1	1	0	0	0	1	1	0	1	1

$|ActualSimilarity - EstimatedSimilarity|$, is now under 0.5 and over 90% of the test cases had a similarity error of less than 0.1.

At first glance, the symmetric simulation method seems identical to the exhaustive simulation approach. As long as sub-circuits have a logical depth of two, this is accurate, however, the symmetric simulation approach can be applied to larger sub-circuits such as Fig. 2.19 with a logical depth of three. In this scenario, the sequences are three gates long with primary inputs P, secondary inputs S, and tertiary inputs T. As shown in Fig. 2.19 the tertiary inputs are repeated across the two gates that create the driving circuit which eliminates the need for input permutations. The IO vectors are stored similarly to the two depth sub-circuits, as in (2.11)

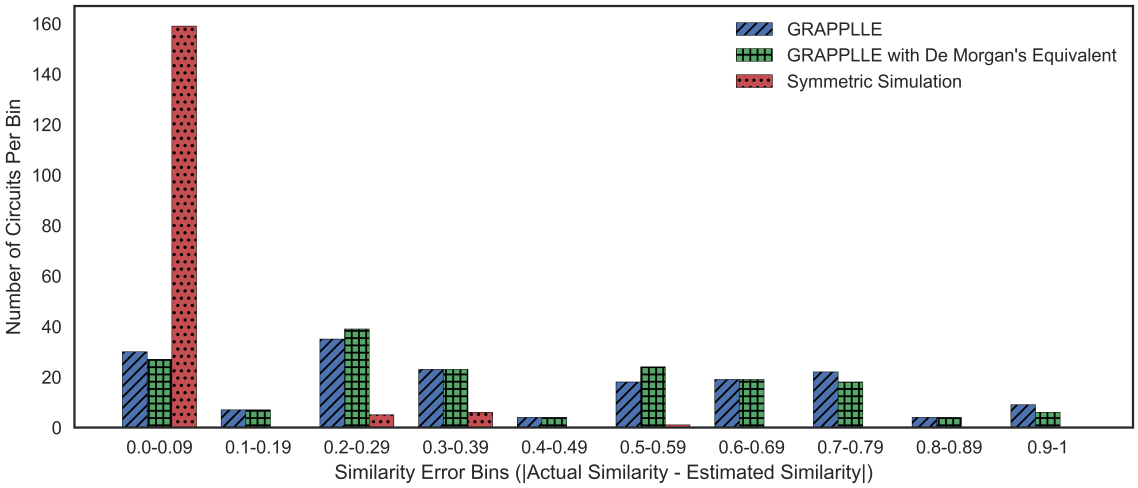


Figure 2.18: Information loss due to similarity error. Data is generated from an exhaustive set of 2-input gates with depth of two circuits and the error is calculated as the difference between actual similarity through a simulation and the estimation technique. The x-axis is the amount or error divided in 10 bins and the y-axis is the number of circuits that fit into each error bin.

$$SigSym = (P1, P2, S1, S2, T1, T2, F, X) \tag{2.11}$$

where $P1$ and $P2$ are the primary inputs, $S1$ and $S2$ are secondary inputs, $T1$ and $T2$ are tertiary inputs, F is the output, and X is the number of times that specific IO vector has occurred. To illustrate the accuracy of the symmetric simulation, a sample of 1 million symmetric simulations versus actual simulations is shown in Fig. 2.20. As demonstrated in the figure, the majority of the 1 million comparison fall under an error of 0.1 with a maximum error under 0.6.

The previous discussion only covers how to simulate the fan-in logic for a net. However, the simulation process for the fan-out logic follows a similar approach, where all the possible gate sequences are generated with their respective driving

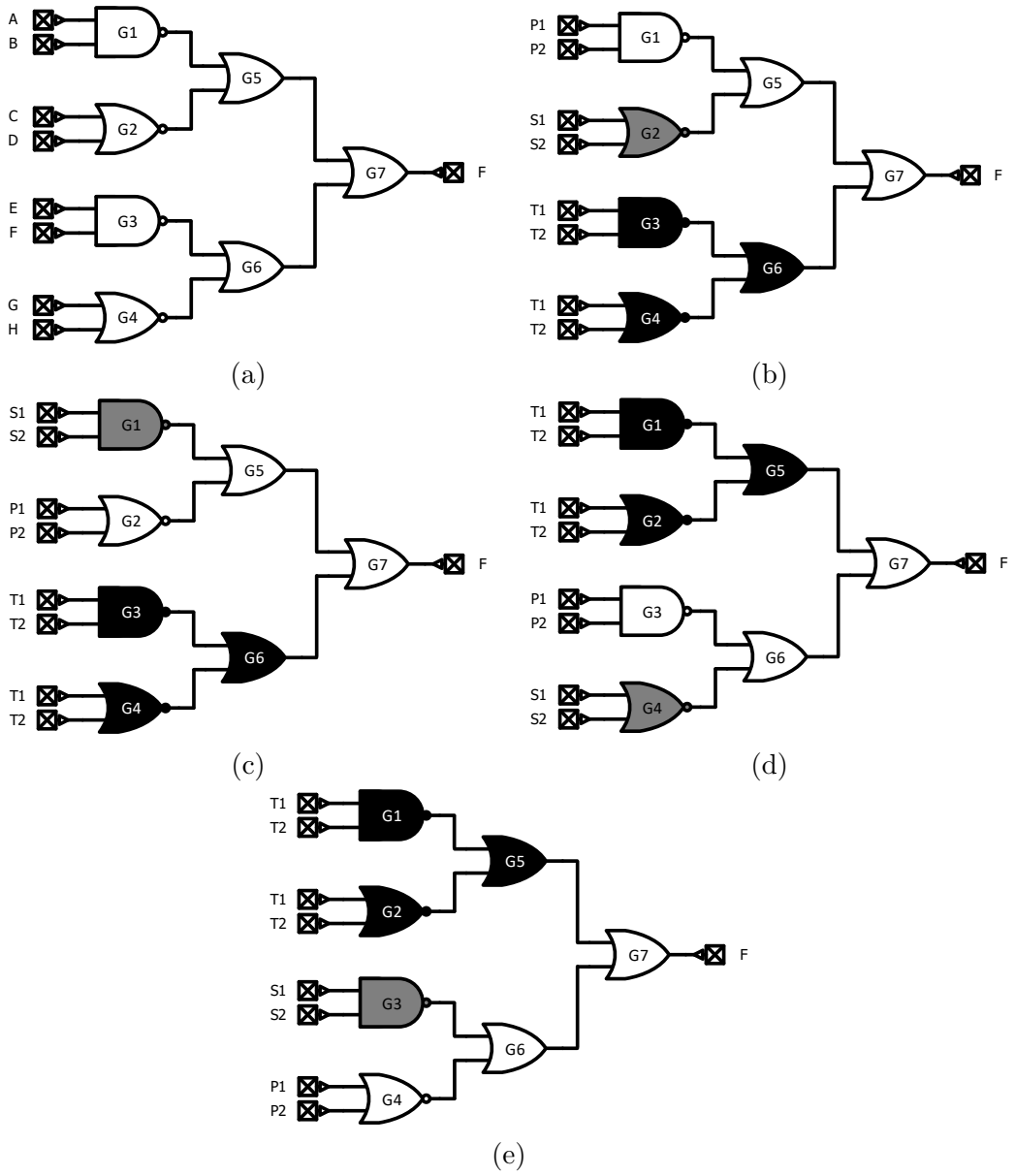


Figure 2.19: Sub-circuit with logic depth of three with sequence (white), secondary driver circuit (gray), and tertiary driver circuit (black): (a) Original Circuit; (b) First sequence; (c) Second sequence; (d) Third sequence; (e) Fourth sequence.

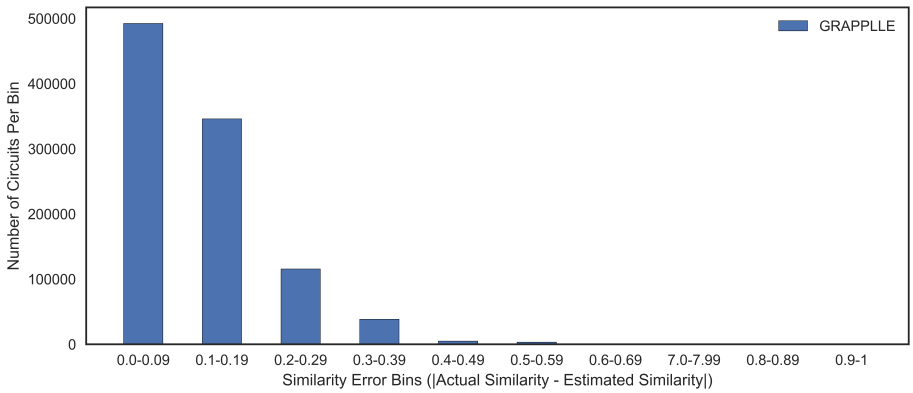


Figure 2.20: Estimated similarity error using symmetric simulation on 2-inputs gates with a logic depth of 3. Data is constructed from 1 million data points (1,000 random circuits compared to a second set of 1,000 random circuits). Error is calculated as the difference between actual similarity through a simulation versus the symmetric simulation. The x-axis is the amount or error divided in 10 bins and the y-axis is the number of circuits that fit into each error bin.

circuits. Fig. 2.21 shows how to break the fan-out into sequences with the appropriate driver circuits. Similar, to the fan-in sequences, there are primary, secondary and tertiary inputs depending on the logic depth, and all the possible forward sequences are separated and simulated.

2.13 Symmetric Simulation to Key Prediction

Considering the symmetric simulation approach to comparing sub-circuits, the original method to predict key-gates needs to be modified. However, the same principals still hold. Specifically, the logic locked circuits are re-synthesized into NI and IN versions, and sub-circuits are still created. However, the sub-circuits are centered around the input net to a key-gate instead of the driving gate. As a result, the net just prior to a key-gate will be used to predict the key value, and henceforth, be referred

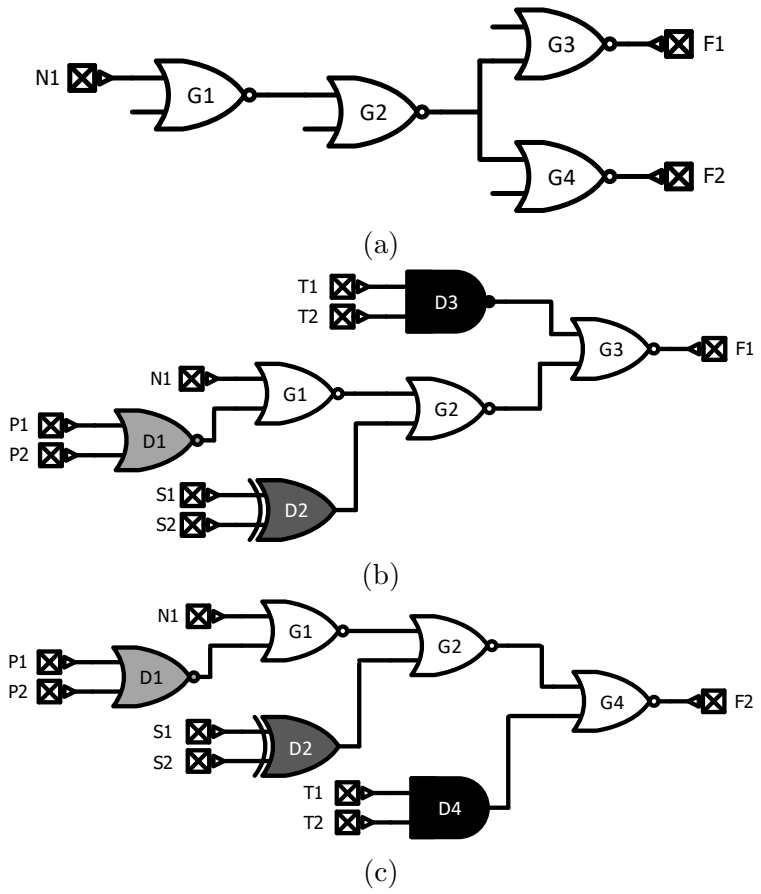


Figure 2.21: Forward Sub-circuit with logic depth of three with sequence (white), secondary driver circuit (gray), and tertiary driver circuit (black): (a) Original circuit; (b) First sequence; (C) Second sequence.

to as the key-net. Therefore, each key-net will have an inverted and non-inverted version from its respective netlist. Additionally, the sub-circuits now only produce two sub-signatures, which are Forward and Reverse. While each sub-signature still generates sequences as before, now these sequences are symmetrically simulated, storing the simulations as the signatures. Similar to sequence comparison, the best matching signature between the key-nets is resolved into the predicted key value, with the match score 2.12

$$MS(i,j) = \frac{|A_F \cap B_F| + |A_R \cap B_R|}{|A_F \cup B_F| + |A_R \cup B_R|} \tag{2.12}$$

where $MS(i,j)$ is the match score which is between zero and one; zero indicating the nodes are not at all similar and one indicates the nodes are identical. It is important to note that the match score equation has been modified for the symmetric simulations versus the sequence matching. This is due to sequence amount disparities from using a 3-logic depth fan-in equation and a 2-logic depth fan-out equation. Therefore, the fan-in equation will weigh more heavily due to having a larger number of simulations.

2.14 Results

A sampling of test articles was taken from Trust Hub. The intention here was not to try different types of LL but to attempt to get versions of locked test articles where there was no interference between key-gates. However, key-gate interference for GRAPPLLE is not the same as functional interference. For functional testing, as long as two key-gates are in the same data path between an input and an output they then interfere. For GRAPPLLE, interference only occures when key-gates are graphically close together such that their sub-circuits overlap. Consequently, only random logic locking was used to minimize key-gate interference. Notwithstanding, the technique is able to perform on interfering gates. However, interfering gates must be solved at the same time such that the sub-circuits will encompass both gates and generate sequences for the four possible key permutations. However, the python scripts to generate and compare signatures are not programmed to account for overlapping key-gates, therefore it was attempted to minimize this occurrence. Additionally,

Algorithm 1: Key Bit Solving Process

```
Result: Solve Key Bits
MinimunJaccard = 0.7;
DeleteKeyGates = False;
while JS > 0 do
    if DeleteKeyGates == True then
        Generate signature for all nodes and key-gates;
    else
        Generate signature for all nodes and key-gates;
        Returning False if signature overlaps with key-gate;
    end
    JS = jaccard index of best matching key-gate;
    if JS ≥ MinimunJaccard then
        Update Netlists;
        DeleteKeyGates = False;
    else if DeleteKeys == False then
        DeleteKeys = True ;
    else
        MinimunJaccard = MinimunJaccard - 0.1 ;
        DeleteKeys = False;
    end
end
```

other locking methods may require additional modifications to sequence generation and further complicate the key prediction process.

