TAL TECH

DOCTORAL THESIS

Advanced Hardware Protection Mechanisms: A Study on Logic Locking and Circuit Obfuscation Techniques

Antonio Felipe Costa de Almeida

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2025

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
42/2025

Advanced Hardware Protection Mechanisms: A Study on Logic Locking and Circuit Obfuscation Techniques

ANTONIO FELIPE COSTA DE ALMEIDA

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

The dissertation was accepted for the defence of the Doctor of Philosophy in Information and Communication Technology degree on 5th June 2025

Supervisor: Professor Dr. Samuel Pagliarini,
Department of Computer Systems, Centre for Hardware Security,
Tallinn University of Technology,
Tallinn, Estonia

Co-supervisor: Dr. Levent Aksoy,
Department of Computer Systems, Centre for Hardware Security,
Tallinn University of Technology,
Tallinn, Estonia

Opponents: Professor Dr. Shahin Tajik,
Department of Electrical and Computer Engineering,
Worcester Polytechnic Institute,
Worcester, United States

Professor Dr. Domenic Forte,
Department of Electrical and Computer Engineering,
University of Florida,
Gainesville, United States

Defence of the thesis: 16th June 2025, Tallinn

Declaration:
Hereby, I declare that this doctoral thesis, my original investigation, and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Antonio Felipe Costa de Almeida _____
signature

European Union
European Regional
Development Fund

Investing
in your future

ISSN 2585–6901 (PDF)
ISBN 978-9916-80-323-3 (PDF)
DOI https://doi.org/10.23658/taltech.42/2025

Costa de Almeida, A. F. (2025). *Advanced Hardware Protection Mechanisms: A Study on Logic Locking and Circuit Obfuscation Techniques* [TalTech Press]. https://doi.org/10.23658/taltech.42/2025

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
42/2025

Täiustatud riistvara kaitsemehhanismid: uuring loogikalukustamise ja hägustamise tehnikate kohta

ANTONIO FELIPE COSTA DE ALMEIDA

TAL TECH
KIRJASTUS

Contents

List of Publications

The present PhD thesis is based on the following publications.

[I] L. Aksoy, Q. -L. Nguyen, F. Almeida, J. Raik, M. -L. Flottes, S. Dupuis, and S. Pagliarini, "Hybrid Protection of Digital FIR Filters," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 6, pp. 812-825, 2023

[II] F. Almeida, L. Aksoy, Q. -L. Nguyen, S. Dupuis, M. -L. Flottes and S. Pagliarini, "Resynthesis-based Attacks Against Logic Locking," in 24th International Symposium on Quality Electronic Design (ISQED), pp. 1-8, 2023

[III] F. Almeida, L. Aksoy, and S. Pagliarini, "RESAA: A Removal and Structural Analysis Attack Against Compound Logic Locking," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 33, no. 3, pp. 1–13, 2025

Other related publications

The author has contributed to other publications during his studies at Tallinn University of Technology.

[IV] L. Aksoy, Q. -L. Nguyen, F. Almeida, J. Raik, M. -L. Flottes, S. Dupuis, and S. Pagliarini, "High-level Intellectual Property Obfuscation via Decoy Constants," in IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 1-7, 2021

[V] M. Imran, F. Almeida, A. Basso, S. S. Roy, and S. Pagliarini, "High-speed SABER key encapsulation mechanism in 65nm CMOS.," in Journal of Cryptographic Engineering, pp. 461–471, 2023

[VI] F. Almeida, M. Imran, J. Raik and S. Pagliarini, "Ransomware Attack as Hardware Trojan: A Feasibility and Demonstration Study," in IEEE Access, vol. 10, pp. 44827-44839, 2022

[VII] F. Almeida, L. Aksoy, J. Raik and S. Pagliarini, "Side-Channel Attacks on Triple Modular Redundancy Schemes," in IEEE 30th Asian Test Symposium (ATS), pp. 79-84, 2021

[VIII] M. Imran, F. Almeida, J. Raik, A. Basso, and S. Pagliarini, "Design Space Exploration of SABER in 65nm ASIC," in Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security (ASHES), pp. 85–90, 2021

Contribution of the Author to the Publications

The author's contributions to the core publications included in this thesis are outlined below:

- **Publication *I*:** As the third author, I contributed specifically to the evaluation of compound logic locking (CLL) schemes. My contributions focused on designing and implementing circuits employing CLL, generating synthesis metrics such as area, delay, and power consumption, and assessing their security properties. Additionally, I participated in attack assessments to measure the effectiveness of the protection mechanisms applied.
- **Publication *II*:** As the first author, I led the research on a resynthesis-based attack against logic locking (LL), managing all stages of the study. This work introduced a methodology for manipulating locked netlists using commercial EDA tools to expose vulnerabilities. I was responsible for designing and implementing the attack framework, conducting experiments, and analyzing results to assess the effectiveness of the proposed approach. Furthermore, I prepared the manuscript, including the presentation of both methodology and results.
- **Publication *III*:** As the first author, I developed the RESAA framework for analyzing and attacking CLL schemes. My work involved designing and implementing a system capable of classifying locked netlists, identifying critical structures, and applying targeted attacks to uncover secret keys. I refined the attack strategies within RESAA, improving their efficiency and increasing their success rate against various protection schemes. Experimental validation demonstrated the effectiveness of RESAA in analyzing LL techniques. I also prepared the manuscript, presenting these findings in a structured and detailed manner.

In addition to these core publications, I have contributed to other research papers published during my PhD studies at Tallinn University of Technology. These works, listed under **Other Related Publications**, cover additional aspects of hardware security, cryptographic implementations, and hardware attack strategies.

Abbreviations

AGR	AppSAT Guided Removal
AI	Artificial Intelligence
AppSAT	Approximate SAT
ATPG	Automatic Test Pattern Generation
BLE	Bilateral Logic Encryption
CAC	Corrupt and Correct
CASLock	Corruptibility and Security Locking
CAVM	Constant Array Vector Multiplication
CG	Critical Gate
CLL	Compound Logic Locking
CNF	Conjunctive Normal Form
CRK	Constant Replacement with Key
DFLT	Double-Flip Logic Locking Technique
DIP	Distinguishing Input Pattern
DSP	Digital Signal Processing
DTL	Diversified Tree Logic
EDA	Electronic Design Automation
Fa-SAT	Fault-Aided SAT
FIR	Finite Impulse Response
FLL	Fault-Based Logic Locking
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GNN	Graph Neural Network
HD	Hamming Distance
HLS	High-Level Synthesis
IC	Integrated Circuit
IoT	Internet of Things
IP	Intellectual Property
KA	Knowledgeable Adversary
LL	Logic Locking
LOOPLock	Logic Optimization-Based Cyclic Logic Locking
LUT	Look-Up Table
MCM	Multiple Constant Multiplication
ML	Machine Learning
MUX	Multiplexer
OA	Oblivious Adversary
OG	Oracle-Guided
OL	Oracle-Less
OoT	Out of Time
PO	Primary Output
PSLL	Provably Secure Logic Locking
PUF	Physically Unclonable Function
QATT	Query Attack

QBF	Quantified Boolean Formula
RLL	Random Logic Locking
RTL	Register Transfer Level
SA	Specific Adversary
SAT	Satisfiability
SCOPE	Synthesis-Based Constant Propagation Attack for Security Evaluation
SFLL	Stripped Functionality Logic Locking
SFLT	Single-Flip Logic Locking Technique
SKG	SAT-Resistant Key Gate
SLL	Strong Logic Locking
SoC	System-on-Chip
TGA	Topology-Guided Attack
TMCM	Time-Multiplexed Constant Multiplication
TTLock	Tenacious and Traceless Logic Locking

List of Figures

List of Tables

1 Introduction

The increasing integration of hardware systems across various industries, including automotive, defense, telecommunication, and beyond, has raised concerns about the security of hardware components. As hardware becomes more complex and interconnected, it faces more significant risks to the security of integrated circuits (ICs), mainly when intellectual property (IP) cores and chips are outsourced for manufacturing in the globalized supply chain [1]. The main concerns include reverse engineering, IC counterfeiting, overproduction, IP piracy, and the insertion of hardware Trojans [2].

Reverse engineering allows adversaries to deconstruct and analyze proprietary designs, uncovering their structure, functionality, and underlying technologies. This process exposes sensitive IP to unauthorized access, industrial espionage, and potential exploitation and facilitates the creation of counterfeit or pirated versions of the original designs, causing financial losses to the original creators [3]. Counterfeiting involves the unauthorized reproduction of proprietary designs, replicating original designs without permission. These imitations often compromise quality and reliability, creating significant security risks and introducing potential vulnerabilities [4]. Overproduction occurs when manufacturers produce quantities exceeding authorized limits, often without the knowledge or consent of the intellectual property owner [5]. IP piracy involves the illegal use of designs to produce unauthorized ICs [2], and hardware Trojans introduce malicious logic that can compromise both functionality and reliability [6].

Various countermeasure techniques have been developed, each offering different levels of protection and trade-offs regarding area, power, and delay overheads. These techniques include split manufacturing, hardware metering, watermarking, and hardware obfuscation, which encompasses logic locking (LL).

In split manufacturing, the metal layers of the IC are divided and fabricated at different foundries to mitigate security risks [7, 8]. Hardware metering involves real-time monitoring of resource usage within the IC to prevent piracy by tracking and regulating the allocation of hardware resources, ensuring secure and efficient utilization [9, 10]. Watermarking allows for the detection of IP theft and misuse by embedding signatures into the design without changing functionality [11, 12].

Hardware obfuscation plays an important role in preventing unauthorized access by modifying circuit architecture, making it significantly more difficult for adversaries to decipher or reverse engineer its functionality. This method effectively hides the correct operation of the circuit, safeguarding it from malicious adversaries [13]. LL is a specific kind of obfuscation technique that uses key-driven gates to ensure that the circuit operates correctly only when the appropriate key is provided [14–19].

Over the years, several LL techniques have been proposed, ranging from simple XOR/XNOR-based designs to more sophisticated approaches incorporating multiple locking strategies for enhanced security. While these methods can successfully obscure circuit functionality against older attacks [20], they remain vulnerable to more advanced threats [21]. As a result, continuous research into more robust and efficient LL techniques is essential for ensuring long-term security. However, LL poses challenges, particularly in balancing security with overhead in hardware complexity in terms of area, delay, and power dissipation. High resource usage can be especially problematic for designs with

low-power requirements, such as Internet of Things (IoT) devices [21]. Maintaining this balance is essential for achieving security and efficiency in resource-limited environments.

One of the most well-known attack methods targeting LL techniques is the Satisfiability (SAT)-based attack, which systematically removes incorrect keys by finding distinguishing input patterns (DIPs) [20]. Introduced in 2015, this attack efficiently reduces the key search space, compromising even advanced LL schemes characterized by a large number of key bits and increased hardware complexity designed to improve resiliency against adversarial attacks [22, 23]. In response, designers have developed SAT-resilient strategies to counter SAT-based attacks, which significantly increase the computational difficulty for such adversaries [16, 24]. Additionally, efforts have been made to address other emerging threats, such as structural analysis and removal attacks, which exploit different vulnerabilities in locking mechanisms [25, 26]. However, as adversaries continue to refine and combine these techniques, securing hardware remains a dynamic and evolving challenge.

Effectively addressing SAT-based attacks requires a clear understanding of the limitations of current defenses. While SAT-resilient techniques, such as Anti-SAT [27] and SARLock [16], have been proven to be effective against SAT-based attacks, they suffer from other challenges when integrated into complex designs due to their hardware complexity overhead and limited output corruption [28]. A combination of LL techniques, adding an extra layer of protection, has been explored. Methods like compound logic locking (CLL) have been developed to combine high output corruption with SAT-resilient mechanisms, increasing the difficulty of key recovery for attackers [29, 30]. However, these enhanced security measures can significantly impact the design's complexity, increasing area, power, and delay, which may pose additional challenges [31].

Attacks on LL are generally divided into oracle-guided (OG) and oracle-less (OL) approaches. In addition to the locked netlist, OG attacks leverage a functional IC as an oracle to compare inputs and outputs, systematically deducing the secret key [20, 32]. SAT-based attacks are examples of this kind of attack and are effective in this context. In contrast, OL attacks assume that the attacker has access only to the locked netlist and no functional IC, making the key extraction process more challenging but still feasible through methods such as *resynthesis-based attacks*, which leverage electronic design automation (EDA) tools to resynthesize the design based on various key guesses, aiming to converge toward the correct solution [33] or to generate functionally equivalent versions of the locked netlist and analyze them for vulnerabilities [34].

As OG and OL attacks become more sophisticated, defenses must evolve to account for both the structural weaknesses in designs and the tools attackers may use. This ongoing cat-and-mouse game between attackers and designers drives the continued advancement of security measures.

1.1 Contribution of this Thesis

This thesis is a compilation of three published papers, as shown in Fig. 1. Each paper addresses specific research questions and challenges related to LL and its vulnerabilities, contributing to a deeper understanding of attack strategies and countermeasures.