In order to minimize the impact of key-gate correlation, the bits were solved using Algorithm 1. The process works by first attempting to solve uncorrelated key-gates, which is accomplished by only generating signatures that do not contain any other key-gates. Consequently, any net, including key-nets, are ignored in this first step if another key-gate resides in its signatures. Next, after each matching round, the netlists are updated by solving the highest scoring key and the process is repeated. Similar to sequence matching, a key value is predicted based on the highest match score of a key-net and the netlist it resides in. When there are no more uncorrelated

Table 2.7: Symmetric simulation GRAPPLLE results on several ISCAS'85 benchmarks.

Benchmark Circuit	Number of Keys	Correct Keys Predicted	Wrong Keys Predicted	Unknown Keys
c880	32	18	9	5
c1908	32	18	9	5
c2670	32	18	12	2
c3540	32	16	16	0
c5315	32	27	5	0
c6288	32	20	8	4
c7552	32	19	13	1

key-net signatures that have a high enough match score, the key-gates are removed from the netlist and therefore signatures can include previous sections of the design where key-gates resided. If a sufficient match score is then located, the netlist is updated and the algorithm tries to solve the uncorrelated key-gates again. Otherwise, if no sufficient match is found, the minimum match score is lowered, and the process starts again. It should be noted that for this round of testing, the NI and IN netlists are not re-synthesized after solving a key bit. Instead, the netlists are simply modified by removing the key-gate, inserting the appropriate inverter, and then reread back into the matching algorithm. While re-synthesis would yield the most accurate results, it is also substantially more time consuming.

The results of several test articles are shown in Table 2.7. For the symmetric simulations, the results are mixed, with GRAPPLLE correctly predicting 84% of c5315 while only correctly predicting 50% of c3540, which is no better than random guessing. However, the remaining test articles fall within 18 to 20 correctly predicted keys and 9 to 13 incorrectly predicted keys out of 32. Additionally, some keys were

not predicted due to either no sufficient matching node, or more commonly, both the NI and IN key-nets matched identically. All together, GRAPPLLE predicted 60% of the key-bits correctly, 32% incorrectly, and 16% unknown. It should be noted that the results will most likely improve if the benchmarks are re-synthesized after every key update, as nearly all key-gates had some interference. Additionally, the new version of GRAPPLLE had greatly increased key-net match scores, so the incorrect prediction was not due to insufficient matches but the similarity of the IN and NI matches. Additionally, the match score of the predicted key bits versus the delta between the NI and IN scores are plotted in Fig. 2.22. As demonstrated in Fig. 2.22, there was nearly a 100% accuracy in predicting key-gates when the match score is 1.0. Furthermore, the average match score of a correct prediction is 0.84 while the average match score of an incorrect prediction is 0.80. Additionally the average difference between NI and IN match scores for correct predictions is 0.12 while the average difference for incorrect predictions is 0.10. Therefore, these results do indicate that increases in match scores and increases in differences between NI and IN match scores increases the likelihood that key bit predictions are correct. Further analysis and investigation needs to be performed to increase the accuracy of GRAPPLLE, however the results still show substantial information leakage that is not being accounted for in current LL metrics.

2.15 Results Summary and Future Direction

Through this analysis we have shown that LL can be broken by analyzing sub-graph repetition. In some circuits 80% of the key-values were recovered without using an oracle or any functional testing. This technique, in essence, increases the amount

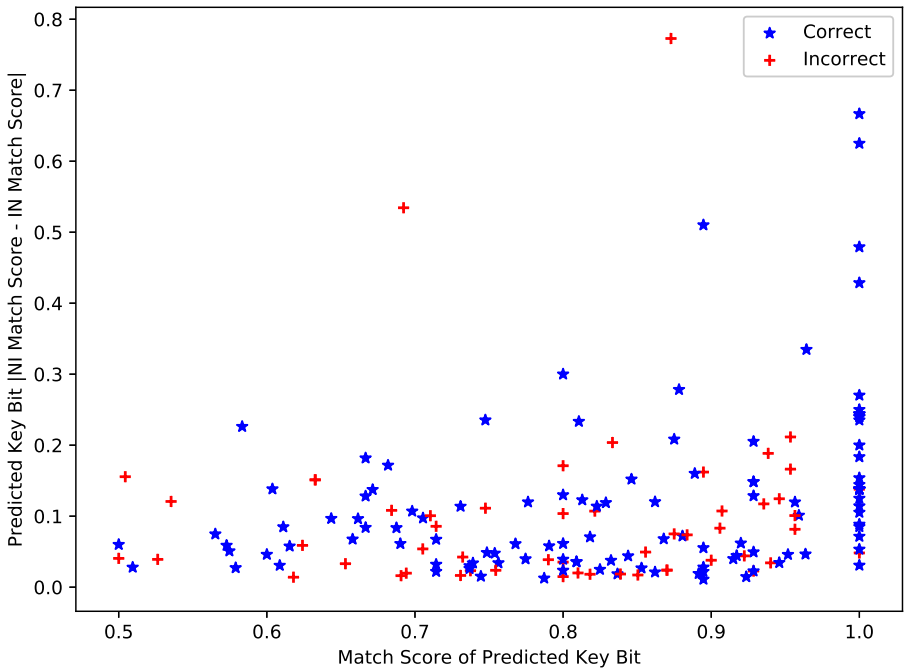


Figure 2.22: Match score and difference of NI and IN match score for predicted key-gates from c880, c1908, c2670, c5315, c6288, and c7552 ISCAS'85 Benchmarks.

of observability into an IC, which defeats the very purpose of anti-SAT techniques. Notwithstanding, the promising results still have several improvements that can be implemented. First, incorporating the synthesis tools into the solving algorithm would most likely increase the accuracy of the symmetric simulation approach and greatly increase the solving speed. Second, the sub-graphs can be altered to incorporate multiple key-gates or take into account key correlation to increase the solving accuracy further. Finally, the key prediction algorithm could be improved to incorporate multiple metrics along with the current method of the highest match score.

Chapter 3: Global Structural Analysis

The GRAPPLLE method focuses on exploiting low level repetition. However, in large designs, executing GRAPPLLE across the entire circuit may be unfeasible. Additionally, the information may prove to be a detriment to predicting key values. The next level of abstraction aims to exploit module repetition across a circuit through the analysis of a more global structure. Most previous research, such as SAT solvers, are limited to a localized circuit view. However, without taking the global picture into account can leave locked circuits vulnerable.

Rather than attacking a chip as a large black box or a collection of disconnected, localized structures around key-gates, the entire formation of a circuit can be attacked. Exploiting circuit formation is not new as previous research has utilized the comprehensive design structure to reconstruct a circuit's hierarchy and identify its function through comparison to a library of representative circuits [42], [14]. While this research has been geared towards HT detection, the same concepts can be applied to obfuscated designs to decipher their true functionality. On the defensive side, these attack methods can also be utilized to determine if a locked design is structurally distant enough from its source to maintain secrecy of its functionality.

The portion of the ISCAS'85 c6288 multiplier benchmark circuit, shown in Fig. 3.1, illustrates the false sense of security SAT solving run times provide and validates

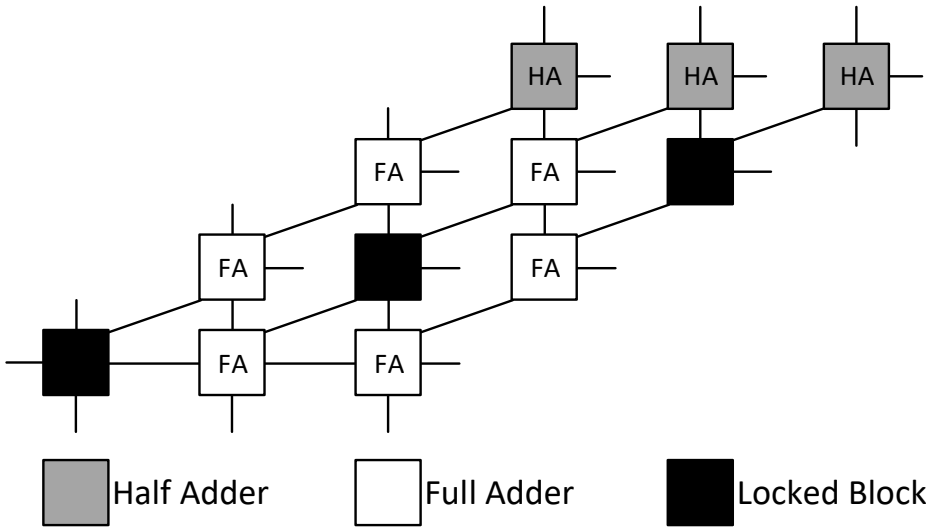


Figure 3.1: Logic locked multiplier composed of half adders (gray), full adders (white), and unknown blocks containing a locked element (black).

the need for a structural difference metric. The work in [27] has shown that this circuit requires the least amount of LL to time-out a SAT solver compared to the other tested ISCAS'85 benchmark circuits. Due to its highly interconnected structure, the key bits are entangled, which forces the entire circuit to be solved as a whole instead of solving one key or small sections of keys at a time. This perceived sense of security only holds true if the attacker relies solely on the SAT solver while ignoring the underlying structure and heavy repetition. To demonstrate this point, several obfuscated elements were placed in some of the blocks so they could not be identified as full adders, as demonstrated by the black blocks in Fig. 3.1. Visually, however, there are instantly identifiable structural patterns in the circuit and many representations of unlocked full adders nearby. Additionally, given the limited number of alternative configurations, it is not unreasonable to assume that a locked multiplier can be compared against the small set of multiplier candidates to decipher its functionality.

3.0.1 Global Structural Attack Vectors

Another more complex example to investigate is an Advanced Encryption Standard (AES) core, where the main components are listed in Table 3.1. Clearly, there is a high degree of repetition in an AES core, which is only made up of a few different types of modules. These individual modules can have local structural uniqueness, but the repetitiveness of the AES core provides ample examples of similar modules. There is also the potential to have multiple AES cores on a single chip, and thus, careful consideration of the entire chip must be observed. Failure to account for repetition can lead to identical modules on the same chip being implemented with different LL placements. Consequently, an attacker can compare a non-locked section of one of the identical modules to unlock the other. However, even if all instances of a particular module are locked with identical key-gate placements, the repetition of these instances still reveals clues that can be utilized to extract their function based on structural similarity. As will be further described in this paper, this concept can expand to any circuit as the vast amount of repetition and IP reuse found in modern ICs provide ample data to construct a circuit library in which to attack LL.

Table 3.1: Core AES Components

Module	Number of Instances
Substitution Box (SBOX)	200
Inverse SBOX	144
Expand Key	10

3.1 Proposed Method for Measuring Netlist Similarity

As previously detailed, LL disrupts functional behavior, but it does not substantially alter the structural appearance. However, unlike the c6288 example, most designs are challenging to visualize, and therefore, relying on qualitative visual similarity is not practical. Thus, a quantitative method of similarity that depends on the graphical properties of a circuit's structure is warranted.

3.1.1 Structural Similarity Analysis: Fundamentals

A structural analysis method, referred to as NetSimile, was proposed by the authors in [8] to determine the similarity of graphs and identify if graphs were produced from the same type of data, e.g., coauthorship networks [39], Forest-Fire [23], and movie collaboration networks [1]. The methodology extracts features from all nodes in a network and condenses this information into a fixed length description vector. These description vectors can then be used to assess the similarity of two networks. To perform this assessment, a distance measure is used between the description vectors, where a smaller distance represents a larger similarity between the networks. Note that a graph comparison technique was also used in [14] to identify circuits such as AES cores and fast Fourier transforms from a limited library of sample circuits. However, the technique used reduced dimensional representations of graphs, and still required graph alignment to compare circuits. The NetSimile method of comparison is adopted in this work for a variety of reasons, but most importantly its scalability. From an attacker's perspective, performing many comparisons on large networks must be feasible, emphasizing the importance of scalability. According to the authors in

[8], NetSimile scales at $O(n)$ for vector generation and $O(n\log(n))$ for distance calculations. Another reason NetSimile is adopted for this work is that it is not dependent on gate type or design function but only on the structural nature of its graph; hence, simple gate replacement is expected to have little to no effect on its efficacy. Finally, the method is flexible in that additional features can be included in the description vectors. The flexibility of this method is crucial when considering that different LL techniques and various circuit types may require entirely new network features in the description vectors.

The NetSimile algorithm relies on extracting several features from each node inside of a network. To enable efficient comparisons, the feature data is condensed into description vectors containing the mean, median, standard deviation, skew, and kurtosis of each feature type, as defined in (3.1)

$$\begin{aligned} DV_j = \{&mean(feature_1), median(feature_1), stdev(feature_1), skew(feature_1), \\ &kurtosis(feature_1), mean(feature_2), median(feature_2), stdev(feature_2), \\ &..., kurtosis(feature_n)\} \end{aligned} \tag{3.1}$$

where j is an analyzed circuit and DV_j is the description vector of circuit j. The similarity of two circuits is then measured by using a distance formula, such as the Euclidean distance, between the vectors of the two graphs. A high-level algorithmic overview of how NetSimile compares the similarity of two circuits, j and k, is described in Fig. 3.2.

3.1.2 NetSimile For Circuit Networks Analysis

While the original features NetSimile used in its description vectors provide a starting point to describe a general graph, they do not lend themselves to circuit-based

networks. There are two key differences between circuit-based graphs and the original models used in NetSimile. First, edges, represented as wires, in circuit-based graphs are assigned directions. Their direction is determined by the nodes, represented as gates, and their clearly defined inputs and outputs. On the other hand, in a general graph, edges have no direction, therefore, disregarding the direction of the edges ignores information that can otherwise prove useful. Second, circuit-based graphs are considered sparse, such that there are a finite number of edges per node due to fan-out and fan-in limitations. In contrast, for a general graph, such as one derived from groups of friends on a social media site, each node or user can have hundreds of friends, where friendships are represented as edges. Thus, allowing a large number of edges per node creates graphs with higher edge densities. Additionally, given that the original features from NetSimile assumed the graphs contained these higher edge densities, some of the original features do not generate useful data for circuit-based graphs.

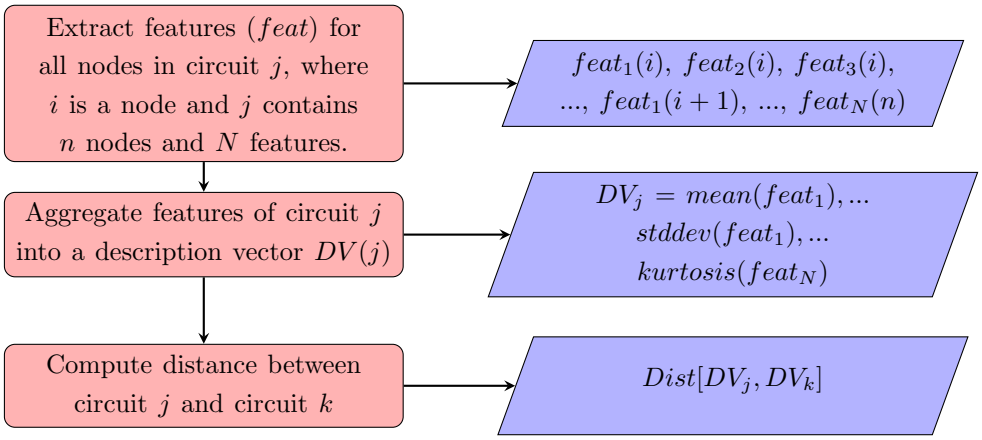


Figure 3.2: High level algorithm flow for generating and comparing description vectors of circuits j and k.