The TVLSI 2023 paper investigates the following research question: Can an attacker

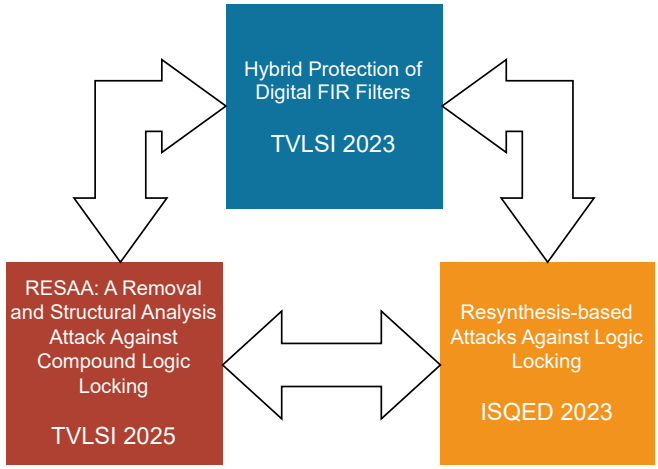


Figure 1: Three foundational works of this thesis.

extract the secret key from an obfuscated finite impulse response (FIR) filter despite existing obfuscation techniques? The paper introduces a query attack that strategically selects input queries to deduce key bits while bypassing current defenses. The results demonstrate that traditional obfuscation schemes fail against this attack, revealing critical security gaps. Furthermore, the paper proposes a hybrid defense strategy that combines hardware obfuscation with LL, enhancing security without significantly impacting implementation constraints [35].

The ISQED 2023 paper questions a common assumption in LL research by asking: How do the synthesis EDA tools impact the security of LL circuits? This study introduces the resynthesis-based attack, showing that transformations introduced during synthesis can weaken LL protections. By generating multiple structurally different but functionally equivalent versions of a locked circuit using EDA tools, attackers can significantly amplify the effectiveness of existing attacks to recover the secret key. The results demonstrate that even advanced LL techniques remain vulnerable, emphasizing the necessity of resilient LL approaches across different synthesis parameters [34].

The TVLSI 2025 paper addresses the research question: Does CLL improve security against attacks or introduce new vulnerabilities? The paper presents RESAA, a framework designed to systematically analyze CLL designs, identify weak points, and execute targeted attacks under both OL and OG models. By partitioning the CLL circuit, RESAA successfully exploits inherent weaknesses in multi-layer LL techniques, improving attack success rates. Experimental results reveal that RESAA can break a wide range of CLL variants, demonstrating fundamental limitations in CLL security and highlighting the need for stronger protection mechanisms [30].

Through these contributions, this thesis advances the field of hardware security by systematically analyzing and exposing weaknesses in both traditional and hybrid LL techniques. The proposed attack methodologies and defensive strategies provide valuable information on the evolving landscape of hardware security, helping shape the development of more resilient countermeasures against LL attacks.

1.2 Outline of this Thesis

The remainder of this thesis is organized as follows:

- **Section 2: Background** – This section provides an overview of hardware security, focusing on essential concepts, such as the IC supply chain and its associated challenges, hardware obfuscation, LL, and various attack models. It establishes a foundation by explaining the core mechanisms and challenges in developing secure ICs.
- **Section 3: Discussions** – This section explores the interrelation between the three papers, covering defensive and offensive hardware security strategies. The first paper introduces a **query attack**, a novel technique capable of breaking obfuscated FIR filters by identifying their hidden coefficients. To counter this, it also presents a **hybrid defense approach** that combines hardware obfuscation with LL, leveraging decoy obfuscation and point functions to enhance security while maintaining competitive hardware complexity. The second paper proposes a **resynthesis-based attack**, which systematically manipulates locked netlists using commercial EDA tools to expose their vulnerabilities. By generating multiple functionally equivalent but structurally different versions of a circuit, this approach reveals weaknesses that remain undetected by traditional attacks, significantly increasing the number of deciphered key bits. The third paper introduces **RESAA**, a framework designed to analyze and attack CLL circuits. RESAA classifies locked netlists, identifies critical gates (CG), and applies structural analysis to expose secret keys. Together, these works contribute to a deeper understanding of both attack methodologies and resilient defense strategies in LL.
- **Section 4: Conclusion and Future Work** – This section summarizes the key contributions from each study and discusses their broader impact on hardware security. It also outlines potential directions for future research, such as enhancing protection techniques, exploring more advanced attack models, and incorporating artificial intelligence (AI)-driven tools to strengthen IC security further.

2 Background

2.1 IC Supply Chain and Security Challenges

Figure 2 presents a generic design flow for ICs composed of many stages. The process begins with **specification and behavioral design**, where the IC's functionality and performance requirements are defined. This step is managed by the design house, a trusted entity that establishes the intended behavior, focusing only on meeting the specifications and design goals. Although this phase involves minimal participation from external parties, there are still potential threats. For example, inside threats or mismanagement of sensitive design data could lead to unintentional leaks or targeted theft. Additionally, adversaries may analyze early design knowledge to identify potential weaknesses in the later stages of the design process [36].

Strict access control policies and secure data management protocols should be implemented to mitigate these risks, especially as cloud-based IC design platforms become more widely adopted [37]. The use of watermarking can also help identify unauthorized use of design data [38]. Regular audits and training for design teams can further minimize the likelihood of insider threats.

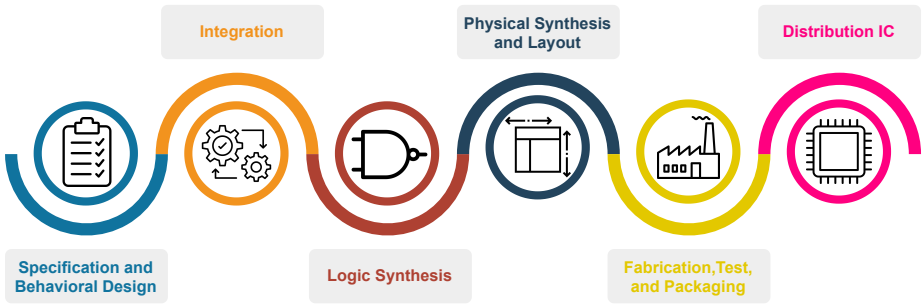


Figure 2: Generic IC design flow.

In the **integration** phase, often done by a contracted team, various IP blocks are assembled into a unified system-on-chip (SoC). Integration poses heightened risks because untrusted entities may gain visibility over the entire chip, enabling them to alter or manipulate the top-level design. An attacker involved in integration could insert hidden vulnerabilities or introduce malicious components. In addition, untrusted IP providers or external design teams could embed subtle backdoors to facilitate reverse engineering or compromise the system's integrity [39].

Verification procedures, such as IP-level integrity checks and top-level validation against tampering, are crucial to mitigate these threats during the integration [40]. Utilizing hardware obfuscation techniques such as LL ensures that sensitive components remain protected even if exposed during integration [41, 42]. Furthermore, employing trusted design environments and ensuring that all IPs are vetted and certified reduces the likelihood of adversarial modifications [43]. Trusted engineers at the design house must closely monitor the integration process to ensure that no unauthorized modifications occur.

In the **logic synthesis** phase, the high-level design is mapped into a gate-level

netlist representing the circuit regarding logic gates and connections. Also, it is the responsibility of the design house, which may involve using integrated third-party IP cores. Unverified third-party IPs integrated during this phase could pose significant risks, including unauthorized data leakage and compromised functionality.

Rigorous verification processes, including checks for malicious alterations and validation of the netlist against design specifications, help ensure the integrity of the output. Hardware security techniques, such as the IEEE P1735 standard for netlist encryption [44], or logic obfuscation, can also deter adversarial exploitation [45]. Collaboration with certified IP providers and maintaining control over the entire synthesis process further mitigates risks and upholds the design's security.

The netlist is converted into a physical representation during **physical synthesis and layout**, from floorplan to place and route information within the IC layout. This phase often involves external parties that may not be fully trusted, especially when outsourcing the physical implementation to third-party companies. Here, untrusted layout engineers or third-party contractors gain access to a more detailed view of the design. An adversary with access to this phase could attempt to reverse-engineer parts of the layout or insert malicious modifications, mainly if they are familiar with layout tools and techniques [46].

In **fabrication, test, and packaging**, the fabrication involves creating the physical IC from its design and translating the layout into silicon. Testing ensures that the fabricated chip functions as intended, identifying defects and failures, and in the packaging step, the chip is enclosed in a protective casing to interface with external systems. These steps, often outsourced to external foundries and facilities, introduce risks as untrusted entities gain access to critical design information. At this stage, adversaries can leverage complete visibility of the chip layout to attempt reverse engineering, necessitating countermeasures such as obfuscation utilized in previous steps to prevent unauthorized duplication or tampering [47].

Finally, the chip reaches the market in **distribution IC**, gets integrated into end-user products, or is deployed for specific applications. Even at this stage, security threats persist, including counterfeiting and tampering during distribution. Adversaries may attempt to duplicate the product, compromise its functionality, or extract proprietary information through reverse engineering [48].

Manufacturers can mitigate these risks by employing anti-counterfeiting measures such as unique identifiers, secure boot mechanisms, and hardware-based authentication protocols [49–51]. Supply chain monitoring and traceability systems ensure that only authorized and secure ICs reach the end-user. In summary, each phase of the IC design flow presents distinct security challenges, requiring tailored countermeasures to enhance the integrity and resilience of the final product.

Figure 3 shows the most common integration of LL techniques to mitigate security threats. LL is introduced by embedding security features directly into the IC's design as part of the **Logic Locking** process. Applying LL early in the flow safeguards critical design components from tampering or reverse engineering attempts, ensuring that the IC remains non-functional until the correct activation key is provided. This approach effectively protects the design's intellectual property and functionality.

Upon completion of manufacturing, the **key activation** phase unlocks the IC's

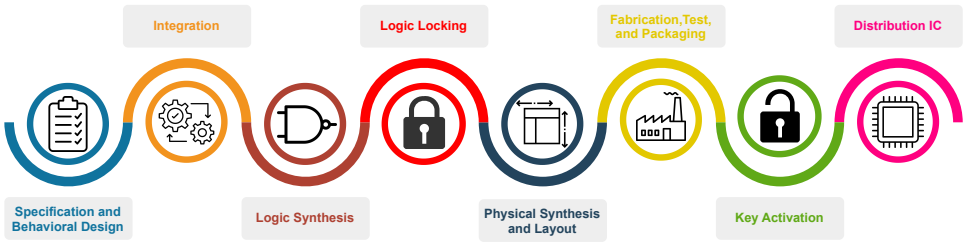


Figure 3: Locking locking in the IC design flow.

functionality. This step is typically managed by the same design house that initiated the design flow and involves securely provisioning the activation (secret) key to the locked design. After fabrication, the locked circuit undergoes the key activation phase, where the secret key is securely stored in a tamper-proof memory within the chip. Ensuring that the key remains protected is critical because if it is compromised, untrusted entities could deactivate or duplicate the IC, rendering the LL measures ineffective.

Once key activation is complete, the IC enters the market as a fully **functional IC**. However, it still faces threats such as counterfeiting, cloning, and reverse engineering, particularly from adversaries with physical access to the chip. These adversaries may exploit invasive or semi-invasive methods to extract sensitive design information. LL mitigates these risks by ensuring that any unauthorized replication or tampering fails without the secret key, preserving the IC's integrity and protecting its intellectual property.

Trusted teams oversee the design, synthesis, and secure key activation processes. At the same time, untrusted actors—including external IP vendors, third-party tool providers, offshore foundries, and unauthorized market participants—pose substantial risks throughout the supply chain. LL is a fundamental defense mechanism, extending protection from the initial design stages to the IC's market deployment. The diversity of adversaries, from insiders to external attackers, underscores the importance of embedding strong security measures at every stage. The following subsection thoroughly discusses hardware obfuscation techniques, emphasizing LL as a key strategy for enhancing IC security.

2.2 Hardware Obfuscation and Logic Locking

2.2.1 Hardware Obfuscation

Hardware obfuscation is a security technique developed to protect the internal design of an IC by making it challenging to analyze, reverse-engineer, and manipulate. By modifying the design to obscure its correct functionality from unauthorized users, obfuscation aims to make it computationally infeasible for adversaries to extract meaningful information, thereby securing IP and preventing malicious alterations [14, 52, 53].

Obfuscation methods can be applied at various stages of the design flow:

- **High-level Obfuscation**: At the high-level design stage, obfuscation targets critical components such as proprietary algorithms, data paths, or constants. This is achieved by introducing decoy elements or misleading logic to obscure sensitive

functions, making it difficult for adversaries to understand the system's intent. Using high-level synthesis (HLS) tools to embed obfuscation directly into the design ensures that sensitive portions' intent remains protected throughout the flow [54–56].

- **Behavioral-Level Obfuscation**: At the register-transfer level (RTL), obfuscation involves altering the control and data flow of the design [57]. Techniques include obscuring finite state machines (FSMs) by modifying state transitions, introducing additional states, and leveraging reconfigurable key-based FSMs [32]. This makes it harder to reverse-engineer the control logic, which explores the interplay between security and functionality in behavioral-level obfuscation [58].

- **Gate-Level Obfuscation**: After logic synthesis, gate-level obfuscation is applied to modify the logical structure of the design. This includes inserting additional gates, modifying or removing existing gates, or embedding techniques such as LL. These modifications obscure the circuit's functionality, requiring a secret key for correct operation, protecting the IP from reverse engineering [15, 16, 28, 59].

- **Layout-Level Obfuscation**: At the physical design stage, layout-level obfuscation or camouflaging is used to protect the physical representation of the IC. This technique involves designing the layout so that different logic functions appear identical, making it challenging for attackers to identify the actual functionality of each component. For example, using standard-cell libraries with indistinguishable layouts or introducing decoy components can significantly enhance protection against physical reverse engineering [25, 60].

By embedding obfuscation techniques throughout the design flow—from high-level to layout-level—hardware designers can ensure robust protection against various adversaries, securing IP and mitigating threats effectively. The following subsection covers the details of LL and its application and efficacy.

2.2.2 LL: A Form of Hardware Obfuscation

LL is a specialized form of hardware obfuscation that works by inserting key-driven gates into the design. These gates ensure that the circuit behaves correctly only when the correct key is applied. Without the secret key, the circuit produces incorrect outputs or remains non-functional, protecting the IP from reverse engineering and unauthorized use [61].