Table 3.2: Circuit Graph Properties

Property	Description
$N_o(m, H)$	Neighbors in the fan out of node m, not beyond H hops.
$N_i(m, H)$	Neighbors in the fan in of node m, not beyond H hops.
$E_o(m, H)$	Edges in the fan out of node m, not beyond H.
$E_i(m, H)$	Edges in the fan in of node m, not beyond H hops.
$ego(m, H)$	Egonet of node m, defined as the subgraph comprised of $N_o(m, H)$ and $N_i(m, H)$
$E^i_{ego}(m, H)$	Set of interior egdes of the egonet of node m, shown in Fig. 3.3c. where the egonet is not beyond H hops of m.
$E^o_{ego}(m, H)$	Set of external edges of the egonet of node m, (Edges exactly $H + 1$ hops from m), shown in Fig. 3.3d.
$N(ego(m, H))$	Neighbors of the egonet of node m, where the egonet is not beyond H hops of node m shown in Fig. 3.3d.

3.1.3 Netlist Features

With the graph properties refined to account for circuit-based networks, shown in Table 3.2, the following features are used to describe a given circuit's structure. The features detailing the number of neighbors are described in (3.2) and (3.3),

$$f^o_m = |N_o(m, 1)| \tag{3.2}$$

$$f^i_m = |N_i(m, 1)| \tag{3.3}$$

where f^o_m is the number of neighbors in the fan-out of node m and f^i_m is the number of neighbors in the fan-in of node m, as shown in Fig. 3.3a. The convergence coefficients are described in (3.4) and (3.5),

$$cc_m^o = \frac{|N_o(m,H)|}{|E_o(m,H)|} \tag{3.4}$$

$$cc_m^i = \frac{|N_i(m,H)|}{|E_i(m,H)|} \tag{3.5}$$

where cc_m^o is the convergence coefficient of fan-out neighbors of node m and cc_m^i is the convergence coefficient of fan-in neighbors of node m. The convergence coefficient refers to the ratio of total neighboring nodes to total edges not beyond H hops, and $H \geq 2$ (Fig. 3.3b). The averages of the neighbor's fan-out and fan-in are described in (3.6) and (3.7),

$$NF_o(m) = \frac{1}{f_m^o} \sum_{\forall p \in N_o(m,1)} f_p^o \tag{3.6}$$

$$NF_i(m) = \frac{1}{f_m^i} \sum_{\forall p \in N_i(m,1)} f_p^i \tag{3.7}$$

where $NF_o(m)$ is the average fan-out of the fan-out neighbors p of node m and $NF_i(m)$ is the average fan-in of the fan-in neighbors p of node m. The averages of the neighbor's convergence coefficients are described in (3.8) and (3.9),

$$cc_{N_o(m)}^o = \frac{1}{f_m^o} \sum_{\forall p \in N_o(m,1)} cc_p^o \tag{3.8}$$

$$cc_{N_i(m)}^i = \frac{1}{f_m^i} \sum_{\forall p \in N_i(m,1)} cc_p^i \tag{3.9}$$

where $cc_{N_o(m)}^o$ is the average convergence coefficient of fan-out neighbors p of node m, and $cc_{N_i(m)}^i$ is the average convergence coefficient of fan-in neighbors p of node m. The number of edges inside the egonet is described in (3.10),

$$e_m^i = |E_{ego}^i(m,H)| \tag{3.10}$$

where $E^i_{ego}(m, H)$ is the set of interior edges of the egonet (Fig. 3.3c) of node m and e^i_m is the number of edges in $E^i_{ego}(m, H)$. The number of external edges to the egonet is described in (3.11),

$$e^o_m = |E^o_{ego}(m, H)| \tag{3.11}$$

where $E^o_{ego}(m, H)$ is the set of external edges of node m and e^o_m is the number edges in the set (Fig. 3.3d). Lastly, the number of neighbors to the egonet is described in (3.12),

$$n^o_m = |N(ego(m, H))| \tag{3.12}$$

where n^o_m is the number of neighboring nodes to the egonet of m. Note that this can be smaller than external edges as neighboring nodes can have multiple edge connections (Fig. 3.3d).

In total, 11 features are extracted from every node in the circuit. Each feature type then contributes five elements to the description vectors, which are the mean, median, standard deviation, skew, and kurtosis of that feature. Therefore, the final description vector for a given circuit consists of 55 elements, five for each of the 11 features.

3.1.4 Determining Egonet Size

The convergence coefficients, (3.4) and (3.5), are created specifically for this analysis, and are intended to describe the density in which nodes cluster together. However, in a general graph, density is described by the clustering coefficient, a common network property, which is defined by (3.13)

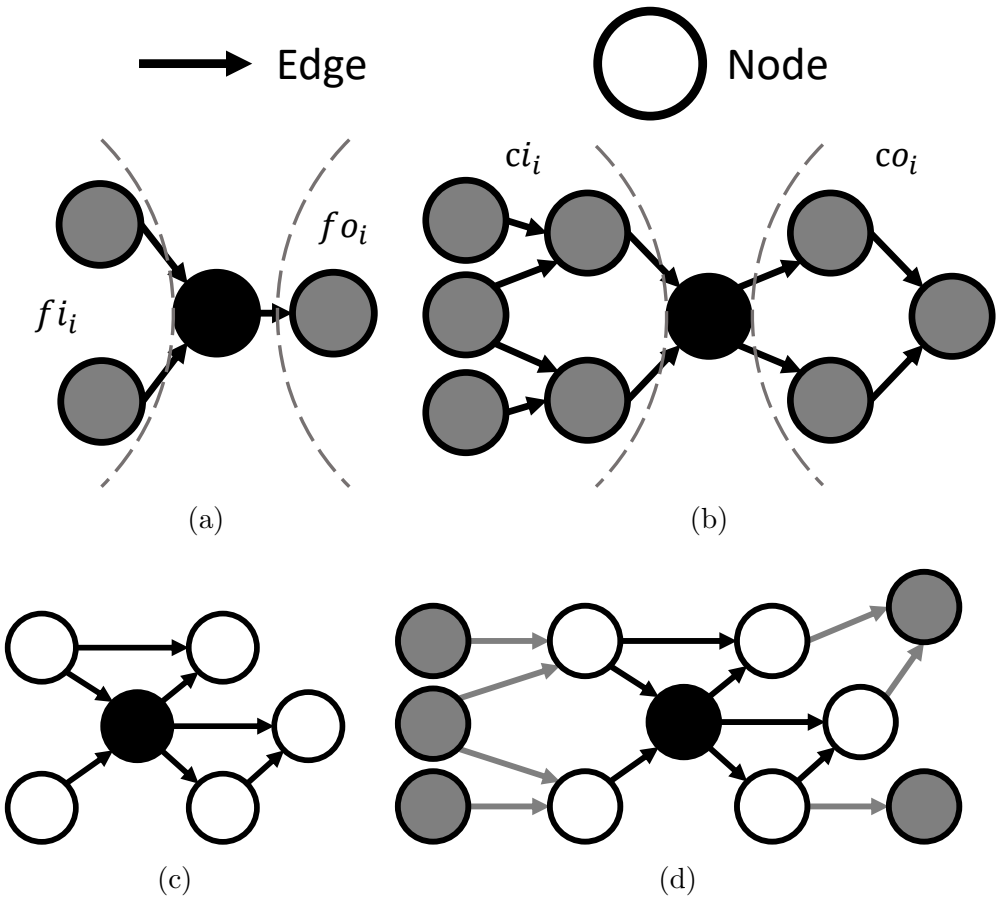


Figure 3.3: Extracted features of node m (shown in black): (a) fan-out and fan-in neighbors $H = 1$ hops away, where $f_m^o = 1$ and $f_m^i = 2$; (b) Convergence coefficient, where $cc_m^o = \frac{3}{4}$ (3 nodes and 4 edges), $\text{cc}_m^i = \frac{5}{6}$ (5 nodes and 6 edges) for $H = 2$; (c) Egonet (shown in white) and edges of egonet (shown in gray), where $H = 1$ and $e_m^i = 7$; (d) External neighboring nodes (shown in gray) with external edges of the egonet (shown in gray), where $H = 1$, $e_m^o = 6$ and $n_m^o = 5$.

$$c = \frac{|Triplet_c(m)|}{|Triplet_o(m)| + |Triplet_c(m)|} \tag{3.13}$$

where c is the clustering coefficient and $|Triplet_c(m)|$ is the number of closed triplets formed from node m and $|Triplet_o(m)|$ is the number of open triplets formed from node m. A triplet in this case refers to a structure created from a source node, m,

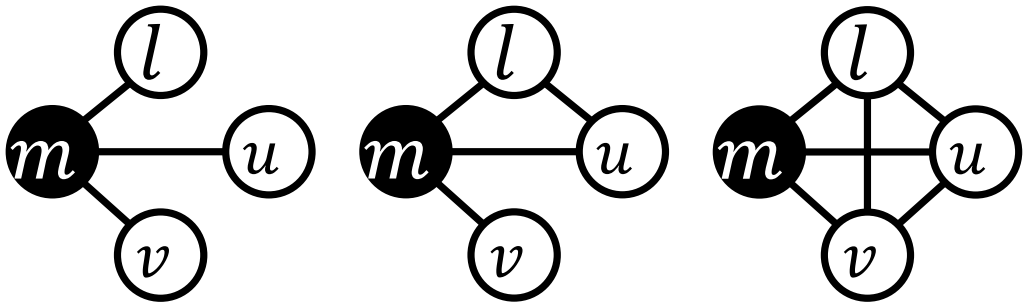


Figure 3.4: Clustering Coefficient c: (a) $c = 0$; (b) $c = \frac{1}{3}$; (c) $c = 1$.

being connected to two additional nodes such as l, and u, as depicted in Fig. 3.4. The triplet is considered open if l and u are not connected, as in Fig. 3.4a, and it is considered closed if l and u are connected, as in Fig. 3.4b. Additionally, node m forms triplets with not only (m, l, u) but also (m, l, v) and (m, u, v). Fig. 3.4c depicts all three triplets as closed, and therefore $c = 1$; while Fig. 3.4a depicts all three triplets as open, resulting in $c = 0$. However, due to the sparseness and directional nature of circuits, a closed triplet is rarely encountered. To illustrate, a closed circuit triplet would take on the form shown in Fig. 3.5, which is unusual in typical netlists. It is, therefore, safe to assume that almost all triplets in a circuit are open, and as a result, their clustering coefficients are nearly zero. Thus, using this feature to describe circuit structure yields no discriminating information.

Contrarily, instead of measuring density with the clustering coefficient c, the convergence coefficients cc^o (3.4), and cc^i (3.5) allow the measurement of density in a sparse and directed network. This is accomplished through the parameterization of the egonet size, as controlled by the number of hops (H), to increase the search space. Utilizing this measurement approach, it is important to consider that if H is chosen

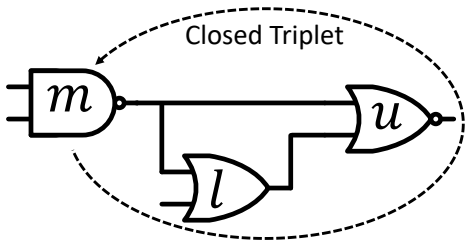


Figure 3.5: Circuit structure with non-zero clustering coefficient.

too small then cc^o and cc^i become zero, and if H is large then the time it takes to generate cc^o and cc^i will be prohibitively long, as the data generation time is exponential to H.

To measure the amount of discriminatory information obtained from both convergence coefficients, description vectors (3.1) are generated from a set of ISCAS'85 benchmarks, in which the vectors only contain the mean, median, standard deviation, skew, and kurtosis of the cc^o and cc^i features. The variance is then calculated from the description vectors as in (3.14),

$$Var(DV_z) = \frac{\sum_i^N (x_i - \mu)^2}{N} \tag{3.14}$$

where Var is the variance of circuit z's description vector DV_z, which contains the 10 elements associated with cc^o and cc^i (mean, median, standard deviation, skew, and kurtosis of both cc^o and cc^i). x_i is an element in DV_z, μ is the mean value of DV_z, and N is the number of elements in DV_z. Note that the amount of variance is directly proportional to the quantity of information obtained from the cc^o and cc^i features. Fig. 3.6 shows how scaling H affects both the total variance and the time it takes to generate the data. Note that a variance of zero, which coincides when

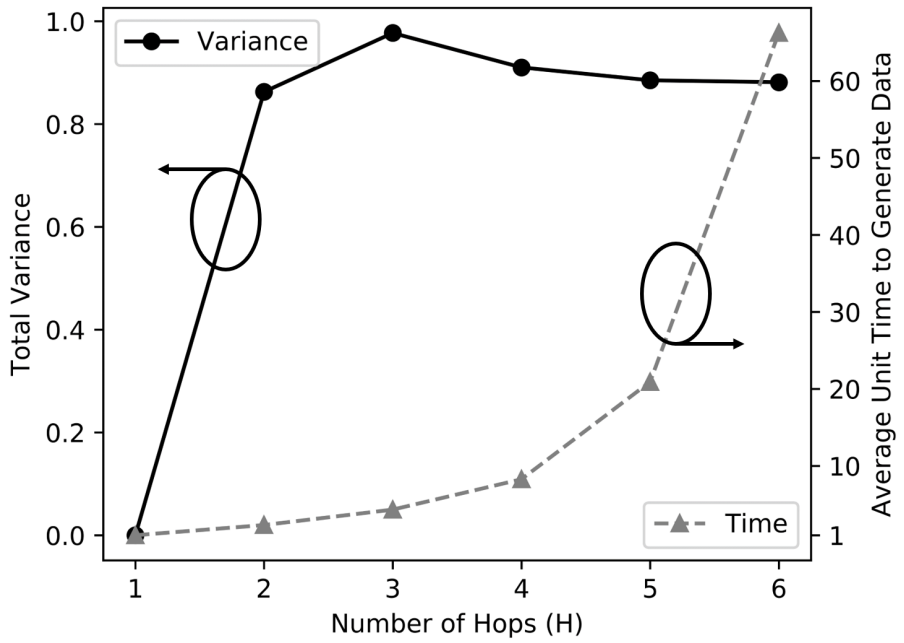


Figure 3.6: Total variance and time to generate data from description vectors containing only the convergence coefficients versus the size of the egonet.

$H = 1$, indicates that all of the description vectors were identical, and the features provide no useful information. From Fig. 3.6, it is determined that the optimal size of the egonet coincides with $H = 3$, due to achieving the maximum variance, while still maintaining reasonable computational efficiency.

To determine the extent obfuscation techniques alter a given circuit's structure, several obfuscated ISCAS'85 benchmarks are utilized from Trust Hub [6]. For each benchmark type, there are four locked versions and a non-locked version referred to as the golden (G) circuit. Note that the four obfuscated versions are selected based on their popularity and the use of an XOR key-based insertion technique. Furthermore, three of the techniques, namely, Logic Cone Size (CS) [31], CYclic Logic Locking (CY) [37], and Secure Logic Locking (SLL) [46], are chosen because they are

specifically designed to protect against SAT attacks. In addition, a fourth RaNdom (RN) technique [33] is added because it distributes locking randomly throughout the circuit; however, it does not protect from SAT solvers. The benchmark circuits are further described in Table 3.3, which lists the circuit type, obfuscation type, gate count, and the obfuscation percentage or the ratio of key-gates to the overall number of gates in the non-locked golden design.

3.2 Results and Discussion

To visualize the description vectors in a 2D space, Principal Component Analysis (PCA) is performed [17], [19], through the use of Singular Value Decomposition (SVD) [40]. PCA reduces the 55 possibly correlated components, which make up the description vectors, into a decreased set of uncorrelated principal components ordered from the largest variance to the smallest. Figs. 3.7a and 3.7b show the 1st vs. 2nd and 3rd vs. 4th Principle Components, which represent 87.2% of the total variance. The benchmarks are also plotted with corresponding 95% confidence ellipses around the circuit clusters, where a cluster consists of the G, CS, CY, SLL, and RN versions of one benchmark design. The goal of this step is to verify that the features provide enough discriminatory information to visually separate the different types of circuits in the 2D space. In addition, this process also assesses whether LL performs its intended goal of removing the similarity between locked circuits and their golden source. As demonstrated in the figures, there is a clear and distinct separation between the different circuit types, therefore, the extracted features do contain enough discriminatory information about each design. However, the clear separation only applies between circuits of different types and not between locked circuits of the same

Table 3.3: Logic locked ISCAS'85 Benchmarks with obfuscation type, number of gates, and obfuscation percentage

Test Article / Design Type	Obfuscation	Number of Gates	Obfuscation Percentage
C1908	G	224	0.0%
	CS	235	3.4%
SEC/DED	CY	236	3.4%
	RN	238	3.4%
	SLL	236	3.4%
C2670	G	370	0.0%
	CS	380	2.2%
12-Bit ALU	CY	381	2.2%
and Controller	RN	381	2.2%
	SLL	383	2.2%
C3540	G	739	0.0%
	CS	785	4.3%
8Bit ALU	CY	786	4.3%
	RN	786	4.3%
	SLL	787	4.3%
C5315	G	1125	0.0%
	CS	1221	5.7%
9-Bit ALU	CY	1219	5.7%
	RN	1227	5.7%
	SLL	1221	5.7%
C6288	G	2252	0.0%
	CS	2351	2.8%
16x16 MULT	CY	2355	2.8%
	RN	2338	2.8%
	SLL	2351	2.8%
C7552	G	1059	0.0%
	CS	1154	6.0%
32-Bit Adder	CY	1157	6.0%
and Comp	RN	1154	6.0%
	SLL	1156	6.0%

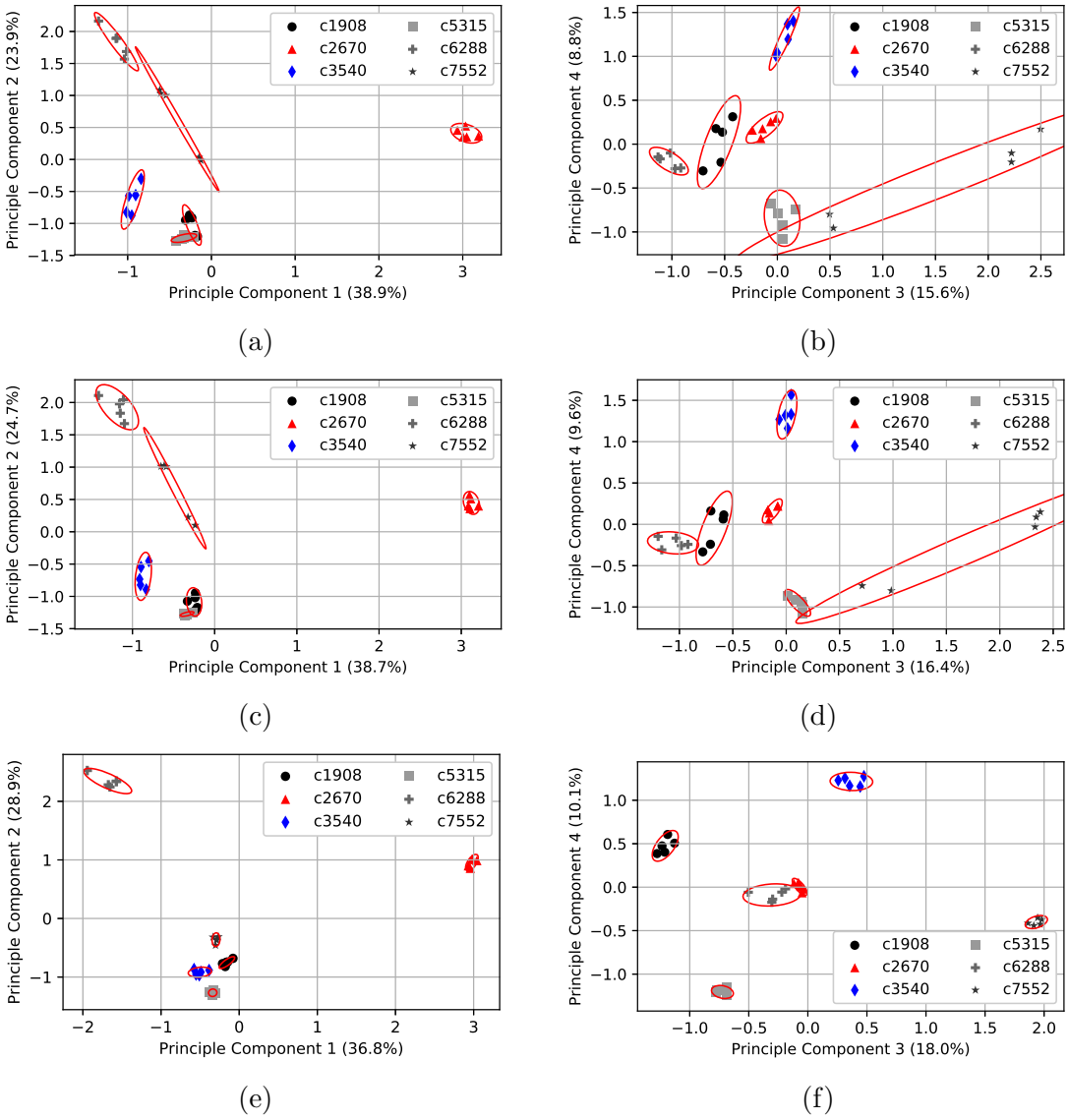


Figure 3.7: Benchmark circuit PCA results and 95% confidence ellipses: (a) 1st vs 2nd components of unmodified circuit; (b) 3rd vs 4th components of unmodified circuit; (c) 1st vs 2nd component w/o key-gates; (d) 3rd vs 4th component w/o key-gates; (e) 1st vs 2nd component w/o key-gates and inverters; (f) 3rd vs 4th component w/o key-gates and inverters.

type. As an example, in Fig. 3.7a, all versions of circuit c2670 form a tight, distinct cluster. This indicates that LL is unable to enact structural change and therefore enables an attacker to identify circuits through structure alone. Ideally, LL would cause increased separation between circuits and thus prevent the creation of visually identifiable clusters. At the same time, however, there is still some overlap that exists between the ellipses, and not all the design types form the same easily identifiable clusters as in circuit c2670. For instance, circuit c7552 has a large variance to where it may be misidentified as a different benchmark type. Therefore, to test how well the circuits can be identified, a k-means clustering algorithm [26] is performed to cluster the groups based on their relative proximity in the PCA space. Note that the k-means clustering algorithm groups the circuits in a way that minimizes the inertia of the system, as calculated in (3.15)

$$inertia = \sum_{i=1}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{3.15}$$

where n is the number of circuits x, C is the set of clusters, and μ_j is the centroid of the respective cluster. The algorithm works by grouping the circuits into clusters, where the center of each cluster is the centroid and the algorithm's objective is to minimize the inertia. In this case, inertia is the summation of the distances of circuits to their respective cluster's centroid, and inertia is minimized by optimal circuit grouping. Note that a lower inertia indicates circuits in the same cluster are more similar. Based on this algorithm, the inertia calculation and the grouping results of the unaltered benchmarks are displayed in the k-means run 1 of Table 3.4a. As shown, two of the obfuscated c7552 circuits are grouped in the incorrect cluster. In addition to the inertia of the k-means clusters, the silhouette score is also utilized as

Table 3.4: K-Means Clustering Results

K-Means Run 1: No Gates Removed

		c1908	c2670	c3540	c5315	c6288	c7552
cluster	1	5	0	0	0	0	2
	2	0	5	0	0	0	0
	3	0	0	5	0	0	0
	4	0	0	0	5	0	0
	5	0	0	0	0	5	0
	6	0	0	0	0	0	3
Total Inertia		8.87	Silhouette Score				0.717

(a)

K-Means Run 2: Key-Gates Removed

		c1908	c2670	c3540	c5315	c6288	c7552
cluster	1	5	0	0	0	0	2
	2	0	5	0	0	0	0
	3	0	0	5	0	0	0
	4	0	0	0	5	0	0
	5	0	0	0	0	5	0
	6	0	0	0	0	0	3
Total Inertia		8.36	Silhouette Score				0.763

(b)

K-Means Run 3: Key-Gates and Inverters Removed

		c1908	c2670	c3540	c5315	c6288	c7552
cluster	1	5	0	0	0	0	0
	2	0	5	0	0	0	0
	3	0	0	5	0	0	0
	4	0	0	0	5	0	0
	5	0	0	0	0	5	0
	6	0	0	0	0	0	5
Total Inertia		2.13	Silhouette Score				0.890

(c)

a metric to quantify how well the circuits can be distinguished. The silhouette score is the mean silhouette coefficient $s(i)$, which is defined by the intra-cluster distance $a(i)$ and nearest inter-cluster distance $b(i)$ for circuit i, as outlined in (3.16) [35].

$$\begin{aligned} a(i) &= \frac{1}{|C_1|-1} \sum_{j \in C_1, i \neq j} d(i,j) \\ b(i) &= \min_{k \neq i} \frac{1}{|C_2|} \sum_{p \in C_2} d(i,p) \\ \text{s(i)} &= \begin{cases} \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}, & \text{if } |C_1| > 1 \\ 0, & \text{if } |C_1| = 1 \end{cases} \end{aligned} \tag{3.16}$$

where there are two circuit clusters, C_1 and C_2. Circuits i and j are both in cluster C_1 and the distance between them is $d(i,j)$. Additionally, $d(i,p)$ is the distance between circuits i and p, where circuit p is in cluster C_2. The silhouette coefficient is calculated for all circuits and the silhouette score is the mean of these results. The score represents a measure of the inter-cluster and intra-cluster similarity, and can range from -1 to +1. Note that a high, positive score indicates circuits within the same cluster are largely similar, while also being dissimilar to circuits from other clusters.

3.2.1 Netlist Modification

These initial clustering results support the hypothesis that LL does not affect the structure of circuits significantly enough to separate them from their original designs. However, there is enough difference created in the case where the c7552 circuit is mis-grouped by the k-means clustering algorithm. Despite this result, LL has not necessarily provided irreversible structural distancing to an adversary. Rather, some structural change can be attributed to artifacts of the LL process and do not require the specific key values to rectify. These artifacts include the XOR/XNOR

key-gates and the additional inserted inverters. Therefore, an attacker can return a locked circuit to the closest resemblance of its non-locked counterpart if the following modifications are made.

Key-Gates

All the LL types in the test circuits use an XOR-based key-gate insertion, where the placement is controlled by the locking method of the particular circuit. It is known that the inserted key-gates are direct artifacts of LL, and therefore are not a part of the original design. Additionally, key-gates are easily located, due to being directly connected to a key input. Returning the circuit to the closest resemblance of its original structure, without solving the key, can be done by removing the XOR key-gates. The removal process is performed by tying the input and output nets of the XOR gate together, while the key-gate and the key input are removed from the network. To measure the impact of key-gate removal on the inertia and the silhouette score, PCA is performed again, and the results are displayed in Figs. 3.7c and 3.7d. The 95% confidence ellipses have decreased in size for both figures, indicating a reduction of the variance within clusters. This is confirmed by the decrease in inertia for run 2 and the minor improvement of the silhouette score (Table 3.4b). These results all indicate that it is now easier for an attacker to correctly cluster the circuit types and consequently defeat the LL. Nevertheless, there are still LL artifacts present in the designs, and the k-means clustering algorithm continues to mis-group the c7552 circuit. Therefore, the next step in reverting structure is to remove the inverters inserted by the XOR-based LL, however, determining if an inverter is intrinsic to the design or added is not as straightforward as removing XOR key-gates.

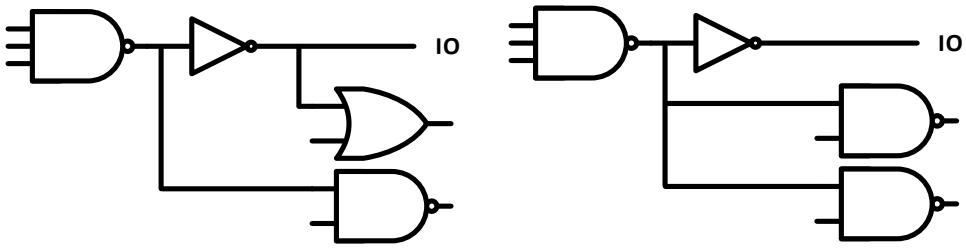


Figure 3.8: Identical circuits with minor structural change due to inverter insertion: (a) Original structure; (b) Structure after re-synthesis. It should be noted that the nearby key-gate is not shown.

Inverters

As detailed in Chapter 2, to implement XOR-based LL, inverters must also be used to allow both key values. A '0' key indicates no additional inverter is inserted, while a '1' key indicates an inverter is placed just before or after the XOR key gate. To hide the added inverter, the design is re-synthesized and the inverter or the inversion bubble is moved around depending on the design optimization, Additionally, the key gates can cause minor structural changes to nearby nets after the re-synthesis process. Fig. 3.8 shows the same section of a design before and after the re-synthesis process, where an identical circuit function is represented with a different structure. While the structure is not directly tied to a key-gate, the presence of a key-gate nearby causes a slight optimization difference, which alters the structure through the inverter placement. Without the key, it is unknown whether a particular inverter is intrinsic to the design or added in support of LL. Instead of attempting to infer this, the proposed solution is to remove all inverters in the designs.