The evolution of LL has seen several variations aimed at improving both security and efficiency. After the introduction of random logic locking (RLL) using XOR/XNOR gates [14], subsequent research expanded to explore different types of key gates, such as AND/OR gates, multiplexors, and look-up tables, while considering the hardware complexity of these gates introduced into the locked circuit [17].

Despite these advances, the original defenses were eventually compromised by the development of the SAT-based attack [20]. Provably secure logic locking (PSLL) was introduced as a paradigm offering formal security guarantees against known attack methods, incorporating point functions that limit the number of incorrect keys DIPs

can eliminate. Recall that a DIP is an input vector that produces different outputs on the locked netlist with two different keys, allowing an attacker to refine the correct key iteratively. By restricting the effectiveness of DIPs, PSLL forces attackers to explore an exponential number of them, making key recovery infeasible [27, 62–65].

In this context, traditional LL techniques can be categorized into two main groups: single-flip locking technique (SFLT) and double-flip locking technique (DFLT). Figure 4(a) presents an SFLT, which relies on a single critical point in the circuit that corrupts an output under a specific input pattern. Although SFLTs demonstrate resilience against SAT-based attacks, they are vulnerable to removal attacks, where an attacker can identify and eliminate the critical point, separating the design into the original netlist and locking unit [25, 66, 67]. On the other hand, in Figure 4(c), DFLTs improve security by introducing two critical points: one in the perturb unit, which initially corrupts an output, and another in the restore unit, which corrects the output when the correct key is applied. While this approach enhances security, DFLTs remain susceptible to advanced structural attacks that exploit the interconnections between the perturb and restore units and their integration with the original circuit [24, 33, 68, 69].

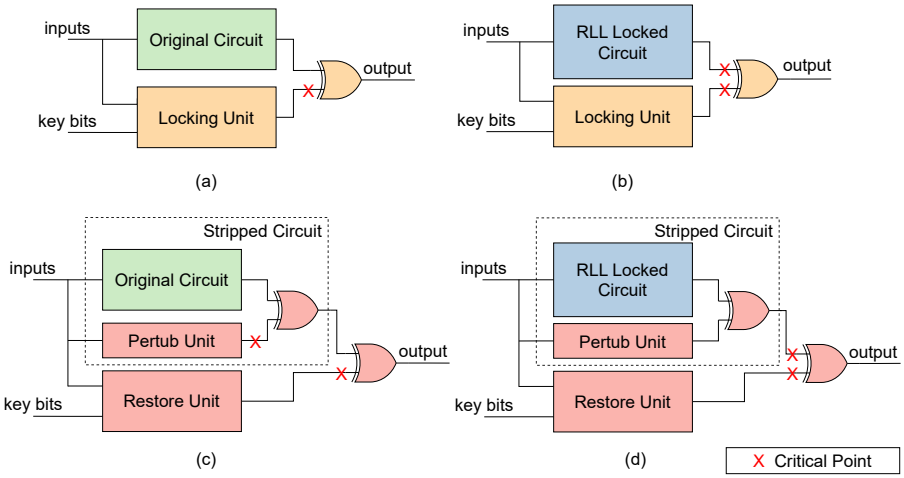


Figure 4: High-level architecture of (a) SFLT, (b) RLL + SFLT, (c) DFLT, and (d) RLL + DFLT in a CLL scheme. This figure is reproduced from Figure 2 in Publication III.

Efforts to strengthen LL techniques have taken various directions, aiming to overcome their perceived weaknesses. These approaches include the insertion of cyclic logic [70], the use of emerging materials [71], and look-up table (LUT)-based obfuscation [56, 72]. Each method introduces additional complexity to the locking mechanism, making it more challenging for adversaries to bypass the protection. However, no single method has been proven entirely secure against all attacks, which has led to the development of more advanced strategies.

Despite the potential of CLL, research into attacks specifically targeting this hybrid technique remains limited, and the exploration of such attacks has only begun to scratch the surface. For example, the combined use of SAT-based and structural attacks against CLL has been studied [73, 74], but these studies are confined to specific combinations of techniques. This underscores the broader need for more comprehensive

research to understand and fully mitigate the vulnerabilities in CLL designs. One of the main contributions of this thesis is that it introduces a novel attack strategy that reveals previously neglected weaknesses in CLL, offering critical insights into its security limitations.

Figures 4(b) and 4(d) exemplify CLL, which integrates double-layer LL techniques to improve the security of ICs. Note that RLL is always used as it delivers the critical feature of (high) output corruption. By combining RLL with other techniques, CLL strives to take advantage of their respective strengths while mitigating individual weaknesses. This combined strategy fortifies security by exploiting complementary aspects of diverse LL techniques, selecting specific corruption levels, and tailoring SAT resilience to optimize protection against attacks. In these cases, a CG is identified in which one of its inputs consists exclusively of key inputs from RLL, while the other input of the CG incorporates key inputs from PSLL. This configuration enhances security by intertwining distinct locking techniques and increasing the complexity of the attack. However, the presence of CGs also introduces a potential dependency between the two layers that attackers may attempt to exploit.

2.3 Threat Models in Logic Locking

Understanding the threat models involved in attacking LL techniques is essential for evaluating their effectiveness. Threat models help identify potential adversaries and their capabilities, which is crucial for securing hardware designs against attacks. There exist two main threat models: **OG** and **OL** models.

2.3.1 OG Threat Model

In addition to the locked netlist, the attacker has access to a functional IC that can be used to query inputs and observe the corresponding outputs. This model assumes that the attacker has the locked netlist and the capability to apply inputs to the functional IC but does not have direct access to the secret key.

SAT-based attacks are the most prominent example of OG attacks [16, 23, 75, 76]. These attacks use an oracle to iteratively eliminate incorrect key guesses by applying DIPs that expose inconsistencies between the locked netlist and the functional IC. The objective is to eliminate incorrect key guesses and continually reduce the search space. These attack methods and others have highlighted the effectiveness of SAT-based methods in exploiting vulnerabilities in LL schemes.

Beyond SAT-based attacks, adversaries may employ other methods to undermine logic locking. Removal attacks focus on identifying and bypassing the obfuscation structures within the locked circuit [25]. Approximation attacks, on the other hand, attempt to create a simplified model of the circuit that replicates its behavior without requiring the original key [23].

2.3.2 OL Threat Model

The OL threat model represents a more restrictive scenario for an attacker, assuming that only the locked netlist is available without access to the functional IC. Unlike the OG threat model, where the attacker can apply inputs to a functional IC and observe its

outputs, the OL model does not allow direct simulation of the correct circuit behavior. This makes the OL threat model significantly more challenging than the OG threat model since there is no oracle to check the original circuit behavior.

In OL attacks, adversaries often use machine learning (ML) techniques, structural analysis, or resynthesis-based attacks. These attacks can propagate a constant to find anomalies in the design after synthesis or take advantage of EDA tools to generate functionally equivalent but structurally different versions of the locked netlist [34, 77, 78]. By comparing these versions, attackers can identify patterns that may reveal the secret key.

Despite the increased difficulty of the OL model, recent advances in attack strategies have demonstrated its abilities. ML approaches have been shown to accurately predict significant portions of the key by analyzing structural features in the locked netlist [79]. Structural analysis techniques, such as identifying discrepancies in wire delays or redundant paths introduced by locking mechanisms, further expose potential weaknesses [80]. Resynthesis-based attacks remain particularly potent, exploiting design inconsistencies during synthesis and optimization, effectively reducing the design's obfuscation strength [34]. These methods illustrate how attackers exploit the lack of functional IC by taking advantage of sophisticated tools and algorithms.

Moreover, advanced techniques, such as structural pruning and graph-based analysis, have been applied to detect areas of high obfuscation density, effectively narrowing down the key search space [81]. These findings highlight that while the OL model imposes significant constraints on the attacker, the rapid evolution of attack methodologies necessitates more resilient and multi-layered defense strategies in LL.

In addition to netlist-based analysis, OL adversaries may exploit physical access to extract the secret key. In [82], it was shown that the pathway between the key storage and the logic gates can be vulnerable, even when tamper-proof memory is used. Optical probing techniques were successfully used to extract the values of key inputs during runtime. This ability to recover the secret key without a functional IC reinforces the need to protect both the key storage and the signal path within the chip.

2.4 LL Defenses

Various defense techniques have been developed to strengthen LL's security against known attacks, focusing on improving resilience while balancing area, power consumption, and delay overhead. These techniques are generally classified into Pre-SAT, Post-SAT, and Beyond-SAT approaches, each addressing different aspects of attack resistance and security enhancement.

2.4.1 Pre-SAT Techniques

Traditional LL techniques were developed before SAT-based attacks became a significant threat. Among the earliest methods, RLL aimed to obfuscate circuit functionality by randomly selecting wires to be locked and inserting additional gates controlled by key inputs. While XOR/XNOR gates were commonly used, other gate types have been introduced depending on the locking strategy. In this case, the locked circuit works as intended only when the correct key is applied [14]. However, due to its random nature,

RLL remained vulnerable to structural analysis and SAT-based attacks that exploited its predictable behavior.

Another early approach, the LUT-based technique, replaced certain circuit parts with programmable LUTs that required key inputs to function correctly. Unlike RLL, which inserted additional gates, LUT-based locking modified the circuit topology, making structural analysis more challenging for attackers attempting to reverse engineer or extract sensitive information [83]. However, while increasing complexity, this technique did not directly counteract SAT-based attacks.

MUX-based techniques introduced an alternative form of obfuscation by replacing certain logic elements with multiplexers (MUXes), where the secret key determined the active logic path. This approach increased ambiguity by encoding multiple logic paths, ensuring that only the correct key selected the intended functionality. Compared to RLL and LUT-based locking, MUX-based locking created structural redundancy, complicating key extraction through direct circuit analysis. However, similar to other Pre-SAT LL methods, it remained vulnerable to OG attacks, where an attacker could infer the correct logic paths by comparing locked and unlocked circuit responses.

Strong logic locking (SLL) was an improvement over RLL aimed at increasing the complexity of key recovery attacks by introducing interdependencies between key bits. Unlike RLL, where each key bit could be analyzed independently, SLL made key bits interdependent, preventing simple brute-force approaches from solving them individually. Additionally, incorrect key values resulted in significant output corruption, making it difficult for an attacker to reconstruct the correct circuit behavior through trial and error [22]. However, despite strengthening resistance against brute force and structural attacks, SLL remained vulnerable to SAT-based attacks.

In addition, some LL techniques were introduced to counter fault-injection attacks, which attempt to extract key information by inducing controlled errors into the circuit. Fault-based logic locking (FLL) was designed to protect against such threats by incorporating fault detection and correction mechanisms, ensuring that any induced faults propagated misleading information to the attacker [84]. FLL was particularly useful in scenarios where physical attacks were a concern, such as embedded systems and cryptographic hardware. However, like other Pre-SAT techniques, it did not directly counter SAT-based attacks, as its primary function was to mislead adversaries by manipulating circuit behavior rather than preventing logic decryption.

2.4.2 Post-SAT Techniques

Anti-SAT was one of the first techniques explicitly designed to mitigate SAT-based attacks by introducing complementary logic trees. As an SFLT, it employs two complementary functions, e.g., AND- and NAND-tree structures, in such a way that the locking circuit shown in Figure 4(a), always generates a constant logic value under all input patterns if the key values are correct and otherwise, generates either 0 or 1 depending on the inputs. It also guarantees that the maximum number of unique wrong keys is close to the exponential number of key inputs. Thus, Anti-SAT forces the SAT-based attacks to explore an exponential number of key possibilities, significantly increasing the time complexity of the attack [27]. However, despite its SAT resilience, Anti-SAT's logic structure can be identified and removed through structural analysis,

making it vulnerable to removal-based attacks.

Anti-SAT-DTL (Diversified Tree Logic) was introduced to address the structural detectability of Anti-SAT. Instead of relying solely on AND/NAND trees, this technique randomly replaces some AND gates with OR/NAND/XOR gates, making the logic structure more unpredictable [36]. These modifications prevent an attacker from quickly identifying and eliminating the Anti-SAT circuit, ensuring that SAT resilience remains intact while reducing vulnerabilities to structural analysis.

Also, as an SFLT, SARLock ensures that only a small subset of inputs yields differing outputs when incorrect keys are applied, limiting the attacker's ability to find DIPs that can eliminate more than one wrong key in each iteration [16]. It adds a layer of protection by making SAT solvers ineffective, as the output for most incorrect keys remains unchanged [27].

InterLock and CASLock were introduced as advanced LL techniques to enhance security against attacks. InterLock improves security by linking critical key bits to specific circuit functions and intercorrelating logic and routing paths, ensuring that incorrect keys lead to unpredictable behavior across various circuit regions. This interdependence creates a complex structure that makes SAT-based attacks highly ineffective, complicating the analysis for attackers [85]. CASLock, on the other hand, uses cascaded logic gates, significantly increasing the number of DIPs and forcing attackers to explore a much larger search space. The cascaded structure provides multiple layers of protection, each with its own locking mechanism, making it challenging for attackers to find a solution, even with sophisticated SAT attacks [63].

SKG-Lock extends the concept of SAT resistance by embedding SAT-resistant key gates (SKG) at multiple locations within the circuit. Unlike Anti-SAT, which relies on structured complementary logic trees, SKG-Lock strategically distributes SKGs across different regions of the design to increase attack complexity [65]. This approach disrupts the ability of SAT attacks to systematically eliminate incorrect keys while also making structural analysis-based attacks more challenging.

A significant advancement in post-SAT techniques was the introduction of stripped functionality logic locking (SFLL), which shifted the focus from DIP-based defenses to functionality removal strategies. Rather than limiting an attacker's ability to extract key information, as a DFLT, SFLL removes specific circuit functionalities, which can only be restored with the correct key [62]. This ensures that an incorrect key results in missing or altered behavior, making it significantly harder for an attacker to infer the correct logic structure. SFLL is particularly effective because it does not introduce easily detectable locking structures, increasing its resilience to both SAT-based and removal attacks.

A refinement of SFLL, tenacious and traceless logic locking (TTLock) enhances its effectiveness by ensuring that the locked design only differs from the original design for a single specific input pattern. This guarantees that, under incorrect keys, the circuit behaves identically to the unlocked version for all other input patterns, making it resistant to common attack analysis methods [28]. However, despite its traceless nature, an attacker can still identify critical points in the locked netlist [30, 69].

An evolution of SFLL, SFLL-HLS, took SFLL to a new level by incorporating obfuscation techniques during the high-level synthesis stage, creating context-sensitive

complex locks. As a DFLT, SFLL-HLS applies multiple layers of protection across functional units, making the circuit more resistant to SAT-based attacks [86].

Cyclic obfuscation introduces cyclic dependencies into the logic, creating feedback loops that mislead SAT solvers [87]. These cycles disrupt the fundamental assumption of acyclic combinational circuits used by traditional SAT-based attacks, causing the solver to enter infinite loops or fail to converge on a solution.

Logic optimization-based cyclic logic locking (LOOPLock) introduces cyclic dependencies into the logic, creating feedback loops that significantly increase the difficulty for SAT solvers to break the locking scheme [88]. By carefully optimizing these loops, LOOPLock ensures that the circuit is resistant to SAT-based attacks, as the solver is forced to handle the inherent complexity of cyclic dependencies, which creates misleading paths and significantly enlarges the solution space. Unlike traditional cyclic locking techniques, LOOPLock employs optimization techniques to minimize performance overhead while maintaining security, making it a more efficient and resilient defense mechanism.

CLL combines two or more LL techniques to create a multi-layered defense against various attack methods. The combination of techniques in CLL is designed to exploit the strengths of each method, thereby making the circuit highly resistant to SAT attacks while also complicating structural attacks [30]. The trade-off for this enhanced security is an increase in hardware overhead in terms of area, power, and delay. However, the benefits of improved security often outweigh the cost in applications demanding high security. Examples include bilateral logic encryption (BLE) and Anti-SAT combined with RLL. BLE introduces two complementary functions, ensuring that only the correct key can yield the correct output, while Anti-SAT with RLL leverages Anti-SAT's SAT resistance alongside RLL's high output corruption to enhance security by utilizing strengths from each technique [29]. These are examples of a CLL technique that integrates multiple locking strategies to reinforce protection against attacks.

Corrupt and Correct (CAC) 2.0 aims to combat structural analysis attacks. CAC 2.0 builds on previous LL techniques by implementing double CAC, applying obfuscation at multiple nodes with different protected inputs, and introducing decoy key values [59]. These enhancements increase the attack complexity, forcing adversaries to analyze multiple incorrect key paths before identifying the correct key. By integrating SAT resistance, structural complexity, and decoy-based misdirection, CAC 2.0 represents an advanced LL defense, effectively neutralizing both SAT-based and structural attacks.

2.4.3 Beyond SAT Techniques

Physically unclonable function-based (PUF-based) LL leverages intrinsic device properties to generate unique keys for each chip. PUFs generate keys based on the physical variations inherent to each manufactured chip, making it impossible for attackers to duplicate or reverse engineer the secret key [89]. This method is tied directly to the manufacturing process, ensuring that each device has a unique and unclonable identity. However, while PUFs prevent large-scale key extraction across multiple chips, they do not invalidate the proposed attacks. Since an adversary's primary goal is to copy the IP rather than compromise an entire batch, breaking a single chip remains sufficient to expose the locked design, making the chip and its IP potentially recoverable regardless of the presence of a PUF.

2.5 LL Attacks

LL has been widely adopted as a defense mechanism in hardware security to protect the IP. However, adversaries have continuously developed sophisticated attack techniques to bypass these protections.

This subsection presents an overview of the principal LL attack methodologies, categorizing them into OG attacks, which leverage a functional IC as an oracle, and OL attacks, which rely solely on the locked netlist for key recovery. Recently, a classification of adversaries based on their knowledge of the locking techniques was introduced in [30], defining three types of adversaries: specific adversary (SA), knowledgeable adversary (KA), and oblivious adversary (OA). SA attacks are tailored to a single, specific LL technique; KA attacks target a broader set of LL techniques where the adversary has prior knowledge of the locking mechanisms; and OA attacks use generic tools capable of yielding effective results without requiring prior knowledge of the LL techniques applied. Most of the existing attacks fall into the SA and KA categories. However, real-world security challenges increasingly demand the development of more OA-based attack strategies.

2.5.1 OG Attacks

One of the first OG attacks was based on automatic test pattern generation (ATPG), a technique initially designed to detect hardware faults. This method was later adapted into the ATPG-based attack, which leverages ATPG techniques to expose secret key values in LL designs. The ATPG-based attack treats locked gates as faults and generates input patterns that reveal key dependencies. By effectively avoiding brute-force techniques, this method exposes weaknesses in locked designs that depend on simple signal dependencies [15]

The SAT-based attack remains one of the most important OG attacks, as it systematically refines the key space using DIPs [20, 76, 84]. The attack begins by duplicating the locked netlist, using the same input signals but with different key inputs. A miter circuit is then introduced, which compares the outputs of the two locked instances and generates a conjunctive normal form (CNF) representation of this circuit. The SAT solver is then tasked with finding a DIP that produces different outputs in the two locked versions. Once a DIP is identified, it is applied to the oracle to determine the correct output. This process ensures that at least one of the found keys is incorrect, allowing the solver to eliminate invalid key values progressively. The attack then adds the CNF constraints corresponding to the observed DIP to prevent the rediscovery of the same pattern and iterates the process. This continues until the SAT solver fails to find a new DIP, at this point, the last remaining key is guaranteed to be the correct secret key. This systematic refinement enables the SAT-based attack to break many traditional LL schemes, making it a formidable threat to hardware security.

An extension of the SAT-based approach, the Double DIP attack, improves attack efficiency by leveraging two sets of DIPs to accelerate key recovery. By reducing the number of iterations required to reach the correct key, Double DIP remains particularly effective against single-layer LL methods [75].

The AppSAT (approximate SAT) attack was introduced to mitigate high-complexity

SAT computations by allowing an approximation mechanism. Instead of fully solving the key recovery problem, AppSAT finds an approximate key that produces a functionally equivalent circuit, even if not an exact match to the original design, using a small number of queries. This approximation is useful against complex LL schemes, as it finds an approximate correct key that makes the circuit functional for the applied queries [23].

Expanding on AppSAT, the AppSAT guided removal (AGR) attack targets CLL techniques, particularly those incorporating point functions such as Anti-SAT. AGR combines AppSAT's approximation approach with structural analysis, refining key recovery through logical inference. This method applies AppSAT to deduce an approximate key and then performs structural analysis to pinpoint the exact key bits associated with the point function-based LL techniques [25].

The Fa-SAT (fault-aided SAT) attack targets CLL designs using a fault-based approach. It introduces faults in the circuit to guide the SAT solver more efficiently. By inducing controlled faults, attackers create additional DIPs that enhance the ability of the SAT attack to converge on the correct key, allowing secret key extraction even in advanced CLLs such as BLE and RLL combined with Anti-SAT. This attack is particularly effective against locking schemes that rely heavily on input-output behavior, as the fault responses provide the SAT solver with rich data, expediting the solution [74].

The query attack applies a set of carefully selected queries to determine the values of key bits of the secret key using the concept of proof by contradiction. Query attack is often combined with structural analysis techniques to enhance efficiency and are exceptionally successful against LL methods that rely solely on input-output behavior for security [30, 35].

The quantified Boolean formula (QBF) solver generalizes the SAT attack by incorporating existential ($\exists$) and universal ($\forall$) quantifiers, allowing adversaries to construct complex logical formulations that expose key dependencies. The QBF-based attack targets the locking/restore unit in SFLT and DFLT in two steps: it first constructs a QBF problem by combining the CNF formulas of individual gates and then generates two separate QBF problems, one where the output is *0* and another where it is *1* for all possible input combinations. The secret key can be extracted by solving these problems using a QBF solver. If a solution exists for either QBF problem, the key for the locked circuit is discovered. This dual-problem formulation ensures that all output possibilities are evaluated. It is effective against designs like SFLT, whose locking unit produces a constant value under the secret key for all inputs [26].

Functional analysis attacks exploit input-output behavior to deduce the correct key in LL schemes by systematically analyzing functional dependencies. This approach bypasses structural defenses, revealing that even obfuscated circuits may be vulnerable to attacks based solely on observable functionality, allowing attackers to break the lock by simplifying the circuit model through functional analysis [68].

The Valkyrie tool is a comprehensive vulnerability assessment and attack framework created to test and break LL techniques that claim to be secure. Valkyrie probes the locked circuit's logic and signal paths by combining structural and functional analysis to identify areas where key bits impact overall functionality, simulating potential attack scenarios to reveal weaknesses. It examines signal flow traces to understand how key values influence the circuit, effectively isolating critical regions susceptible to attack.

This approach allows Valkyrie to uncover the correct key by focusing on critical points within the circuit. Valkyrie is a KA attack where the adversary knows the techniques employed in advance. Effectiveness across fifteen different techniques demonstrates that even theoretically secure LL methods may exhibit vulnerabilities in real-world applications, underscoring the need for continued innovation in hardware security [69].

2.5.2 OL Attacks

The synthesis-based constant propagation attack for security evaluation (SCOPE) attack identifies constant signal propagation within circuits to expose key values by analyzing the impact of setting a constant logic value $0/1$ to a key input on area, delay, power, fan-in, fan-out, and other critical design metrics of the locked netlist. Adversaries can deduce key-related patterns, significantly narrowing down possible key values [78].

The topology-guided attack (TGA) is a structural analysis attack and leverages the circuit's structural layout to infer key dependencies. TGA deduces key bits relationships by mapping the circuit's topology and identifying sensitive nodes, enabling attackers to break LL schemes based solely on circuit structure. TGA is quite effective against LL techniques that do not mask structural dependencies, as it exploits the inherent connections between key bits and critical nodes [90].

Structural analysis techniques serve as the foundation for many OL attack strategies. These approaches identify weaknesses in LL designs by detecting irregularities in circuit connectivity, enabling attackers to extract key dependencies without an oracle [91].

Advances in ML-based attacks have introduced graph neural networks (GNNs) as a powerful tool for predicting key bits. Attackers can identify patterns in new, unseen circuits by training ML models on known locking schemes. This approach provides an OL attack to predict key bits, which is especially effective in modern circuits with complex locking patterns, as GNNs excel at detecting relational dependencies [92].

Resynthesis-based attacks exploit the synthesis process to reveal key dependencies. These attacks re-synthesize the locked circuit using various design constraints, generating functionally equivalent but structurally different versions of the netlist. By analyzing structural differences across iterations, adversaries can systematically expose key dependencies [34].

As LL advances, attackers quickly adapt, developing sophisticated methods to bypass protections. SAT-based attacks remain central among OG approaches, with variants like Double DIP, AppSAT, and Fa-SAT bringing notable gains in effectiveness against complex LL designs. Meanwhile, the rise of synthesis-based and ML-driven techniques has expanded the range of OL attacks, enabling intrusions that sidestep the need for output data. This continuous evolution in attack strategies emphasizes the need for LL defenses to keep pace, ensuring robust protection against these persistent threats.

2.6 Benchmark Circuits and Metrics

The benchmark circuits analyzed in this thesis include combinational designs from the ISCAS'85 [93] and ITC'99 [94] suites, which are widely used to evaluate the resilience of LL schemes against both OG and OL attacks. Additionally, circuits from the CSAW'19 [95] benchmark set were used to assess CLL techniques, as they integrate

two LL strategies simultaneously. FIR filter circuits were also included to evaluate the impact of hybrid protection in more application-driven contexts.

Throughout the thesis, a wide range of LL techniques are investigated. These include RLL [14] as a baseline, as well as more sophisticated approaches such as Anti SAT [27], Anti SAT DTL [96], CASLock [63], SARLock [16], SKG Lock [65], TTLock [28], and SFLL [64]. Combinations of RLL with other PSLL schemes were also employed to construct CLL circuits. Although all these techniques are explored in detail in later chapters, in this subsection, we focus on evaluating the impact of five LL techniques, namely RLL, Anti-SAT, TTLock, RLL+Anti-SAT, and RLL+TTLock, on the hardware complexity in terms of area, power, and delay and on the resiliency to the existing attacks. These techniques were applied to five circuits from the ISCAS'85 suite and synthesized using the Cadence Genus logic synthesis tool with a commercial 65*nm* standard cell library. The number of key inputs varied between 32 and 64 bits, depending on the circuit size, to balance security and hardware complexity on a single LL technique. In CLL, we use the same number of key inputs in the RLL and PSLL techniques.

Table 1 presents the characteristics of the original ISCAS'85 circuits, where #in and #out denote the number of inputs and outputs, respectively, *area* is the total area in μm^2, *power* is the total power dissipation in mW, and *delay* is the critical path delay in ps. The corresponding implementation overheads introduced by each LL scheme—measured in terms of area, power, and delay—are summarized in Table 2.

Table 1: Details of ISCAS'85 circuits.

Circuit	Original Netlist				
	#in	#out	area	power	delay
c2670	157	64	1046	3.36	1264
c3540	50	22	1518	6.58	1977
c5315	178	123	2460	9.9	1864
c6288	32	32	3133	8.48	4621
c7552	206	105	2702	1.32	1663

Table 2: Overhead in area, power, and delay for each LL technique, and run-time of attacks.