Again, to assess how this modification affects an attackers ability to correctly identify the circuits, PCA is performed and the principal components are depicted in Figs. 3.7e and 3.7f. As shown in the figures, the 95% confidence ellipses have significantly decreased in size, indicating a further reduction in inertia. Additionally, the 1st and 2nd principal components in Fig. 3.7e show a tighter correlation for the c1908, c3540, c5315, and c7552 circuits. At first glance, this indicates a decline in the inter-cluster variance; however, this is negated by the 3rd and 4th principal components in Fig. 3.7f having very large inter-cluster variance with distinct separation. To validate this visual result, the k-means clusters, inertia, and the silhouette score are recalculated and the results are listed in run 3 of Table 3.4c. With the removal of inverters, the k-means clustering algorithm now correctly labels all of the circuits. In addition, the total inertia is reduced by 74.5% compared to only removing key-gates. This indicates that there is significantly less variance within the clusters, which is also displayed in the reduced size of the 95% confidence ellipses. Moreover, there is an increase in the silhouette score, indicating a larger difference between the intra-cluster and inter-cluster similarity. Overall, with the removal of inverters, the results indicate a large reduction in the difficulty of identifying a circuit through its structural similarity.

3.3 Structural Similarity Metric

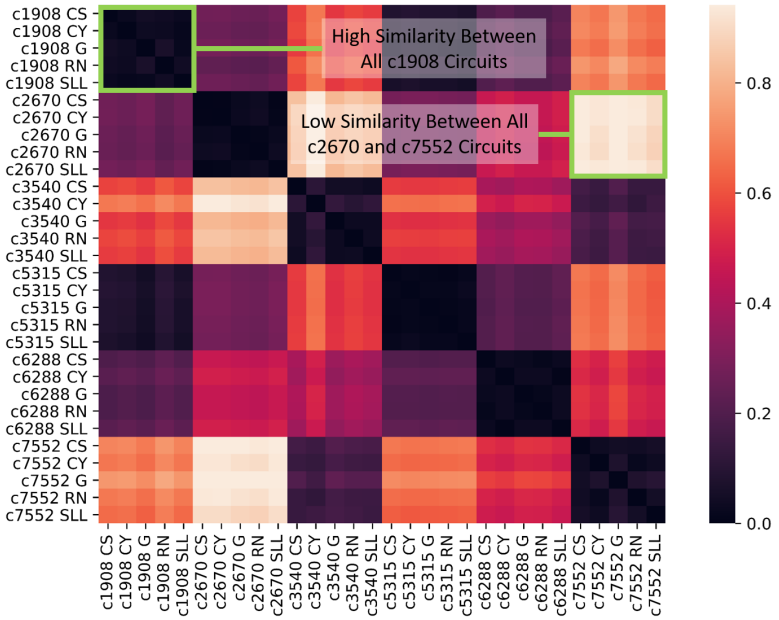
While description vectors provide the basis for identifying circuit clusters, there still exists a need for a metric to determine the similarity between two circuits. This allows an obfuscated circuit to be measured against its golden source to indicate if enough structural change has been implemented to hide its functionality from the

Table 3.5: Common Distances Formulas

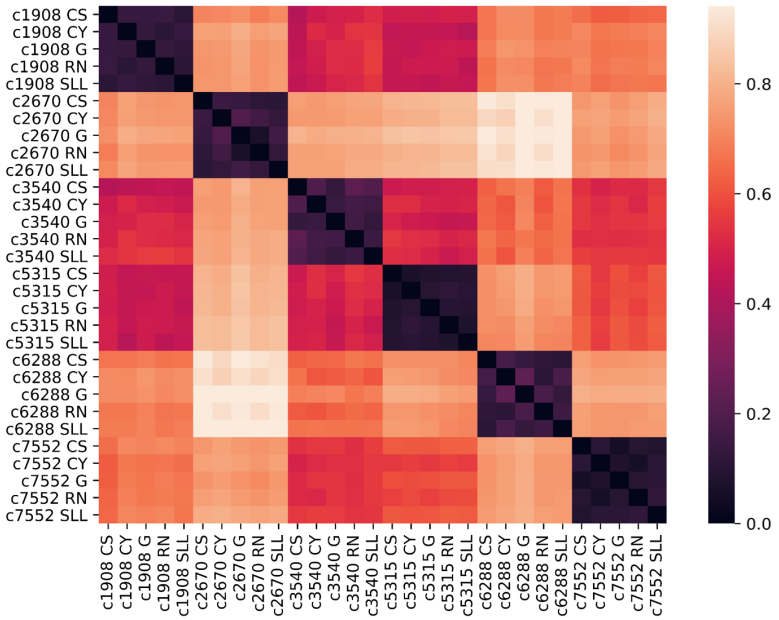
Canberra	$\sum_{i=1}^{n} \frac{P_i - Q_i}{P_i + Q_i}$
Braycurtis	$\frac{\sum_i^n \lvert P_i - Q_i \rvert}{\sum_i^n \lvert P_i + Q_i \rvert}$
Chebyshev	$\max_{i \in n} P_i - Q_i$
Cityblock	$\sum_i^n \lvert P_i - Q_i \rvert$
Cosine	$\frac{P \cdot Q}{\lvert\lvert P \rvert\rvert_2 \lvert\lvert Q \rvert\rvert_2}$
Euclidean	$\sqrt{\sum_{i=1}^{n} \lvert (Q_i - P_i) \rvert^2}$
Hamming	$\sum_{i=1}^{n} \delta(Q_i, P_i)$ $\delta(Q_i, P_i) = 0$ if $Q_i = P_i$; 1 if $Q_i \neq P_i$
Squared Euclidean	$\sum_{i=1}^{n} \lvert (Q_i - P_i) \rvert^2$

original circuit. In addition, this will provide an estimate of the level of effort required for an attacker to break the inserted LL. Typically, this type of measurement is accomplished with a distance formula, which calculates the difference between two multi-element description vectors and condenses the information into a single number. Ideally, the distance formula should equate to a small distance between graphs of the same type and a large distance between graphs of different types. Therefore, the assessment of distance formulas, such as Euclidean, Canberra, Cosine, and City Block, is based on two conditions. First, locked circuits from the same source should have minimal distance from their golden source circuit. Second, locked circuits derived from different sources should have maximum distance. For example, the c6288 benchmark circuit should have minimal distance from its locked versions, while also having maximal distance from the c1908, c2670, c3540, c5315, and c7552 benchmarks. Some common distance formulas between generic n-dimensional description vectors P and Q are listed in Table 3.5.

The utilization of a distance formula allows for the difference between the multi-dimensional description vectors to be represented as a single value. Therefore, the difference between the 55-element description vectors, which occupy a 55-dimensional space, can be described in a human readable format and visualized through a heatmap. Fig. 3.9a depicts a heatmap of the more common Euclidean Distance, while Fig. 3.9b depicts the heatmap of the Canberra distance. Compared to the Euclidean distance, the Canberra distance shows increased separation between benchmark circuits of different types at the cost of increasing the distance between locked versions of the same circuit. However, the difference between the average intra-distance (distance between circuits of the same group, e.g. c1908_CS to c1908_SLL) to the average inter-distance (distance between circuits of different groups, e.g. c1908_CS to c2670_RN) is increased, providing a greater separation between benchmark circuit types. As an example, the average normalized Euclidean intra-distance is 0.011 and 0.019 for benchmarks c5315 and c6288, respectively; while their average normalized Canberra intra-distance is 0.072 and 0.121 respectively. While the Euclidean distance yields marginally better results for intra benchmark comparisons, it yields notably inferior results for the average inter-distance. For example, the inter-distance between benchmarks c5315 and c6288 is 0.216 for Euclidean and 0.737 for Canberra. To this end, the Canberra distance heatmap (Fig. 3.9b) provides a visual identification of benchmark circuit types, and further illustrates two important elements of structural similarity. First, all of the locked circuits derived from the same source are similar to one another, as can be visually verified from the figure. This indicates that not only are locked circuits identifiable from a non-locked source but also from a circuit with a completely different locking scheme. Second, circuits from different sources are



(a)



(b)

Figure 3.9: Distance scores for various distance formulas. A higher score indicates more similarity within circuits derived from the same source and less similarity to circuits from other sources.

distinctly dissimilar. This indicates that there is enough information and resolution to prevent misidentification of locked circuits.

At first glance, the Canberra distance is superior to the Euclidean distance for this analysis. However, to quantitatively establish the ideal distance formula d, various metrics (from table 3.5), are evaluated based on the aforementioned conditions of minimal intra-cluster distance and maximum inter-cluster distance, using (3.17)

$$S(d) = \sum_{i} (mean(\sum_{j,j\neq i} \sum_{l} d(i, j_l)) - mean(\sum_{l} d(i, i_l))) \tag{3.17}$$

where $S(d)$ is the difference between the average inter-cluster and average intra-cluster separation of distance formula d; i and j are non-locked circuits; i_l and j_l are locked circuits i and j, respectively, with LL technique l. Additionally, a higher score, $S(d)$, indicates the distance formula is more adept at separating circuit types. Accordingly, the distance formula scores are shown in Fig. 3.10. Notably, the Canberra distance has the highest inter-cluster to intra-cluster difference of 3.4. This indicates large similarities between non-locked to locked versions of the same circuit, while also maintaining low similarity between different circuits. Conversely, the Hamming distance score is the worst, indicating low similarity between non-locked to locked circuits of the same type and high similarity between circuits of different types. Considering that the Hamming distance is designed for binary vector values, any disagreement between description vector elements, large or small, results in the same distance. Therefore, its poor score is an artifact of slight value variations, resulting in unnecessarily large distances. Likewise, the Euclidean distance also demonstrates an inadequate score value of 1.98, which places it fourth overall. This is a result of the Euclidean distance calculating the absolute distance between description vectors, assuming each element in the vectors are on the same scale. However, each circuit feature will have vastly

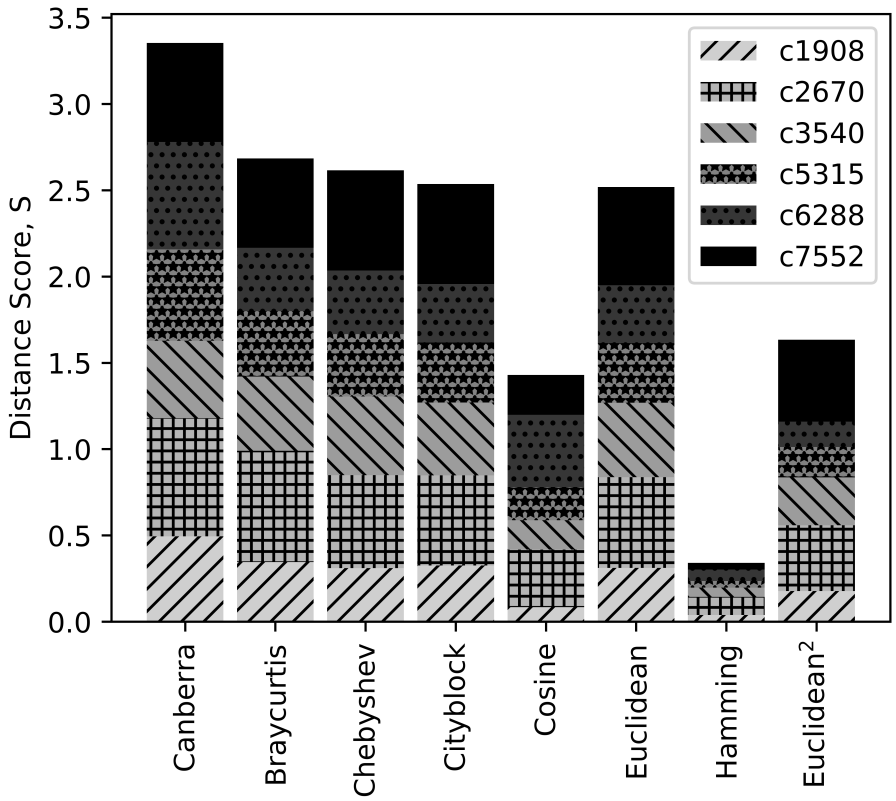


Figure 3.10: Distance scores for various distance formulas. A higher score indicates more similarity within circuits derived from the same source and less similarity to circuits from other sources.

different minimum and maximum values, which causes the Euclidean distance to assign disproportional weighting to features with large absolute values. Conversely, the Canberra distance includes a normalization function to account for the variations in value ranges, and as a result, the Canberra distance yields the best outcome.

The above analysis has shown that the Canberra distance is the most adept formula for this application. The next step is then to assess if any particular type of LL scheme is more favorable in increasing the Canberra distance. To compare the

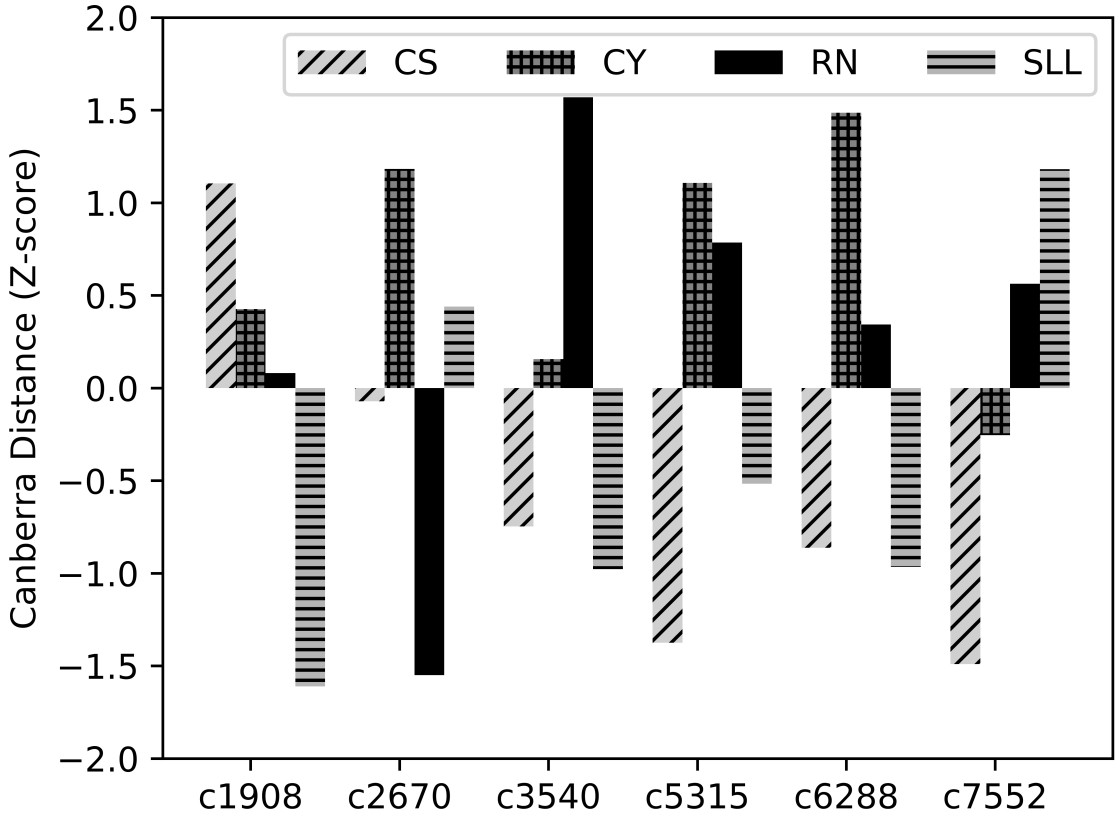


Figure 3.11: Z-score Canberra distance of logic locked benchmark circuits to golden source.

four LL techniques in regards to the average amount of structural change, the Canberra distance between the golden version of each benchmark circuit and the CS, CY, RN, and SLL versions are calculated and the Z-score results are shown in Fig. 3.11. Note that a positive Z-score indicates a LL scheme performs better than the average for a particular benchmark circuit while a negative score indicates it performs worse than average. From the results, none of the LL methods show a significant advantage in terms of structural distance, as no locking scheme has the highest Z-score for all benchmarks. While some locking techniques generate a reasonable structural

distance for certain circuits, they have almost no impact on others. For example, RN LL performs the best on circuit c3540, but performs the worst on c2670. Similarly, SLL performs the best on c7552, but performs the worst on c1908. Conversely, CY displays the most structural change with the largest distance on three of the five circuits but performs worse than average on c7552. It is worth noting that both SLL and CS have worse than average distance on four of the five circuits, indicating they are poor choices for structural obfuscation. In general, no locking method has a distinct advantage, since every method has at least one circuit where it performs worse than the average.