Technique	**Area (%)**	**Power (%)**	**Delay (%)**	**Attack**	**Run-time (s)**
RLL	19.1 to 31.2	58.2 to 78.0	6.3 to 17.6	SAT [20]	0.29 to 2.69
Anti-SAT	8.2 to 31.9	11.5 to 35.1	2.3 to 10.1	KRATT [26]	0.26 to 1.13
TTLock	5.7 to 30.0	4.4 to 31.8	1.3 to 6.9	KRATT [26]	72.38 to 333.20
RLL+Anti-SAT	20.3 to 42.9	35.8 to 47.3	3.6 to 13.2	RESAA [30]	540.43 to 929.78
RLL+TTLock	24.7 to 54.6	34.8 to 47.7	6.4 to 15.3	RESAA [30]	418.11 to 662.32

To illustrate the impact of these techniques, RLL incurs area overheads ranging from 19.1% to 31.2%, with corresponding power increases from 58.2% to 78.0% and delay overheads from 6.3% to 17.6%. Anti-SAT exhibits lower power and delay overheads than RLL, while its impact on area varies from 8.2% to 31.9%. TTLock presents an overhead ranging from 5.7% to 30.0% in area, between 4.4% and 31.8% in power, and between 1.3% and 6.9% in delay. When RLL is combined with Anti-SAT or TTLock in CLL configurations, overheads increase significantly. RLL combined with Anti-SAT leads to area, power, and delay overheads of up to 42.9%, 47.3%, and 13.2%, respectively, while RLL combined with TTLock has the largest impact on the hardware complexity.

Despite the additional complexity, all these locked circuits remain vulnerable to known attacks, as shown in Table 2. RLL is broken by SAT-based attacks [20] within 0.29 to 2.69 seconds. Anti-SAT and TTLock are defeated using the structural analysis attack KRATT [26] in 0.26 to 1.13 seconds and 72.38 to 333.20 seconds, respectively. CLL schemes, such as RLL combined with Anti-SAT and RLL combined with TTLock, are broken by the RESAA attack [30], which increases attack time to a range between 418.11 and 929.78 seconds, still within practical limits for an adversary. These results and evaluations serve as a baseline for the subsequent experimental discussions, which further analyze attack strategies and the complexity and security tradeoff.

3 Discussion

This chapter provides an overview of the three papers on which this thesis is based, showing the main findings, defense methods, and proposed attacks in the field of hardware security for digital circuits. These works demonstrate the growing sophistication of defense and attack mechanisms focused on hybrid techniques and CLL methods that could integrate resilience against attacks while maintaining performance.

3.1 Hybrid Protection of Digital FIR Filters

FIR filters are fundamental blocks in digital signal processing (DSP) due to their stability and predictable phase characteristics. Unlike generic digital circuits, where the IP is often linked to architecture and functional logic, the actual value in FIR filters lies in their coefficients. These coefficients define the filter's frequency response and, consequently, its behavior in applications such as communications, image processing, and biomedical signal analysis. Without adequate protection, adversaries can extract these coefficients through reverse engineering, effectively replicating the IP while bypassing the cost and effort associated with its design [32]. However, only a limited number of techniques have been proposed to obfuscate DSP circuits, particularly digital filters. Existing methods have significant vulnerabilities when faced with more advanced attacks [32, 97], necessitating more robust security mechanisms.

This subsection provides an overview of the research published in *IEEE TVLSI* 2023 as given in Appendix A. This study introduces a novel **query attack** that exploits weaknesses in existing protection schemes. An adversary can extract values of key bits by performing targeted queries, ultimately reconstructing the filter's coefficients. This method highlights the limitations of conventional hardware obfuscation and demonstrates that existing LL techniques alone are insufficient to secure FIR filter designs.

As a countermeasure, a **hybrid protection mechanism** is proposed, integrating hardware obfuscation with LL to safeguard both the filter's coefficients and functionality. This approach enhances obfuscation by leveraging decoy constants and a key-based technique, making it significantly harder for adversaries to extract the correct coefficients or replicate the filter's behavior.

The security and efficiency of locked architectures depend on the underlying multiplication method used. Constant multiplication generally occurs in many DSP applications, particularly in FIR filters, where coefficients are multiplied by input values. Three architectures are commonly used for such operations depending on the filter architecture, i.e., transposed or direct form: constant array vector multiplication (CAVM), multiple constant multiplication (MCM), and time-multiplexed constant multiplication (TMCM). CAVM performs the multiplication of a $1 \times n$ constant array by an $n \times 1$ input vector, generating a single output through the summation of all partial products. MCM, on the other hand, performs the multiplication of a set of *n constants* by a single input variable. Finally, TMCM allows for the time-multiplexed selection of one constant from a set of *n constants*, multiplying it by an input variable at each time step. These architectures enable efficient implementation of FIR filters, particularly in hardware-constrained environments where reducing multiplications through addition and shift operations is essential.

Table 3: Results of obfuscated and protected multiplier blocks. This table is the same as the Table V in Publication I.

Block	Architecture	Technique	Synthesis			Attacks							
						SAT	ATPG	AppSAT	DoubleDIP	Query		SCOPE	
			area	delay	power	time	time	time	time	prv	time	cdk/dk	time
CAVM	CAVM-MUL	Decoy [32]	15435	4616	4641	155143	OoT	OoT	OoT	32	9893	20/32	8
		Proposed Hybrid	15710	4611	4757	OoT	OoT	OoT	OoT	0	OoT	1/1	13
	CAVM-SA	Decoy [32]	15465	4611	4475	36083	4539	OoT	OoT	32	9944	20/32	8
		Proposed Hybrid	15704	4715	4497	OoT	OoT	OoT	OoT	0	29328	1/1	12
	CAVM-CRK	Constant [58]	18737	3982	4756	110	1446	OoT	OoT	32	897	21/27	11
		Proposed Hybrid	18976	4265	4809	OoT	OoT	OoT	OoT	0	1937	2/3	16
MCM	MCM-MUL	Decoy [32]	10949	3102	2839	106	OoT	324	243	32	176	27/32	7
		Proposed Hybrid	11173	3031	2897	OoT	OoT	OoT	OoT	0	197	1/1	11
	MCM-SA	Decoy [32]	10493	3112	2493	119	OoT	342	254	32	152	27/32	7
		Proposed Hybrid	10705	3159	2495	OoT	OoT	OoT	OoT	0	115	1/1	11
	MCM-CRK	Constant [58]	12799	2772	2412	159	OoT	415	300	32	459	18/32	7
		Proposed Hybrid	13038	2970	2456	OoT	OoT	OoT	OoT	0	759	1/1	11
TMCM	TMCM-MUL	Decoy [32]	1545	3517	610	241	6783	378	496	32	7	21/32	4
		Proposed Hybrid	1794	4278	639	OoT	OoT	OoT	OoT	0	17	2/3	2
	TMCM-SA	Decoy [32]	2043	4452	1037	738	963	935	OoT	32	37	17/32	4
		Proposed Hybrid	2245	4536	1065	OoT	OoT	OoT	OoT	0	42	1/2	3
	TMCM-CRK	Constant [58]	1566	2997	623	1035	15571	1032	OoT	32	29	20/32	4
		Proposed Hybrid	1776	3655	642	OoT	OoT	OoT	OoT	0	48	1/1	2

The experimental results are presented in Tables 3 and 4. These tables compare the performance and security resilience of various locked multiplier blocks, showing the superiority of the hybrid approach. These tables present the evaluation metrics and attack results. Note that *area*, *delay*, and *power* represent the total area in μm^2, critical path delay in ps, and total power dissipation in μW, respectively. It includes results from SAT, ATPG, AppSAT, DoubleDIP, and SCOPE attacks, where *cdk* and *dk* denote correctly and total deciphered key bits for SCOPE, respectively. All attacks had a 2-day time limit, and designs are highlighted where the secret key remained undiscovered.

Different architectures implementing the multiplication block of the FIR filter with varying hardware complexity were tested to validate the proposed protection mechanism. MUL-based represents a conventional approach where multiplications are performed directly using hardware multipliers, as in the CAVM-MUL, MCM-MUL, and TMCM-MUL implementations. SA architecture eliminates the need for multipliers by implementing constant multiplications using only addition, subtraction, and shift operations, as seen in CAVM-SA, MCM-SA, and TMCM-SA. Constant replacement with key bits (CRK) enhances security by replacing selected constant values with key bits, making it harder for an adversary to infer the correct functionality, as demonstrated in CAVM-CRK, MCM-CRK, and TMCM-CRK. Each of these architectures was locked and evaluated using different LL techniques, providing a robust dataset for comparing the effectiveness of hybrid protection against traditional LL techniques.

Table 3 compares the performance of locked multiplier blocks under different obfuscation techniques. The hybrid protection method consistently outperforms existing obfuscation techniques, particularly in its resilience to the query attack. Although existing obfuscation techniques may be broken within a specific timeframe, the hybrid method provides a more robust defense, enduring prolonged attack attempts without compromising the secret key.

As seen in Table 4, hybrid protection generally demonstrates greater resilience against the query attack than traditional LL techniques, though some locked designs remain secure. While LL can be compromised within certain time limits, the hybrid technique offers a more robust defense, resisting extended attacks without leaking critical

Table 4: Results of locked multiplier blocks. This table is the same as the Table VI in Publication I.

Block	Logic Locking	Synthesis			Attacks							
					SAT	ATPG	AppSAT	DoubleDIP	Query		SCOPE	
		area	delay	power	time	time	time	time	prv	time	cdk/dk	time
CAVM	RLL	16411	3385	4183	171	OoT	205	2609	32	254	0/0	9
	RLL+AntiSAT	16492	3416	4127	OoT	OoT	OoT	OoT	16	364	0/0	14
	RLL+CASLock	16473	3502	4110	OoT	OoT	OoT	OoT	16	443	0/0	13
	RLL+SARLock	16512	3572	4191	OoT	OoT	48855	OoT	32	441	8/8	13
	RLL+SFLL	16506	3473	4205	339	829	39664	OoT	32	436	0/0	14
	RLL+SKGLock	16559	3894	4308	OoT	OoT	OoT	OoT	19	513	5/6	14
MCM	RLL	8090	2398	2039	86	122	293	371	32	89	0/0	6
	RLL+AntiSAT	8244	2434	2065	OoT	OoT	OoT	OoT	16	249	0/0	9
	RLL+CASLock	8121	2404	2024	OoT	OoT	1054	OoT	16	164	0/0	9
	RLL+SARLock	8209	2467	2041	OoT	OoT	45013	OoT	32	228	10/10	9
	RLL+SFLL	8166	2452	2019	576	7870	2626	OoT	32	243	0/0	9
	RLL+SKGLock	8252	2464	2056	OoT	OoT	OoT	OoT	19	160	5/5	9
TMCM	RLL	1587	3712	646	8	39	57	77	32	15	0/0	2
	RLL+AntiSAT	1659	3545	632	OoT	OoT	OoT	OoT	13	28	0/0	2
	RLL+CASLock	1653	3664	628	OoT	OoT	OoT	OoT	15	20	0/0	2
	RLL+SARLock	1659	3756	658	OoT	OoT	2662	OoT	31	36	6/6	3
	RLL+SFLL	1644	3800	653	1095	1089	5363	OoT	32	36	0/0	2
	RLL+SKGLock	1694	3739	676	OoT	OoT	OoT	OoT	18	34	10/10	2

information. This enhanced resilience, however, incurs a modest trade-off in synthesis metrics: hybrid protection typically increases the total area and power consumption by around 1–5%. The delay overhead is usually modest but varies significantly with the underlying architecture, ranging from negligible to approximately 21% in extreme cases, such as the TMCM multiplier block. Nonetheless, this trade-off remains reasonable, given the substantial security improvements.

In conclusion, the hybrid protection technique significantly improves the security of FIR filters against advanced attack methods such as query attacks. While not entirely immune to future attack evolutions, this approach considerably strengthens the defense of critical components in digital signal processing systems, ensuring their integrity against unauthorized access.

3.2 Resynthesis-based Attacks Against Logic Locking

Attacks using the EDA tools engine have emerged as a powerful technique to break LL schemes by exploiting changes to the netlist or constant propagation to uncover secret keys [34, 98]. Synthesis tools translate a high-level behavioral description of a circuit into standard cells from a target technology library. This process involves mapping abstract logic functions to specific components available in the library, ensuring that the generated design meets constraints such as area, power, and delay.

This subsection presents a novel **resynthesis-based attack** that enhances existing methodologies under the OG and OL threat models. The attack leverages structural diversity generated by multiple synthesis iterations with varying parameters, enabling key recovery by exposing inconsistencies across locked netlists. This work demonstrates that resynthesis inadvertently weakens the security of LL designs, making them more susceptible to previously ineffective attacks.

The proposed approach and its implications for LL security have been published in the *ISQED Conference*, 2023. The complete experimental results and further details are provided in the Appendix B. The main contribution of this work is to demonstrate

that resynthesis not only increases the chances of key recovery but also weakens the locked circuits against attacks that would otherwise fail.

The resynthesis process systematically generates a large set of unique, yet functionally equivalent, locked circuits by varying synthesis parameters such as timing and logic optimization effort. These variations introduce subtle differences in the circuit's structure, which attackers exploit to recover key information. This resynthesis-based attack is applicable to a wide range of LL techniques, including Anti-SAT [27], CASLock [63], SFLL [64], and SKG-Lock [65]. By creating multiple unique versions of the locked netlists, the attack uncovers significantly more key bits than traditional attacks on the original netlists. Furthermore, by combining OL and OG attacks, the proposed technique strengthens key recovery accuracy, making it a powerful tool against single LL and CLL schemes.