3.3.1 Impact of Key Size

As shown in the previous section, LL provides minimal structural change between locked and non-locked benchmarks derived from the same circuit. The next step is to study the relationship between the key size, which equates to percentage of locking, and structural difference. To assess this effect, the ISCAS'85 c5315 benchmark is implemented with locking types RN, SLL and CS with varying key size bits/obfuscation percentage of 32/2.8%, 64/5.7%, 128/11.4%, and 256/22.8% for each locking method. The measured Canberra distance versus key size is plotted in Fig. 3.12. As shown in the figure, for RN, SLL, and CS circuits, the distance is not always correlated with the key size. For example, the distance of the 64-bit key is less than the 32-bit key for RN LL. Also, CS and SLL maintain nearly identical distances with 64-bit and 128-bit keys. Moreover, the jump from 128-bits to 256-bits increases CS distance by over 20%, while only increasing SLL distance by 3%.

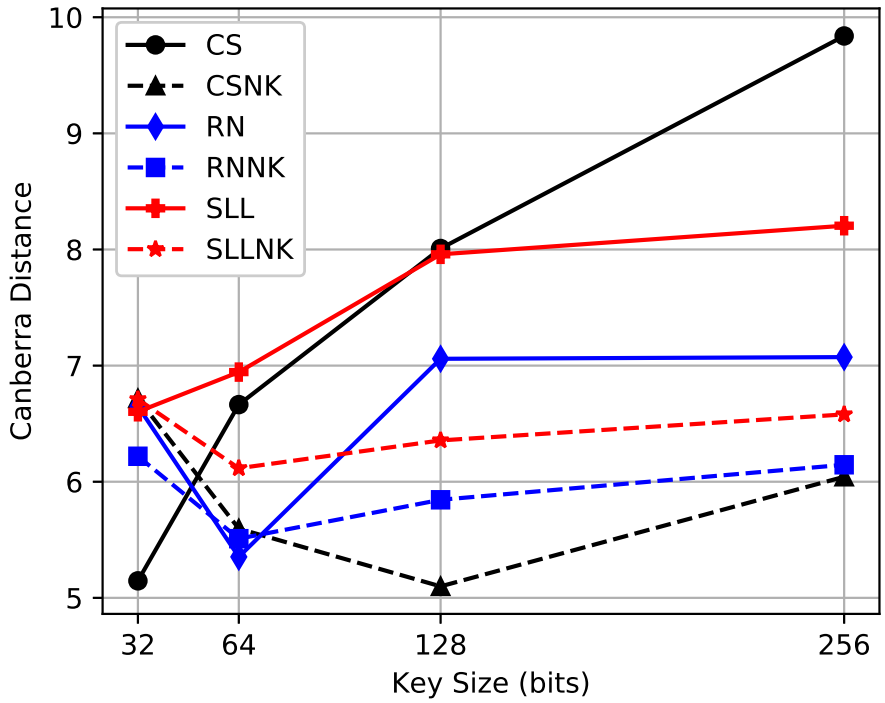


Figure 3.12: Canberra distance of logic locked c5315 benchmarks from the golden source circuit.

Some of these results can be explained by the process in which the different sized benchmarks are generated. For example, the 64-bit and the 32-bit RN LL circuits are generated independently from one another. Consequently, the locations of the key-gates differ between the 64-bit and 32-bit RN LL circuits. Furthermore, the amount of structural change per inserted XOR gate is dependent on the particular net. Considering the location of each XOR gate is random, the amount of structural change per insertion is also random. Additionally, increasing the key size is only meant to alter the functionality, and structural change is only an unintended by-product. Therefore, the amount of structural change per key-gate is effectively random, which in this case results in the decrease in structural change when comparing the 64-bit

and 32-bit RN LL. Nevertheless, all three locking methods show a correlation between increased structural change and increased key size, which is particularly evident for the 256-bit keys.

Referring back to how the locked circuits are generated, XOR gates are inserted and the circuit is re-synthesized to hide the possible net inversions. However, the insertion of these additional XOR gates not only results in changes to the circuit functionality, but also contributes additional delays on the nets where they reside. As a result, during re-synthesis, structural modifications occur to compensate for the increased path delays. However, this change is minimized with smaller key sizes, as shown in Fig. 3.12, where the 32-bit and 64-bit keys have less structural change than the 128-bit and 256-bit keys for CS, RN, and SLL LL types. Despite the netlist modifications of removing inverters and key-gates, the 256-bit key, corresponding to 23% of the circuit size, still ultimately results in significant changes to net delays and consequently, noticeable structural change after re-synthesis.

3.3.2 Additional Netlist Modifications

The results for CS, RN, and SLL LL indicate that the vulnerability of structural similarity can be thwarted by sufficiently increasing the number of key-gates. However, the required number of key-gates becomes a significant portion of the design. Additionally, the structural modifications are not necessarily caused by the functional changes, but the optimizations during the re-synthesis process. In order to reduce the effects of synthesis optimizations, a no key (NK) version of each locked circuit is created, where the NK versions are generated through the removal of the XOR/XNOR key-gates and subsequently re-synthesizing the designs. The re-synthesized circuits

are denoted as CSNK, RNNK, and SLLNK, for locked circuits CS, RN, and SLL, respectively. It should be noted that the key bits in the NK circuits are not solved, as only removing key-gates will leave the inversion logic intact. The associated Canberra distance for the NK circuits is then calculated and presented in Fig. 3.12. As shown in the figure, removing the key-gates and re-synthesizing the circuits eliminate almost all correlation of key size to structural distance. Notably, the structural distance for the largest key size (256-bit) is less than the smallest key size (32-bit), for all NK circuits. Therefore, these results show that the structural similarity vulnerability cannot be thwarted by increasing the key size. Moreover, to reduce this vulnerability, additional methods to incorporate structural change into the locking process must be considered.

3.4 Results Summary and Future Direction

Through this analysis, it has been shown that LL cannot be secured in the same way as encryption, considering that the amount of information contained in a circuit netlist is vastly different than an encrypted data file. The nodes, node types, how they are connected, and their repetition reveal clues to the functionality of a circuit. Moreover, the multi-vector attack space of functional and structural information further increases the difficulty of adequately securing a logic locked circuit. While the functional attack space has been studied extensively through SAT solvers, the structural attack space has yet to be explored, and numerous vulnerabilities do exist. For example, all the separate modules on a single chip may be structurally compared to identify repetition and help solve key values. Additionally, a library of common circuit components can be created and described through their structure. Due to

the efficiency of the matching process, this library of components can be compared against every module within a locked design, uncovering the functionality of some of the circuit, and thereby aiding in unlocking the design. In other words, even if an individual module is secure, the vast amount of available information to an attacker can assist in its identification. Therefore, the current attack model needs to be updated to include attacker access to not only the obfuscated netlist and oracle, but also all the information and repetition from the entire chip. Notably, as LL benchmark circuits get larger, the difficulty of functional attacks will increase but information for structural attacks will become more accessible. Additionally, the structural similarity distance metric of a locked circuit to its non-locked source is only one piece of the puzzle. There also needs to exist a way to correlate that metric back to the security of a device. While there is likely a positive correlation between structural distance and obfuscation strength, the relationship between the probability that an attacker can decipher the locked circuit and the structural distance is unknown at this time. To address this, future research can analyze the correlation of structural divergence and the level of effort required to recover the original circuit. Additionally, methods to disrupt design recovery through structure should also be studied, where the desired effect is to widen the confidence ellipses and cause greater overlap between design types. As a result, this would significantly increase the difficulty of design recovery, and in turn increase the effectiveness of LL.

Chapter 4: Security Framework

4.1 Information Leakage

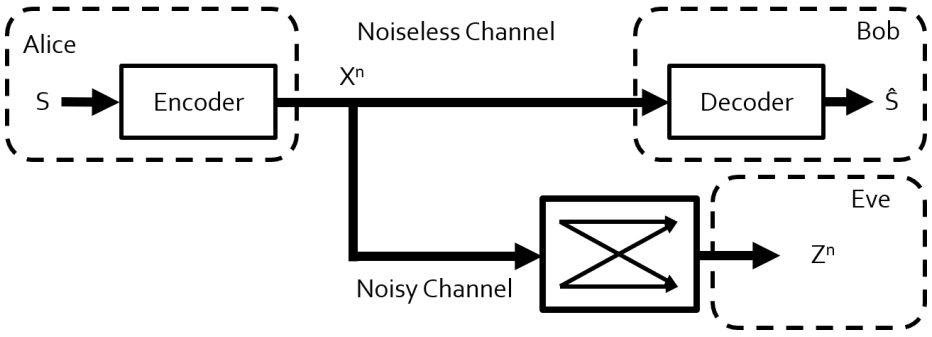
A systematic method to extract information from a logic locked netlist was detailed in Chapter 2. Additionally, by using sub-circuit pattern similarity, the attack is successful in predicting key bits without the use of an oracle. However, the key bit prediction is not necessarily a reflection on the amount of information leaked from the circuit as key bits are predicted based on whichever key value has a closer sub-circuit match to another node's signature in the remaining design. It is entirely possible that the key bit can be predicted incorrectly regardless of if there is a nearly exact sub-circuit. This is not surprising, given that GRAPPLLE uses a simple solving algorithm where the predicted key-bit is determined based on the best matching signature, the key-gate type, and the netlist from which the key-signature is from. Conversely, some intelligence could be brought into the decision on why a certain signatures matches to make a more informed prediction. Additionally, this method was discussed from an attackers perspective, and described how many key bits it predicted, how many it didn't know, and how many it got wrong, but not from a pure information perspective. What is important here is not necessarily that the algorithm predicted the right bits but what an attacker can do with the information.

Namely, if the wrong bit is predicted, is there still enough information that an attacker could determine the correct bit from the matching sub-circuit?

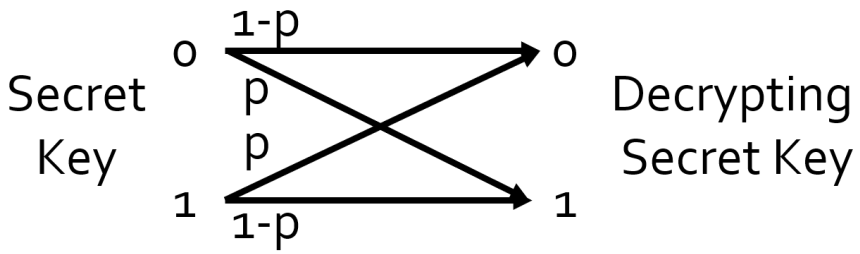
Furthermore, in Chapter 3 it was shown that additional information can be obtained by comparing the global structure of a logic locked design to other modules and circuits within a chip or to a library of sample circuits. This method was able to overcome the structural changes inflicted from LL. Together with GRAPPLLE, these techniques demonstrate the phenomenon of information leakage and how logic locked circuits are a treasure trove of clues which are freely available to an adversary. Moreover, it is through this measurement of information leakage, that resilience to de-obfuscation can be assessed. However, SAT solvers, GRAPPLLE, and NetSimile are only small pieces of the information leakage puzzle and to truly asses the security of LL, many facets of attacks and analysis techniques must fit together into a singular framework. The two previously detailed approaches are ways to measure information leakage, but they are by no means a comprehensive solution.

4.2 Information Leakage of a Generic Logic Locked Chip

Understanding information leakage and how it effects the security of logic locking on a fundamental level can be illustrated through the use of a generic circuit, henceforth referred to as GENC. GENC uses an XOR based locking scheme and the placement of key gates is done randomly. Furthermore, since the placement is random, GENC is also assumed to have uncorrelated key bits. Considering these assumptions, each bit can be represented as a binary symmetric channel inside a binary wire tap model, as shown in Fig. 4.1a. In this classical model the secret message is replaced with the LL key. The actors in this example are as follows. Alice is the IC



(a)



(b)

Figure 4.1: Classical information theory example: (a) Wiretap model where Alice is sending data to Bob and Eve is observing on a noisy channel represented as; (b) Binary symmetric channel.

designer. Bob is the intended recipient of the IC. Eve is an attacker who is trying to decode each bit of the key by observing it through a noisy channel. On account of the key bits being uncorrelated, the probability of Eve intercepting a single bit can be represented as Fig. 4.1b, where p is the cross over probability. A p of zero indicates Eve has a 100% probability of determining the correct transmitted key bit. A p of one indicates that Eve has a 100% probability of determining the key bit is the inverse of the transmitted key. In either case, a cross over probability of zero or one is non-ideal because it indicates Eve has enough information to decide with absolute certainty the key value. Even if that prediction is wrong 100% of the time, as when

p is one, the channel can be flipped so it becomes always correct. Therefore, both extreme cases, where p is zero or one, can be modeled as if Eve is observing the key from a noiseless channel, and consequently, can steal the key with absolute certainty.

Contrarily, in an ideal scenario for a defender, the channel is perfectly noisy, leading to a cross over probability of 0.5 for each key bit. In other words, Eve can make no determination on whether the key bit is a 0 or a 1, which equates to a coin flip. This scenario is ideal because it indicates there is no information leakage to Eve.

To capture this model into a metric in which to asses the vulnerability of a locked circuit, the channel capacity of Eve's wiretap can be combined with the key length to derive the effective key length (EKL). Key length is a common metric to measure the security of encryption, since it equates to how long it would take an adversary to break the encryption through a brute force attack. Moreover, at one time, LL was thought to be similar to encryption and thus used brute force as the defense metric. Additionally, all LL types have a key or configuration bits, where for XOR based LL, the key is simply the key bits. Furthermore, in more complex LL techniques such as LUT based, the key is the programming bits. For example, a 4-input LUT would correspond to a 4-bit key. The key size however is not the EKL. For instance, the Data Encryption Standard (DES) has a key length of 64-bits. However, eight of those bits are used for parity which means the EKL is only 56-bits. Therefore, the security against a brute force attack is only 2^{56} and not 2^{64}.

Accordingly, LL can be assessed in the same fashion, as there is the actual key length, which is determined by the number of key gates, or the number of bits needed to program the LL. Then there is the EKL which is determined by the number of keys, and the amount of information an attacker can obtain for each key or the channel

capacity the attacker has on that key, and thus the probability in which the keys can be predicted. This effective key length EKL can be computed as in (4.1),

$$EKL = K - \sum_{n}^{K} C_n \tag{4.1}$$

where n is a key bit in a circuit with K keys, and C_n is the channel capacity of the wiretapped channel of key bit n. The channel capacity C_{bsc} is represented by (4.2) and (4.3)

$$C_{bsc} = 1 - H_b(p) \tag{4.2}$$

$$H_b(p) = -plog_2(p) - (1-p)log2(1-p) \tag{4.3}$$

where $H_b(p)$ is the binary entropy function and p is the cross over probability. For example, an ideal 1-bit key with a cross over probability p of 0.5 would yield a channel capacity C_{bsc} of zero as shown in (4.4).

$$C = 1 - [-0.5log_2(0.5) - (1-0.5)log2(1-0.5)] = 0 \tag{4.4}$$

A channel capacity of zero indicates that an attacker gets zero bits of information per transmitted bit or per key bit in this case. Hence, there is enough noise in the system to prevent an attacker from determining the key any better than flipping a coin.

Assuming a single bit channel, the EKL of that 1-bit is then $1 - C_{BSC}$ or the 1-bit key example has an EKL of 1-bit when the channel capacity is 0-bits. However, in a LL system the channel is not perfectly noisy and as demonstrated earlier, certain attacks can help determine key values. Altering the previous example, assume an attack method reveals some information of the locked circuit so the new crossover probability is 0.3, shown in (4.5)

Table 4.1: 4-bit key with corresponding cross over probabilities.

Key Bit	Crossover Probability
1	0.8
2	0.7
3	0.9
4	0.5

$$C = 1 - [-0.3log_2(0.3) - (1 - 0.3)log2(1 - 0.3)] = 0.12 \tag{4.5}$$

where the new channel capacity is 0.12 and thus the EKL is reduced to 0.88 bits. Expanding this to a 4-bit example with crossover probability for each key shown in Table 4.1. Accordingly, the channel capacity of the attacker for each bit is calculated as in (4.6),

$$\begin{aligned} C_1 &= 1 - [-0.8log_2(0.8) - (1 - 0.8)log2(1 - 0.8)] = 0.28 \\ C_2 &= 1 - [-0.7log_2(0.7) - (1 - 0.7)log2(1 - 0.7)] = 0.12 \\ C_3 &= 1 - [-0.9log_2(0.9) - (1 - 0.9)log2(1 - 0.9)] = 0.53 \\ C_4 &= 1 - [-0.5log_2(0.5) - (1 - 0.5)log2(1 - 0.5)] = 0 \end{aligned} \tag{4.6}$$

where the attacker is able to receive in total, 0.93 bits of key information. Therefore, the EKL is calculated to be 3.07 bits as shown in 4.7.

$$EKL = 4 - \sum_{n}^{4} C_n = 4 - (0.28 + 0.12 + 0.53 + 0) = 3.07 \text{ bits} \tag{4.7}$$

4.3 Incorporating Logic Locking Attacks into the Framework

Each specific attack contributes some amount of information and consequently a reduction in noise to the wiretap channel. The way this information is recorded

is in the form of the crossover probability, where the more information obtained is correlated to a reduction in crossover probability, from a maximum p of 0.5 to a minimum p of 0. In order to estimate the information contribution from an individual attack and bound the results between 0 and 0.5, a generic equation was created. The equation combines a quality metric which describes the specific attacks efficacy and a confidence metric which describes the how probably the predicted outcome is. The resultant equation is calculated as (4.8),

$$p_i = \frac{1}{2} - \frac{Q_t S_i}{2} \tag{4.8}$$

where p_i is the crossover probability of key bit i, Q_t is the quality factor of the specific attack t, and S_i is the confidence score of bit i. It is the intention that the quality factor Q will be based on an empirical study of how well a specific prediction technique works on the specific LL technique. Additionally, the confidence score S indicates how strongly the analysis technique believes it is correct, where when $S = 1$ indicates 100% confidence and when $S = 0$ indicates no confidence.

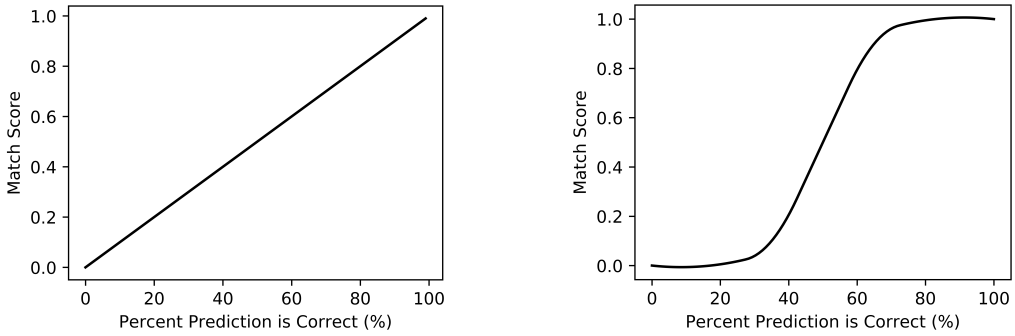
4.3.1 GRAPPLLE Quality and Confidence

Since GRAPPLLE is the only current technique that outputs a probability function we will assume its quality factor $Q = 1$. The confidence score S can be an involved calculation but for simplicity sake, confidence will be determined by (4.9)

$$S_i = \max_{n \in N; n \neq i} (J(i, n)) \tag{4.9}$$

where S_i is the confidence score of key bit i, which is in circuit N, J is the Jaccard index between the sub-circuit associated with i's signature and the sub-circuit associated with n's signature. Additionally, n is a node in circuit N and n and i's

sub-circuits do not overlap. In this scenario, the key bit prediction is ignored, relying only on the match score of the most similar signature even if the prediction is incorrect. Furthermore, for simplicity, it is assumed that a match score follows the probability of guessing the key gate linearly as shown in Fig. 4.2a. However, after a more through analysis, this may not be the exact relationship as predictions could follow other distributions, where Fig. 4.2b shows only one of the many possible distributions. To this end, more study needs to be performed on GRAPPLLE to finalize how S is determined but for this work the linear model is assumed.



(a) (b)

Figure 4.2: Match score versus the accuracy of the key-bit prediction: (a) Linear Distribution; (b) Polynomial Distribution.

4.3.2 Incorporating NetSimile

In contrast to GRAPPLLE, NetSimile does not contribute information to a singular key bit but to the entire circuit. However, this information can be used in conjunction with GRAPPLLE to determine the crossover probability of a key-gate, as shown in 4.10,

$$S_i = \max_{n \in N}(\frac{1}{1 + d_{can}(N, I)} * J(i, n)) \tag{4.10}$$

where S_i is the confidence factor of Key bit i which is in circuit I, J is the Jaccard index between i and n, where n is a node in circuit N, and $d_{can}(N, I)$ is the Canberra distance between circuits N and I. The higher the distance, the lower the Jaccard index is weighted, where the minimum distance of zero weights the Jaccard index the same as if the signatures appeared in the same module. Therefore, to analyze the entire information leakage due to GRAPPLLE involves using NetSimile across all modules in a SoC and a library of circuits. Then those circuits can generate signatures with GRAPPLLE. An example of this process is detailed in section 4.4.

4.4 Security Assessment Example Scenario

A simple example scenario of information leakage and security of a SoC involves a single chip with a simple feed forward circuit. The example chip has 256 inputs pins and 128 output pins. LL has been implemented on the SoC where an off chip memory provides the 128 LL bits, which consist of 128 XOR/XNOR logic gates distributed randomly throughout the design. The first attack utilized against the chip is a SAT solver. The high interconnection count proves too complicated for a SAT solver and it is unable to provide any DIPs as the function times out. In the next approach, the attacker is able to break down the circuit into 364 modules and the LL is distributed amongst modules 0 to 199. The next tool applied is GRAPPLE which the attacker uses independently on each module. However, each module is unable to provide similar enough signatures to the key-nets and the attacker can make no determination of the bits with GRAPPLLE alone. Therefore, information

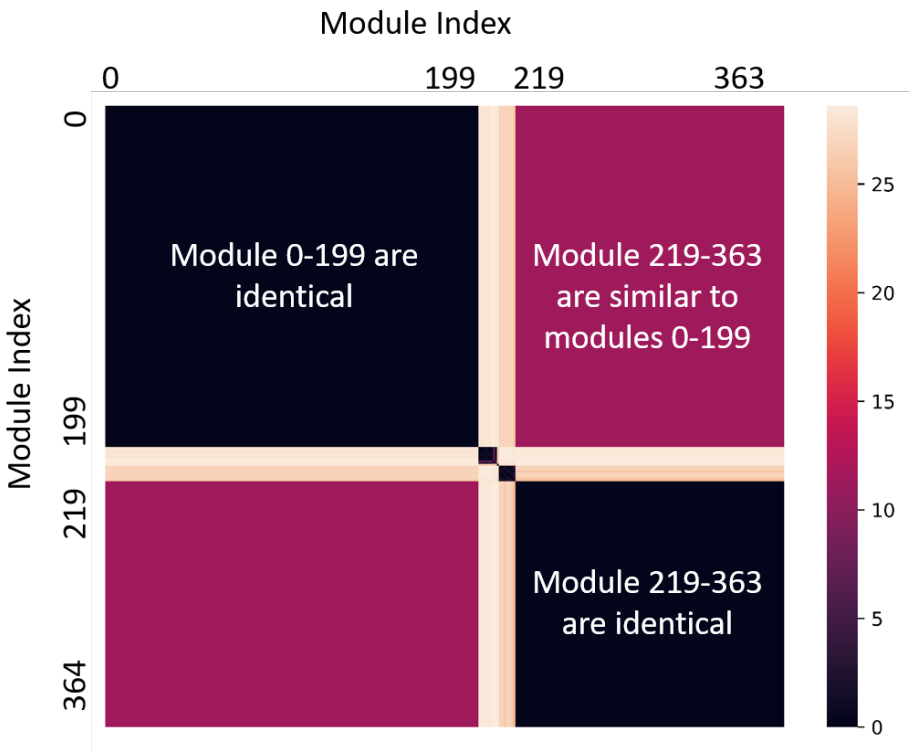


Figure 4.3: Heat map showing Canberra distance of large locked circuit separated into modules where a distance of 0 indicates the modules are identical.

from other modules in the SoC must be utilized to break the LL technique. NetSimile is used to determine if there is enough inter-module similarity to warrant the use of GRAPPLE between modules, where this similarity is shown in the heat map in Fig. 4.3. As shown in the figure, modules 0 through 199 are identical. This region is also where the entirety of the key gates are located. Additionally, the figure illustrates a high similarity between modules 0-199 and 219-262 indicating that these modules are not identical but may be generated from a similar source. Granted that modules 0-199 have identical structures, GRAPPLLE is used again with all the sub-circuits between the 200 modules to predict key values. Consequently, every key bit finds

a nearly identical match in one of the other modules and the resulting key bits, their closest match, and prediction can be found in Table 4.2. Furthermore, the EKL is calculated as the summation of the key length (128 bits) minus the attacker channel capacity C_{bsc} of each bit (117.4 bits), resulting in an effective key length of $EKL = 128 - 117.4 = 10.6$ bits. Thus, the brute force effort to decipher the key has been reduced by over 99.9%.

This example may seem contrived but the analyzed circuit is actually an AES core and the key gates were all located inside the substitution boxes within that core. Hence, this is a very real example where the security level of the circuit actually lessens with increasing size, allowing the attacker to utilize the entire circuit information to solve a locked circuit. Additionally, encryption cores are naturally SAT resistant due to the complex interconnected structure and are even used as Anti-SAT blocks [45], hence functional testing would even deem this circuit secure. Therefore, this example brings credence to the necessity of a holistic approach to LL to accurately determining its security.

Table 4.2: Logic locked circuit key bits with confidence prediction score S and attacker's channel capacity of each bit C_{bsc}.

Key	S	C_{bsc}	Key	S	C_{bsc}	Key	S	C_{bsc}	Key	S	C_{bsc}
0	0.996	0.98	32	0.988	0.95	64	0.988	0.95	96	0.934	0.79
1	0.934	0.79	33	0.978	0.91	65	0.984	0.93	97	0.968	0.88
2	0.968	0.88	34	0.984	0.94	66	0.98	0.92	98	0.996	0.98
3	0.984	0.93	35	0.974	0.90	67	0.934	0.79	99	0.996	0.98
4	0.978	0.91	36	0.98	0.92	68	1.0	1.0	100	0.996	0.98
5	0.968	0.88	37	0.974	0.90	69	1.0	1.0	101	0.984	0.94
6	0.98	0.92	38	0.98	0.92	70	0.996	0.98	102	0.996	0.98
7	0.978	0.91	39	0.974	0.90	71	0.968	0.88	103	0.992	0.96
8	0.992	0.96	40	0.996	0.98	72	0.968	0.88	104	0.968	0.88
9	0.97	0.89	41	0.992	0.96	73	0.998	0.99	105	0.996	0.98
10	0.978	0.91	42	0.996	0.98	74	0.978	0.91	106	0.964	0.87
11	1.0	1.0	43	0.93	0.78	75	0.97	0.89	107	0.961	0.86
12	0.992	0.96	44	0.978	0.91	76	0.992	0.96	108	0.964	0.87
13	0.992	0.96	45	1.0	1.0	77	0.956	0.85	109	0.964	0.87
14	0.988	0.95	46	0.968	0.88	78	0.988	0.95	110	0.961	0.86
15	0.978	0.91	47	0.956	0.85	79	1.0	1.0	111	0.964	0.87
16	0.978	0.91	48	0.978	0.91	80	0.938	0.80	112	0.961	0.86
17	0.964	0.87	49	0.984	0.93	81	0.968	0.88	113	0.984	0.93
18	0.95	0.83	50	1.0	1.0	82	0.988	0.95	114	0.984	0.93
19	0.968	0.88	51	0.978	0.91	83	0.984	0.93	115	0.996	0.98
20	1.0	1.0	52	0.996	0.98	84	0.988	0.95	116	0.984	0.94
21	0.968	0.88	53	0.978	0.91	85	1.0	1.0	117	0.984	0.93
22	0.992	0.96	54	0.978	0.91	86	0.984	0.93	118	0.978	0.91
23	0.978	0.91	55	0.978	0.91	87	0.988	0.95	119	0.984	0.94
24	0.988	0.95	56	0.934	0.79	88	0.968	0.88	120	0.961	0.86
25	0.974	0.90	57	0.992	0.96	89	0.934	0.79	121	0.992	0.96
26	0.978	0.91	58	1.0	1.0	90	0.934	0.79	122	0.992	0.96
27	0.97	0.89	59	0.961	0.86	91	0.988	0.95	123	0.97	0.89
28	0.974	0.90	60	0.974	0.90	92	0.934	0.79	124	0.97	0.89
29	0.984	0.94	61	1.0	1.0	93	0.984	0.94	125	0.98	0.92
30	0.978	0.91	62	0.992	0.96	94	0.98	0.92	126	0.984	0.93
31	0.984	0.94	63	0.988	0.95	95	0.984	0.94	127	0.978	0.91

Chapter 5: Conclusions and Future Work

5.1 Major Contributions

The hardware security research space is filled with different attacks and defenses. Additionally, a new paper is published everyday that circumvents the protection or vulnerability from a previou work. However, what currently does not exist are good metrics to asses the security level of hardware. Specifically, in the logic locking domain, security assessments are relegated to decryption through functional tests and SAT solvers. This approach is severely limited in scope and does not accurately account for the comprehensive attack surfaces present for logic locking. Moreover, non-functional based attacks are limited in their contribution to current metrics as they do not incorporate well with functional attacks. Therefore, up to now, there has not been an effective way to tie the information of all these attack methods into a single comprehensive metric of security.