Figures 5 and 6 highlight the structural differences between two synthesized netlists generated with different delay constraints. These differences, though subtle, are critical for understanding why resynthesis amplifies the success of attacks like SCOPE. The figure shows how small changes in synthesis parameters can drastically alter the circuit's gate count and logic depth. Such structural diversity is critical to exposing vulnerabilities in the locked design, as it allows the attack to leverage variations in the logic structure to recover more key bits.

Table 5 shows the results of OL attacks on locked ISCAS'85 circuits. The table compares the success of the SCOPE attack on the original locked netlist versus the resynthesized netlists. It is evident that while the SCOPE attack is not entirely successful on the original locked netlist, resynthesis enables the recovery of a significant number of key bits. For instance, circuits like *c2670*, when locked with SKG-Lock, have 100% of their key bits recovered after resynthesis, compared to partial or unsuccessful recovery without resynthesis. The secret key is also found in other benchmarks when the undetermined key inputs are assigned a constant logic value of *0* or *1*. This underscores the effectiveness of resynthesis in compromising LL techniques previously considered resilient to attacks like SCOPE. The results demonstrate that even SKG-Lock, an SAT-resilient locking technique, is susceptible to resynthesis, revealing the hidden key bits.

Table 6 extends the analysis to CSAW'19 circuits, which use CLL combining RLL and SFLL-rem [95]. The table compares the results of the proposed resynthesis-based attack with traditional OL attacks. Full key recovery is achieved for RLL, even under complex circuits. While SFLL-rem presents a more difficult challenge, the resynthesis-based attack still outperforms other OL methods, demonstrating a large number of recovered key bits, recovering up to approximately 44% of the previously undiscoverable key bits. This result underscores the robustness of the resynthesis approach in handling both OL and OG threat models. The ability to break RLL and still recover partial key bits under more complex schemes like SFLL-rem highlights the broad applicability of the method across different LL techniques.

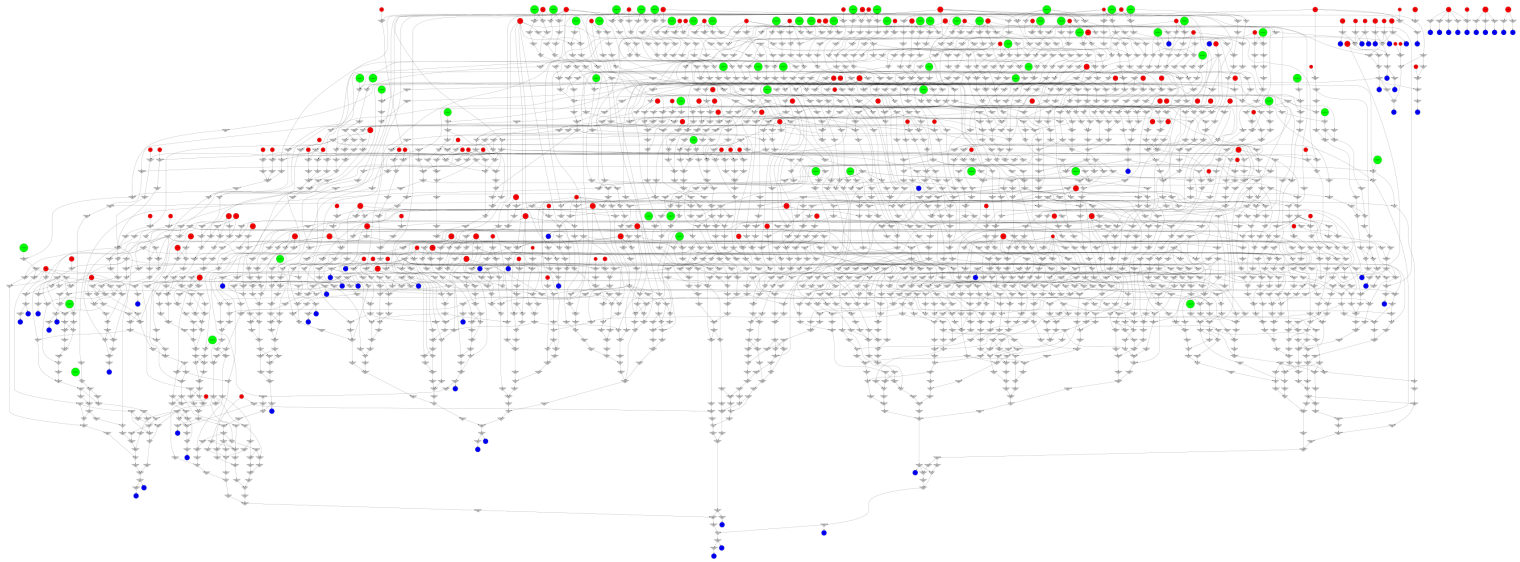


Figure 5: Graph of the netlist resynthesized when the delay constraint is 990 ps. This figure is reproduced from Figure 5(a) in Publication II.

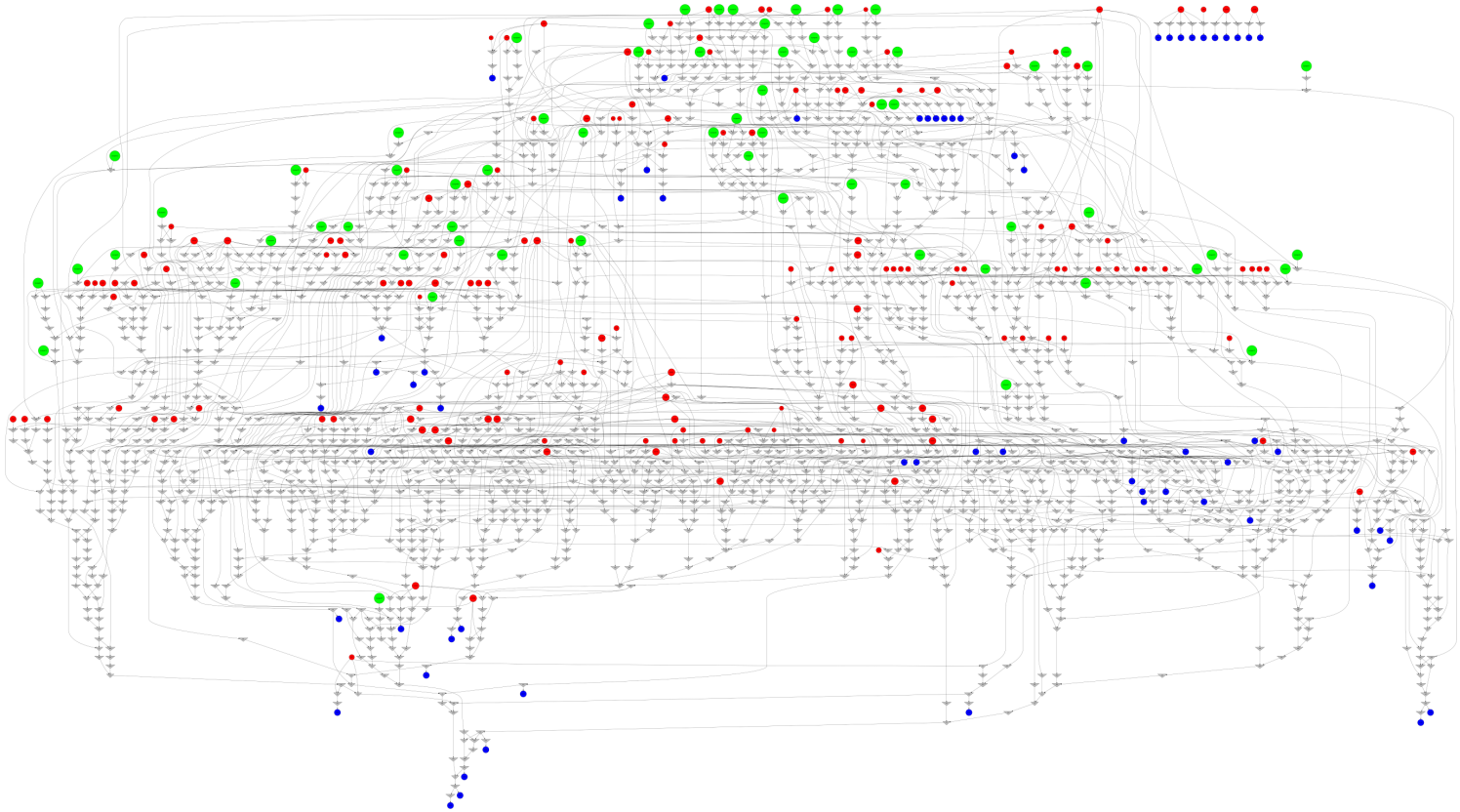


Figure 6: Graph of the netlist resynthesized when the delay constraint is 496 ps. This figure is reproduced from Figure 5(b) in Publication II.

Table 5: Results of OL Attacks on the locked ISCAS'85 Circuits. This table is the same as the Table III in Publication II.

Circuit	Anti-SAT			CASLock			SFLL			SKG-Lock		
	SCOPE	Resynthesis		SCOPE	Resynthesis		SCOPE	Resynthesis		SCOPE	Resynthesis	
	cdk/dk	time	cdk/dk time	cdk/dk	time	cdk/dk time	cdk/dk	time	cdk/dk time	cdk/dk	time	cdk/dk time
c2670	0/0	4s	37/64 34m18s	0/0	4s	35/64 33m47s	0/0	4s	34/64 37m32s	32/32	4s	64/64 44m37s
c3540	0/0	3s	17/32 21m27s	0/0	3s	17/32 18m12s	0/0	2s	19/32 21m29s	17/17	2s	32/32 24m30s
c5315	0/0	5s	38/64 42m34s	0/0	5s	30/64 43m54s	0/0	5s	33/64 46m23s	32/32	5s	62/62 52m06s
c6288	0/0	3s	18/32 29m08s	0/0	3s	16/32 27m18s	0/0	3s	16/31 33m19s	16/16	3s	31/31 34m24s
c7552	0/0	6s	38/64 45m31s	0/0	6s	47/64 49m13s	0/0	6s	38/63 52m26s	32/32	6s	61/61 56m45s

Table 6: Results of attacks on the locked CSAW'19 Circuits. This table is the same as the Table VII in Publication II.

Approach	Attack Scenario	Circuit							
		small (40+40)		medium (60+60)		large (80+80)		bonus (128+128)	
		RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem
Key sensitization [15]	OG	40/40	—	60/60	—	80/80	—	—	—
Hamming distance-based attack [95]	OG	30/30	—	50/50	—	72/72	—	—	—
Automated analysis + SAT [95]	OG	11/18	—	31/50	—	10/34	—	—	—
Sub-circuit SAT [95]	OG	17/17	—	29/29	—	—	—	—	—
Redundancy-based [99]	OL	28/28	4/12	35/35	23/28	45/45	0/51	66/66	8/27
Bit-flipping attack [76]	OG	40/40	—	60/60	—	80/80	—	—	—
Topology guided attack [90]	OL	15/32	—	19/50	—	36/73	—	75/108	—
Resynthesis-based attack	OG	40/40	20/39	60/60	29/60	80/80	35/79	128/128	55/124

These results demonstrate that a resynthesis-based attack is critical in exposing weaknesses in logic-locked circuits. The proposed attack significantly improves the effectiveness of OL attacks by exploiting the structural variations introduced during resynthesis. Even for circuits locked with SAT-resistant techniques, resynthesis enables attackers to recover key bits that would have otherwise remained hidden. Furthermore, by combining OL and OG methods, this approach achieves higher key recovery accuracy, making it a robust tool against advanced locking schemes like SFLL-rem.

This study highlights the importance of resynthesis in hardware security research. By systematically altering the structure of locked circuits, we expose vulnerabilities that are not easily detectable with traditional attacks. The results achieved under both OL and OG threat models reinforce the effectiveness of the proposed attack.

3.3 RESAA: A Removal and Structural Analysis Attack Against Compound Logic Locking

CLL schemes have been introduced to integrate a multi-layered LL strategy to protect ICs against advanced attack methodologies. By combining multiple LL techniques, CLL aims to leverage the strengths of different approaches while mitigating their weaknesses. However, recent studies have identified structural vulnerabilities that adversaries can exploit [25, 74]. These vulnerabilities arise from how LL techniques interact within the CLL structure, potentially exposing CGs or enabling key differentiation through attack strategies that partition the locked design.

This subsection introduces **RESAA**, a novel attack framework that leverages both removal and structural analysis techniques to break CLL. RESAA was developed as part of this thesis and has been published in *IEEE TVLSI 2025*. For further details, the full version of this work, including extended experimental results, is provided in Appendix C. The following sections detail the attack methodology, experimental validation, and implications for future LL defenses.

RESAA exposes vulnerabilities in CLL designs by leveraging a combination of SAT-

based attacks, structural analysis, and query-based attacks. The core of RESAA lies in classifying the keys of a CLL-locked circuit into two categories: RLL and PSLL. By leveraging the CG, where all key inputs converge before reaching the primary output (PO), RESAA partitions the design into two distinct netlists. This partitioning not only simplifies the attack but also reduces the complexity of applying traditional attack techniques, such as SAT-based or SCOPE attacks.

Figure 7 provides the overview of the RESAA, illustrating the entire workflow from pre-processing the CLL-locked circuit to uncovering the secret key. The left side of the figure demonstrates the initial pre-processing steps, where the original circuit is locked using CLL techniques. It begins by applying RLL to the original circuit, adding XOR/XNOR gates driven by key inputs, which offer minimal overhead in terms of area and power but contribute to an increased delay, as described in the paper. Then, CLL techniques such as Anti-SAT [27], Anti-SAT-DTL [96], CASLock [63], SARLock [16], or TTLock [28] are applied. These methods increase resistance to attacks but also introduce hardware complexity and overhead. This locked circuit is then translated into a mapped Verilog netlist by the synthesis process, which forms the basis for the subsequent steps.