This work has three major contributions to Logic Locking and hardware security. The first contribution is GRAPPLLE, a novel approach to predict key values in a locked circuit by exploiting the repetition introduced through the synthesis process. The methodology is able to extract sub-circuits and then, using a symmetric simulation algorithm or sequence matching, is able to determine the sub-circuit similarity

with a high degree of accuracy in a scale-able and efficient manner. This information can then be used to predict key values through the comparison of sub-circuits around key gates to other sub-circuits in the same design. Furthermore, GRAPPLLE is able to supply a confidence score for those predictions, which are an estimation of the amount of information leakage due to design repetition.

The second major contribution of this work is the ability to describe circuit structure through an extraction of graphical features from each node in a design. Additionally, we demonstrate that by condensing the feature information into description vectors, the structural similarity of two circuits can be assessed using a distance formula. Furthermore, this analysis technique is minimally affected by LL, as circuits from the same source have considerably smaller distances than circuits from different sources. Moreover, not only does this property remain true regardless of LL being present in the circuits, this property is also not affected by the amount of LL. Finally, none of the tested LL techniques held any advantage in addressing the structural similarity threat space. As a result, this work demonstrates a new attack vector to LL that is not accounted for in the current protection models. Perhaps more importantly, this contribution is able to supply a confidence score in the form of a distance formula indicating circuit similarity, and therefore, how useful the information is to an attacker.

Lastly and perhaps the most significant contribution of this work is the framework in which to asses the security of logic locking. The framework is completely attack agnostic and only requires a confidence and quality factor that any LL attack can supply. While useful, the algorithms and processes that were created to build and demonstrate the framework are simply a means to estimate information which will

be improved and replaced in the future. The intention of creating these algorithms was to simply provide a proof of concept for information leakage and to be able to show how this information can be combined and implemented into a security metric.

Furthermore, the framework is able to estimate the noise reduction based on all the available attacks methods, producing a final metric which is the effective key length. The EKL metric can be equated to a brute force approach and gives a realistic expectation of security. Additionally, the framework is able to evolve over time, given its ability to incorporate new information estimation algorithms and attacks.

5.2 Future Work

5.2.1 GRAPPLLE

While GRAPPLLE proved useful in predicting key gate values, it is by no means perfect. The first potential area of improvement is to be able to detect and predict correlated key gates. This includes generating permutations of multibit keys instead of only inverted and not inverted. Additionally, it would be beneficial to detect how and what effects one key bit has on another, and therefore, not require synthesis permutations if the impact is determined to be insignificant.

Next is increasing the reliability of the prediction. There were quite a few cases where a high similarity sub-circuit provided the wrong prediction. This means key gate prediction is not as simple as using the key gate variant (Inverted or Non-inverted) with the best matching sub-circuit in a design. While the information is present, it might indicate that the correct key is opposite of the best match. Therefore, improvements could be made by analyzing how and why each sub-circuit matches, and possibly, using machine learning algorithms on the data to make the predictions.

Additionally, the data can be further improved by generating several sub-circuit sizes and representations to increase matching possibilities. GRAPPLLE can be further automated and tied into the synthesis tools to enable quicker turnaround on key predictions and assist in generating permutations for correlated key bits to further improve accuracy.

Lastly, the topic of scalability was not directly addressed in this work. While every effort was made to create efficient algorithms, the small benchmark circuits used, do not allow for an accurate study of how well GRAPPLLE can scale to larger designs. However, given how the match process is executed, the run time is expected to be $O(n*k*\log(k))$ where n is the number of gates in a design and k is the number of key bits, assuming all key bits are uncorrelated. This function also assumes that one round of matching is performed per key bit, and that after each successive round, one key-bit is solved, reducing the number of match functions by one for each successive round. Correlated key bits will cause further increases in run times due to the possibility of permutations being required. For large designs, the intention would be to break them into smaller pieces prior to using the GRAPPLLE approach which would allow this algorithm to scale to much larger designs. Additionally, the match process can be completely parallelized to aid in reducing run times.

5.2.2 NetSimile

In Chapter 3, the focus was on combinational logic and specifically the ISCAS'85 benchmarks. The reasoning behind using these circuits was the availability of logic locked versions. Granted that the NetSimile method was indeed able to identify

circuits, the golden models used were an exact representation and the small size allowed the circuits to be analyzed in their entirety at the same time.

To advance this analysis further, several improvements can be made. First, different types of circuits and circuit variations need to be assessed. This includes analyzing a large quantity of designs and determining how each circuit type represents itself. Through this analysis, the features extracted for the description vectors can be tailored to a specific design type and enable the ability to identify circuit class even in the event that a golden representation is not available. This step is crucial to enable library level structural attacks. Next, the ability to break down a circuit into modules is key in assessing SoCs or any moderately complex design. This can be done through various techniques including simple algorithms to separate by registers and by using community detection algorithms to extract clusters. Additionally, physical separation can be used as the circuit layout can indicate obvious cores or circuit partitions. Moreover, when analyzing different classes of circuits, the extracted features need to be reassessed to ensure they are still useful and develop new features.

The modified NetSimile approach was shown to work on XOR based LL but every type of LL technique and cell replacement type will have its own nuances that will need to be taken into account. Take for example the LUT approach to gate replacement. In this case, there is not an insertion of additional gates but a direct replacement. Consequently, the best solution might be to do no modifications to the netlist as there is a one for one replacement and therefore no alterations to the structure. However, depending on the standard cell library, the LUT could replace a few gates and therefore it would be prudent to replace all LUT instantiations with multiple gate placeholders.

5.2.3 Metrics and Frameworks

Lastly, the framework is a promising start but is missing several features. Namely the framework is based on uncorrelated single bit channels. However, this is rarely the case as key bits are often correlated and therefore can't be solved as isolated instances. This will require the use of conditional probability values. Additionally, different estimation techniques can have different correlations between keys bits. For example, K0 and K1 could be correlated for a SAT attack but not for GRAPPLLE. Thus, these scenarios will have to be built into the framework. Lastly, different techniques also need to be included into the framework such as Desynth and SAIL.

5.3 Complexity Assessment

This work originated in the pursuit of assessing the complexity of benchmark circuits and LL. The idea was that the current circuit complexity metric of gate count is too simplistic to capture to nuances of toady's ICs. This idea extends to LL, as the complexity of a circuit can also be correlated to how much security is needed to protect it. One major component of complexity is repetition where the c6288 benchmark circuit is relatively simply, due to its repetitive pattern of a simple sub-circuit. However, this complexity was disconnected from the LL community where functional testing found the c6288 multiplier circuit too difficult for a SAT solver. Our analysis has shown the opposite where the repeatable patterns created exploitable sub-circuits. While the information was used to predict key values, the technique finds its roots in analyzing the complexity of an IC or the predictability of it. The predictability of each sub-graph in an IC can be used as a measurement to assess the Shannon entropy or degree of randomness of that design, and therefore, its complexity.

Increasing the complexity is the end goal, as it correlates to the difficulty of predicting key values or determining design intent. For example, we can assess the complexity of a single core RISC processor versus a dual core RISC processor. Assuming each core is exactly the same; the current complexity metric would rate the dual core processor as double the complexity of the single core. However, the introduction of repetition yields an increase in predictability which actually means the dual core processor is less complex than the single core. Therefore, it is actually harder to implement LL on the dual core processor. This type of analysis is crucial when assessing any LL techniques effectiveness or determining the necessary amount of locking needed.

5.4 Final Thoughts

This work has shown that logic locking is vulnerable from a variety of different attack methods and the community is not accurately taking this into account. This work has made contributions to formalize a standardized metric in which we can accurately asses logic locking and give an accurate prediction of security. Through this metric, we can provide an apples to apples comparison of different techniques where we can rate the efficiency of LL by its key length to EKL ratio. Finally, the results and methods obtained in this work could have a profound impact on the LL community and how circuits are protected in the future.

Bibliography

[1] Imdb datasets. https://www.imdb.com/interfaces/.

[2] Defense science board task force on high performance microchip supply. Technical report, Office of the Under Secretary for Acquisition, Technology, and Logistics, U.S. Department of Defense., December 2005.

[3] software & hardware reverse engineering. https://ghbintellect.com/intellectual-property-consulting-services/engineering-support/reverse-engineering/, 2018.

[4] S. Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, May 2008.

[5] Sarah Amir, Bicky Shakya, Domenic Forte, Mark Tehranipoor, and Swarup Bhunia. Comparative analysis of hardware obfuscation for ip protection. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, GLSVLSI '17, pages 363–368, New York, NY, USA, 2017. ACM.

[6] Sarah Amir, Bicky Shakya, Xiaolin Xu, Yier Jin, Swarup Bhunia, Mark Tehranipoor, and Domenic Forte. Development and evaluation of hardware obfuscation benchmarks. *Journal of Hardware and Systems Security*, 2(2):142–161, Jun 2018.

[7] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. Stealthy dopant-level hardware trojans. In *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'13, pages 197–214, Berlin, Heidelberg, 2013. Springer-Verlag.

[8] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *CoRR*, abs/1209.2684, 2012.

[9] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan. Hardware trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, Aug 2014.

[10] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems,*, pages 1929–1934 vol.3, May 1989.

[11] Prabuddha Chakraborty, Jonathan Cruz, and Swarup Bhunia. SAIL: machine learning guided structural analysis attack on hardware obfuscation. *CoRR*, abs/1809.10743, 2018.

[12] R. S. Chakraborty and S. Bhunia. Harpoon: An obfuscation-based soc design methodology for hardware protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1493–1502, Oct 2009.

[13] Dean Collins. Darpa trust in ic's effort, 2007.

[14] J. Couch, E. Reilly, M. Schuyler, and B. Barrett. Functional block identification in circuit design recovery. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 75–78, May 2016.

[15] Mohamed El Massad, Siddharth Garg, and Mahesh V Tripunitara. Integrated circuit (ic) decamouflaging: Reverse engineering camouflaged ics within minutes. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015*, 2015.

[16] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai. A secure camouflaged threshold voltage defined logic family. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 229–235, May 2016.

[17] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[18] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the iscas-85 benchmarks: a case study in reverse engineering. *IEEE Design Test of Computers*, 16(3):72–80, July 1999.

[19] H Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, page 417–441, Sep 1933.

[20] Anirudh Iyengar and Swaroop Ghosh. Threshold voltage-defined switches for programmable gates. *CoRR*, abs/1512.01581, 2015.

[21] Paul Jaccard. The distribution of the flora in the alpine zone.1. *New Phytologist*, 11(2):37–50, 1912.

[22] H. Mardani Kamali, K. Zamiri Azar, K. Gaj, H. Homayoun, and A. Sasan. Lut-lock: A novel lut-based logic obfuscation for fpga-bitstream and asic-hardware protection. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 405–410, July 2018.

[23] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

[24] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *CoRR*, abs/1801.01207, 2018.

[25] D. Liu, C. Yu, X. Zhang, and D. Holcomb. Oracle-guided incremental sat solving to reverse engineer camouflaged logic circuits. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 433–438, March 2016.

[26] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[27] H. Mardani Kamali, K. Zamiri Azar, K. Gaj, H. Homayoun, and A. Sasan. Lutlock: A novel lut-based logic obfuscation for fpga-bitstream and asic-hardware protection. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 405–410, July 2018.

[28] M. E. Massad, S. Garg, and M. Tripunitara. Reverse engineering camouflaged sequential circuits without scan access. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 33–40, Nov 2017.

[29] Mohamed El Massad, Jun Zhang, Siddharth Garg, and Mahesh V. Tripunitara. Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism. *CoRR*, abs/1703.10187, 2017.

[30] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin. Revisit sequential logic obfuscation: Attacks and defenses. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.

[31] S. Narasimhan, R. S. Chakraborty, and S. Chakraborty. Hardware ip protection during evaluation using embedded sequential trojan. *IEEE Design Test of Computers*, 29(3):70–79, June 2012.

[32] J. Rajendran, O. Sinanoglu, and R. Karri. Is split manufacturing secure? In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1259–1264, March 2013.

[33] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &; Communications Security*, CCS '13, pages 709–720, New York, NY, USA, 2013. ACM.

[34] Jordan Robertson and Michael Riley. The big hack: How china used a tiny chip to infiltrate u.s. companies, 2018.

[35] Peter Rousseeuw. Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65, 11 1987.

[36] J. A. Roy, F. Koushanfar, and I. L. Markov. Epic: Ending piracy of integrated circuits. In *2008 Design, Automation and Test in Europe*, pages 1069–1074, March 2008.

[37] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. Cyclic obfuscation for creating sat-unresolvable circuits. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, GLSVLSI '17, pages 173–178, New York, NY, USA, 2017. ACM.

[38] Alexander Simoes and César Hidalgo. The economic complexity observatory: An analytical tool for understanding the dynamics of economic development, 2011.

[39] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 243–246, New York, NY, USA, 2015. ACM.

[40] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566, 1993.

[41] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 137–143, May 2015.

[42] P. Subramanyan, N. Tsiskaridze, W. Li, A. GascÛn, W. Y. Tan, A. Tiwari, N. Shankar, S. A. Seshia, and S. Malik. Reverse engineering digital circuits using structural and functional analyses. *IEEE Transactions on Emerging Topics in Computing*, 2(1):63–80, March 2014.

[43] Cheng Ting-Fang. Tsmc to invest record $20bn in advanced 3-nm chips, Dec 2017.

[44] Steve Trimberger. Trusted design in fpgas. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 5–8, New York, NY, USA, 2007. ACM.

[45] Y. Xie and A. Srivastava. Anti-sat: Mitigating sat attack on logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2018.

[46] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. On improving the security of logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1411–1424, Sept 2016.