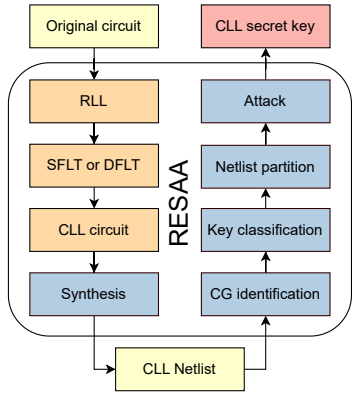
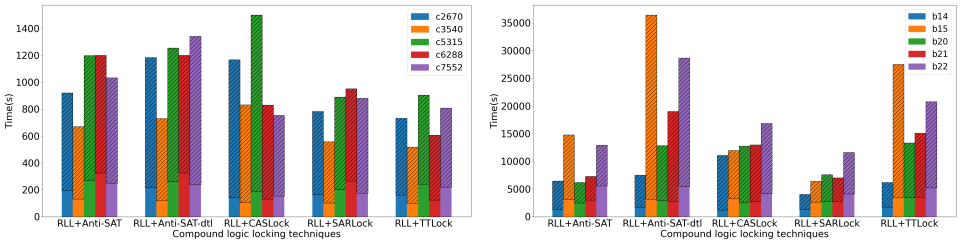


Figure 7: Overview of the RESAA framework. This figure is reproduced from Figure 4 in Publication III.

The classification process starts by identifying the CG and a key classification between RLL and PSLL. Subsequently, the netlist is divided into two distinct partitions. Each partition is processed separately, with the RLL-protected portion subjected to SAT-based and/or query-based attacks. In contrast, the PSLL-protected portion is analyzed using advanced techniques like QBF-based attacks. This partitioning strategy improves the attack's efficiency and allows RESAA to target specific vulnerabilities in each part of the circuit. By isolating one partition with only RLL keys and another with only PSLL keys and attacking the distinct portions of the CLL-protected design, the RESAA framework effectively exposes the secret keys embedded within, overcoming many of the defenses presented by the CLL mechanisms.

The experiments were conducted on ten benchmark circuits from the ISCAS'85 [93]



(a) ISCAS'85 benchmark *(b) ITC'99 benchmark*

Figure 8: Classification and execution times (seconds) for attacking ISCAS'85 and ITC'99 benchmarks in the CLL scheme. Bottom: Classification and partition time. Hatched: Attack time. Combined: Total execution time. This figure is reproduced from Figure 9 in Publication III.

and ITC'99 [94] suites. The locking mechanisms were implemented using a combination of the Neos tool [100], Python, and Perl scripts.

Table 7: Details of existing attacks in ISCAS'85 and ITC'99 circuits locked using a CLL scheme. This table is the same as the Table IV in Publication III.

Circuit	Locked Netlist																									
	RLL+Anti-SAT					RLL+Anti-SAT-DTL					RLL+CASLock					RLL+SARLock					RLL+TTLock				
	sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt	
	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time
c2670	OoT	998	OoT	52	48	OoT	170	OoT	50	45	OoT	238	OoT	50	25.4	OoT	114	OoT	55	59	OoT	1055	OoT	48	48
c3540	OoT	766	OoT	30	20	OoT	203	OoT	32	19	OoT	66	OoT	30	8	OoT	91	5	31	22	48656	4499	2	31	19
c5315	OoT	239	OoT	62	50	OoT	9949	OoT	62	52	OoT	131	OoT	62	42	OoT	64	OoT	62	72	OoT	2768	OoT	62	60
c6288	OoT	OoT	OoT	32	77	OoT	58470	OoT	32	69	OoT	3936	OoT	32	90	OoT	935	OoT	32	119	OoT	6270	OoT	32	93
c7552	OoT	172	OoT	55	96	OoT	543	OoT	55	105	OoT	279	OoT	55	59	OoT	247	OoT	55	108	OoT	51	OoT	55	79
b14_C	OoT	OoT	OoT	109	1822	OoT	OoT	OoT	112	1383	OoT	OoT	OoT	106	2082	OoT	OoT	OoT	109	1335	OoT	OoT	OoT	111	1783
b15_C	OoT	OoT	OoT	98	466	OoT	OoT	OoT	97	635	OoT	OoT	OoT	96	836	OoT	OoT	OoT	80	513	OoT	OoT	OoT	87	671
b20_C	OoT	OoT	OoT	115	1562	OoT	OoT	OoT	115	1647	OoT	OoT	OoT	109	2645	OoT	OoT	OoT	115	1645	OoT	OoT	OoT	113	2187
b21_C	OoT	OoT	OoT	113	1119	OoT	OoT	OoT	114	1337	OoT	OoT	OoT	113	1935	OoT	OoT	OoT	110	1385	OoT	OoT	OoT	109	1738
b22_C	OoT	OoT	OoT	113	1034	OoT	OoT	OoT	113	1214	OoT	OoT	OoT	108	1489	OoT	OoT	OoT	105	938	OoT	OoT	OoT	111	1342

Table 7 shows the runtime of attacks required to find the secret key, where out-of-time (OoT) indicates that no solution could be found within the allowed 48-hour time limit. As observed from Table 7, the SAT-based (sat) and DoubleDIP (dp) attacks exhibit low efficiency in deciphering key inputs, as expected. They only found a solution for one single RLL+TTLock case in the *c5315* circuit. The AppSAT attack showed promising results for small circuits but demanded significant execution time compared to other attacks. Although AppSAT (appsat) demonstrated nearly 100% efficiency in breaking ISCAS'85 circuits, it could not solve certain more complex cases, such as the *c6288* circuit locked with RLL+Anti-SAT. Lastly, the query attack from [101], qatt, displayed varying execution times and degrees of success in deciphering key inputs, as shown in Table 7. The average proportion of proven values of key inputs prv is approximately 41%, with execution times ranging from 8 to 2645 seconds.

Next, Figure 8 presents the classification and execution times under the OG threat model. In this context, "classification time" refers to the duration required for categorizing the LL technique utilized in the CLL design, depicted in the lower section of the graph. Conversely, "attack time" is indicated by the hatched portion of the graph. In contrast, "execution time" represents the total time, including both classification/partition time and the subsequent attack on each circuit, as depicted by both sections in the graph.

Table 8 compares the results of RESAA under the OL threat model with the SCOPE attack. RESAA successfully recovers a substeantial portion of the key bits across

multiple benchmarks. For example, the attack reveals 37 out of 64 key bits for the *c2670* circuit locked with RLL + Anti-SAT, and a similar performance is observed across other circuits. These results demonstrate RESAA's ability to uncover a large portion of the key even when no functional IC (oracle) is available for validation. In contrast, traditional attacks, such as SCOPE, fail to reveal any key bits, further underscoring the effectiveness of RESAA's partitioning strategy. This attack methodology is particularly potent when applied to complex CLL designs, where a direct attack on the whole circuit would otherwise be computationally infeasible.

Table 8: Results of OL Attacks on the locked ISCAS'85 and ITC'99 circuits. This table is the same as the Table V in Publication III.

Circuit	RLL+Anti-SAT				RLL+Anti-SAT-DTL				RLL+CASLock				RLL+SARLock				RLL+TTLock			
	SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA	
	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time
c2670	0/0	14	73/104	4	0/0	13	80/104	4	0/0	9	84/104	4	32/64	8	97/105	4	16/106	10	68/105	9
c3540	0/0	6	37/43	2	0/0	6	40/64	2	0/0	4	40/45	2	15/32	4	41/43	2	8/42	7	32/45	4
c5315	0/0	15	77/107	5	0/0	16	78/107	5	0/0	12	81/107	5	32/64	10	100/107	5	30/107	12	69/109	11
c6288	0/0	7	41/56	3	0/0	7	42/56	2	0/0	5	42/56	3	17/32	5	49/56	3	4/53	6	40/56	5
c7552	0/0	17	78/108	5	0/0	17	79/105	5	0/0	8	81/108	4	40/64	11	98/107	6	28/107	13	82/109	12
b14_C	0/0	91	160/210	67	0/0	91	168/215	69	0/0	67	180/213	44	68/128	71	168/200	46	36/203	87	38/72	58
b15_C	0/0	117	166/210	87	0/0	120	190/214	90	0/0	88	180/210	58	72/128	89	186/214	59	49/202	109	58/82	72
b20_C	0/0	155	172/203	115	0/0	156	182/200	117	0/0	116	191/211	77	67/128	118	188/209	77	44/201	136	62/86	93
b21_C	0/0	159	183/210	119	0/0	164	185/212	122	0/0	119	173/213	78	77/128	121	178/200	82	52/196	142	60/70	97
b22_C	0/0	231	180/209	172	0/0	233	185/212	175	0/0	177	190/205	116	86/128	175	189/199	116	48/192	193	56/70	139

In summary, RESAA offers a powerful new approach for attacking CLL-locked designs by partitioning the circuit and applying a combination of SAT-based, structural analysis, and/or query-based attacks. The results highlight the vulnerabilities in current LL techniques, even in advanced configurations like TTLock and SARLock combined with RLL. While these techniques provide better resistance to attacks, RESAA demonstrates that, with CG identification and proper partitioning, even these defenses can be overcome. The study emphasizes the need for further research into more robust locking mechanisms to counter the growing sophistication of attacks like RESAA.

4 Conclusions and Future Work

This thesis focuses on one of the most important fields of hardware security, i.e., LL techniques and their resiliency against existing attacks. As the complexity of ICs increases, so do the risks of IP theft, reverse engineering, and overproduction. LL, a heavily studied hardware obfuscation method, is crucial in addressing these threats by embedding secret keys that ensure proper circuit functionality only when the correct key is applied. Through rigorous experimentation and novel theoretical contributions, this thesis makes significant advancements in understanding LL techniques' vulnerabilities and potential enhancements.

The research presented in this thesis emphasizes the need for continuous threat evaluation in the field of hardware security. It introduces a novel query attack that exploits vulnerabilities in obfuscated FIR filters, revealing how attackers can use carefully crafted inputs to reveal the secret key. This attack exposes critical flaws in the current defenses and underscores the importance of continuously evaluating new threats in LL applications. A hybrid protection mechanism combining hardware obfuscation and LL was proposed to counter these vulnerabilities. This hybrid solution is more resilient than standalone LL techniques, mainly when dealing with parallel direct and transposed FIR filter designs. Experimental results showed that traditional LL methods had 100% key exposure, while the proposed hybrid protection had a 0% success rate of key extraction. This was at the expense of modest increases in area and power, typically between 1% and 5%, and up to approximately 21% in some instances of delay overhead. The effectiveness of this solution is demonstrated through comprehensive experimental results, revealing that hardware complexity remains competitive while security levels are significantly improved.

Second, the thesis introduces a resynthesis-based attack demonstrating how attackers can exploit structural weaknesses in locked circuits. This attack, applied under both OG and OL threat models, illustrates the limitations of traditional LL techniques, including SAT-resilient schemes. By leveraging commercial EDA tools, this attack identifies design vulnerabilities in the LL mechanisms and successfully compromises a significant number of key bits. For instance, experiments demonstrated 100% of key recovery from circuits like *c3540* locked with SKG-Lock after resynthesis. This work underscores the importance of considering the effects of synthesis tools in evaluating LL security and introduces new challenges for defending against these types of attacks.

Finally, the development of the RESAA framework represents a significant advancement in attacking CLL techniques. The framework was proposed to be an OA attack, where the adversary does not previously know the LL techniques applied in the CLL. It systematically identifies critical gates, applies structural analysis, and partitions the design, revealing specific weaknesses in the CLL technique. RESAA's ability to break down designs and efficiently target different layers of security emphasizes the complexities of securing modern ICs. For example, the average proportion of proven values of key inputs is approximately 41% under the OG attack, with execution times ranging from 8 to 2645 seconds. Under the OL attack, RESAA recovered a good portion compared to the SCOPE attack. For example, the attack reveals 37 out of 64 key bits for the *c2670* circuit locked with RLL + Anti-SAT. The experimental results demonstrate

the framework's ability to overcome even the most SAT-resilient defenses, providing valuable insights into how CLL techniques can be exploited. The findings call for more sophisticated methods that take into account the interactions between different LL techniques.

While this research has made significant progress in understanding and addressing hardware security challenges, it also emphasizes the need for continued innovation in this rapidly evolving field. Future work should focus on enhancing the resilience of LL techniques, particularly in the face of more sophisticated attacks. Integrating AI and ML tools into defensive strategies may offer new opportunities for improving LL's robustness. These technologies could detect attack patterns, predict vulnerabilities, and dynamically adapt the locking mechanisms to different attack scenarios.

Moreover, the interplay between high-level obfuscation methods, such as hardware watermarking and camouflaging, and LL should be further explored to develop a more robust and comprehensive defense strategy. Combining these techniques could enable layered protection, effectively addressing vulnerabilities at multiple stages of the IC supply chain. Additionally, the integration of novel materials offers promising opportunities for enhancing hardware security. This integrated approach creates multiple interdependent defense layers, significantly increasing the complexity of attacks and providing a resilient protective framework. These advances could redefine the way how LL is implemented, solidifying its role as a critical component in safeguarding future hardware designs against evolving threats.

Future work can investigate adapting the LL techniques presented in this thesis to emerging computing paradigms, such as neuromorphic architectures. These systems offer promising avenues for dynamic security by enabling LL implementations capable of real-time adaptation to evolving threats. Such adaptability can significantly enhance resistance to contemporary attacks. Investigating these novel architectures may lead to more resilient protection mechanisms and extend hardware security strategies to technologies with inherently different vulnerabilities.

In conclusion, as the semiconductor industry evolves, so must the security measures that protect it. This thesis demonstrates that while LL remains a powerful tool in the fight against IP theft and reverse engineering, it is far from secure. The contributions made here, including the novel attacks and hybrid defenses, represent critical steps forward in the ongoing effort to secure the integrity of IC designs. The challenges outlined in this work highlight the need for a dynamic and proactive approach to hardware security, ensuring that as attackers evolve their methods, defenders are equipped with the tools and strategies necessary to protect the core of modern technology.

References

[1] H.-C. Hung, Y.-C. Chiu, and M.-C. Wu, "Analysis of competition between idm and fabless–foundry business models in the semiconductor industry," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 3, pp. 254–260, 2017.

[2] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[3] C. Wiesen, S. Becker, M. Fyrbiak, N. Albartus, M. Elson, N. Rummel, and C. Paar, "Teaching hardware reverse engineering: Educational guidelines and practical insights," in *IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pp. 438–445, 2018.

[4] G. Zarrinchian, "A chip activation protocol for preventing ic recycling," *Microprocessors and Microsystems*, vol. 101, p. 104872, 2023.

[5] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[6] T. M. Supon, M. Seyedbarhagh, R. Rashidzadeh, and R. Muscedere, "A method to prevent hardware trojans limiting access to layout resources," *Microelectronics Reliability*, vol. 124, p. 114212, 2021.

[7] T. D. Perez and S. Pagliarini, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184013–184035, 2020.

[8] H. Chakraborty and R. Vemuri, "Combined split manufacturing and logic obfuscation based on emerging technologies at high level for secure 3d ic design," in *IEEE 67th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1403–1407, 2024.

[9] F. Koushanfar and G. Qu, "Hardware metering," in *38th Annual Design Automation Conference (DAC)*, p. 490–493, 2001.

[10] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, no. 1, pp. 51–63, 2012.

[11] G. Qu and L. Yuan, *Secure Hardware IPs by Digital Watermark*, pp. 123–141. Springer New York, 2012.

[12] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *Design Automation Conference (DAC)*, pp. 776–781, 1998.

[13] J. Zhang, "A practical logic obfuscation technique for hardware security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 1193–1197, 2016.

[14] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1069–1074, 2008.

[15] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *Design Automation Conference (DAC)*, pp. 83–89, 2012.

[16] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.

[17] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *J. Electron. Test.*, vol. 35, no. 3, pp. 273–291, 2019.

[18] J. Zhou and X. Zhang, "Generalized sat-attack-resistant logic locking," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 16, pp. 2581–2592, 2021.

[19] V. S. Rathor, M. Singh, K. S. Sahoo, and S. P. Mohanty, "Gatelock: Input-dependent key-based locked gates for sat resistant logic locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 2, pp. 361–371, 2024.

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

[21] H. M. Kamali, K. Z. Azar, F. Farahmandi, and M. Tehranipoor, "Advances in logic locking: Past, present, and prospects," *Cryptology ePrint Archive*, 2022.

[22] 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 (TCAD)*, vol. 35, no. 9, pp. 1411–1424, 2016.

[23] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately deobfuscating integrated circuits," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 95–100, 2017.

[24] F. Yang, M. Tang, and O. Sinanoglu, "Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) – Unlocked," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 10, pp. 2778–2786, 2019.

[25] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 8, no. 2, pp. 517–532, 2020.

[26] L. Aksoy, M. Yasin, and S. Pagliarini, "Kratt: Qbf-assisted removal and structural analysis attack against logic locking," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2024.

[27] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 2, pp. 199–207, 2019.

[28] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "Ttlock: Tenacious and traceless logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 166–166, 2017.

[29] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-sat era by locking & obfuscation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 13–18, 2020.

[30] F. Almeida, L. Aksoy, and S. Pagliarini, "Resaa: A removal and structural analysis attack against compound logic locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, pp. 1–13, 2025.

[31] 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 (TCAD)*, vol. 28, no. 10, pp. 1493–1502, 2009.

[32] L. Aksoy, Q.-L. Nguyen, F. Almeida, J. Raik, M.-L. Flottes, S. Dupuis, and S. Pagliarini, "High-level intellectual property obfuscation via decoy constants," in *IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–7, 2021.

[33] Z. Han, M. Yasin, and J. Rajendran, "Does Logic Locking Work with EDA Tools?," in *USENIX Security Symposium*, pp. 1055–1072, 2021.

[34] F. Almeida, L. Aksoy, Q.-L. Nguyen, S. Dupuis, M.-L. Flottes, and S. Pagliarini, "Resynthesis-based attacks against logic locking," in *24th International Symposium on Quality Electronic Design (ISQED)*, pp. 1–8, 2023.

[35] L. Aksoy, Q.-L. Nguyen, F. Almeida, J. Raik, M.-L. Flottes, S. Dupuis, and S. Pagliarini, "Hybrid protection of digital fir filters," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 6, pp. 812–825, 2023.

[36] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "IP Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 6, 2019.

[37] X. Yuan, J. Weng, C. Wang, and K. Ren, "Secure integrated circuit design via hybrid cloud," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 29, no. 8, pp. 1851–1864, 2018.

[38] A. Kahng, J. Lach, W. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design ip protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 20, no. 10, pp. 1236–1252, 2001.

[39] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pp. 166–171, 2009.

[40] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, "Security assurance for system-on-chip designs with untrusted ips," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 12, no. 7, pp. 1515–1528, 2017.

[41] M. R. Muttaki, S. Saha, H. M. Kamali, F. Rahman, M. Tehranipoor, and F. Farahmandi, "Rtlock: Ip protection using scan-aware logic locking at rtl," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2023.

[42] S. Muzaffar and I. A. M. Elfadel, "Logic locking of finite-state machines using transition obfuscation," in *IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2022.

[43] T. Hoque, R. S. Chakraborty, and S. Bhunia, "Hardware Obfuscation and Logic Locking: A Tutorial Introduction," *IEEE Design & Test*, vol. 37, no. 3, pp. 59–77, 2020.

[44] IEEE Standards Association, "IEEE Standard for Encryption and Management of Electronic Design Intellectual Property (IP)." `https://standards.ieee.org/ieee/1735/7237/`, 2014. IEEE P1735-2014.

[45] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *IEEE/ACM International Conference on Computer-Aided Design*, pp. 674–677, 2008.

[46] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[47] M. Nagata, T. Miki, and N. Miura, "Physical attack protection techniques for ic chip level hardware security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 5–14, 2022.

[48] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 363–381, 2009.

[49] Y.-C. Chen, W.-L. Wang, and M.-S. Hwang, "Rfid authentication protocol for anti-counterfeiting and privacy protection," in *The 9th International Conference on Advanced Communication Technology*, vol. 1, pp. 255–259, 2007.

[50] C. Profentzas, M. Günes, Y. Nikolakopoulos, O. Landsiedel, and M. Almgren, "Performance of secure boot in embedded systems," in *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 198–204, 2019.

[51] V. Lakafosis, A. Traille, H. Lee, G. Orecchini, E. Gebara, M. M. Tentzeris, J. Laskar, G. DeJean, and D. Kirovski, "An rfid system with enhanced hardware-enabled authentication and anti-counterfeiting capabilities," in *IEEE MTT-S International Microwave Symposium*, pp. 840–843, 2010.

[52] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[53] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *IEEE 20th International On-Line Testing Symposium (IOLTS)*, pp. 49–54, 2014.

[54] C. Pilato, F. Regazzoni, R. Karri, and S. Garg, "Tao: techniques for algorithm-level obfuscation during high-level synthesis," in *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, (New York, NY, USA), Association for Computing Machinery, 2018.

[55] K. Zamiri Azar, H. M. Kamali, S. Roshanisefat, H. Homayoun, C. P. Sotiriou, and A. Sasan, "Data flow obfuscation: A new paradigm for obfuscating circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 643–656, 2021.

[56] Z. U. Abideen, S. Gokulanathan, M. J. Aljafar, and S. Pagliarini, "An overview of fpga-inspired obfuscation techniques," *ACM Comput. Surv.*, vol. 56, no. 12, 2024.

[57] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall, "Interlocking obfuscation for anti-tamper hardware," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, 2013.

[58] C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg, and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 7, pp. 1306–1318, 2021.

[59] L. Aksoy, M. Yasin, and S. Pagliarini, "Cac 2.0: A corrupt and correct logic locking technique resilient to structural analysis attacks," in *IEEE 25th Latin American Test Symposium (LATS)*, pp. 1–6, 2024.

[60] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," in *ACM Conference on Computer and Communications Security (ACM CCS)*, pp. 709–720, 2013.

[61] J. Mellor, A. Shelton, M. Yue, and F. Tehranipoor, "Attacks on logic locking obfuscation techniques," in *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–6, 2021.

[62] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[63] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2020, no. 1, pp. 175–202, 2019.

[64] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-Based Perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 12, pp. 4439–4452, 2020.

[65] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 114–119, 2021.

[66] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," in *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 189–210, 2017.

[67] A. Sengupta, N. Limaye, and O. Sinanoglu, "Breaking CAS-Lock and Its Variants by Exploiting Structural Traces," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2021, no. 3, p. 418–440, 2021.

[68] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *Design Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 936–939, 2019.

[69] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 17, pp. 744–759, 2022.

[70] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 85–90, 2018.

[71] D. Divyanshu, R. Kumar, D. Khan, S. Amara, and Y. Massoud, "Logic locking using emerging 2t/3t magnetic tunnel junctions for hardware security," *IEEE Access*, vol. 10, pp. 102386–102395, 2022.

[72] 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 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 405–410, 2018.

[73] M. John, A. Hoda, R. Chouksey, and C. Karfa, "Sat based partial attack on compound logic locking," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2020.

[74] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-sat: Fault-aided sat-based attack on compound logic locking techniques," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1166–1171, 2021.

[75] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Great Lakes Symposium on VLSI (GLSVLSI)*, p. 179–184, 2017.

[76] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based Bit-Flipping Attack on Logic Encryptions," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 629–632, 2018.

[77] A. Raj, N. Avula, P. Das, D. Sisejkovic, F. Merchant, and A. Acharyya, "Deepattack: A deep learning based oracle-less attack on logic locking," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2023.

[78] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 8, pp. 1529–1542, 2021.

[79] D. Sisejkovic, F. Merchant, L. M. Reimann, and R. Leupers, "Deceptive logic locking for hardware integrity protection against machine learning attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 6, pp. 1716–1729, 2022.

[80] D. Sisejkovic, L. M. Reimann, E. Moussavi, F. Merchant, and R. Leupers, "Logic locking at the frontiers of machine learning: A survey on developments and opportunities," in *IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021.

[81] L. Alrahis, S. Patnaik, J. Knechtel, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "Unsail: Thwarting oracle-less machine learning attacks on logic locking," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 16, pp. 2508–2523, 2021.

[82] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The key is left under the mat: On the inappropriate security assumption of logic locking schemes," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 262–272, 2020.

[83] 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 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 405–410, 2018.

[84] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers (TC)*, vol. 64, no. 2, pp. 410–424, 2015.

[85] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "Interlock: An intercorrelated logic and routing locking," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.

[86] M. Yasin, C. Zhao, and J. J. Rajendran, "Sfll-hls: Stripped-functionality logic locking meets high-level synthesis," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–4, 2019.

[87] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Great Lakes Symposium on VLSI (GLSVLSI)*, p. 173–178, 2017.

[88] H.-Y. Chiang, Y.-C. Chen, D.-X. Ji, X.-M. Yang, C.-C. Lin, and C.-Y. Wang, "Looplock: Logic optimization-based cyclic logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 10, pp. 2178–2191, 2020.

[89] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using puf-based logic," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 270–271, 2014.

[90] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "Tga: An oracle-less and topology-guided attack on logic locking," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, p. 75–83, 2019.

[91] W. Zeng, A. Davoodi, and R. O. Topaloglu, "Obfusx: Routing obfuscation with explanatory analysis of a machine learning attack," in *26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 548–554, 2021.

[92] L. Alrahis, S. Patnaik, F. Khalid, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "Gnnunlock: Graph neural networks-based oracle-less unlocking scheme for provably secure logic locking," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 780–785, 2021.

[93] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 663–698, 1985.

[94] F. Corno, M. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design & Test of Computers (D&T)*, vol. 17, no. 3, pp. 44–53, 2000.

[95] B. Tan, R. Karri, N. Limaye, A. Sengupta, O. Sinanoglu, M. M. Rahman, S. Bhunia, D. Duvalsaint, R. D., Blanton, A. Rezaei, Y. Shen, H. Zhou, L. Li, A. Orailoglu, Z. Han, A. Benedetti, L. Brignone, M. Yasin, J. Rajendran, M. Zuzak, A. Srivastava, U. Guin, C. Karfa, K. Basu, V. V. Menon, M. French, P. Song, F. Stellari,

G.-J. Nam, P. Gadfort, A. Althoff, J. Tostenrude, S. Fazzari, E. Breckenfeld, and K. Plaks, "Benchmarking at the frontier of hardware security: Lessons from logic locking." arXiv preprint arXiv:2006.06806, 2020.

[96] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the approximation resiliency of logic locking and ic camouflaging schemes," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 2, pp. 347–359, 2019.

[97] L. Aksoy, A. Hepp, J. Baehr, and S. Pagliarini, "Hardware obfuscation of digital fir filters," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 68–73, 2022.

[98] A. Alaql, "Scope." Available: `https://github.com/alaql89/SCOPE`.

[99] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *Design, Automation and Test in Europe Conference (DATE)*, pp. 540–545, 2019.

[100] K. Shamsi, "Netlist Encryption and Obfuscation Suite." `https://bitbucket.org/kavehshm/neos/src/master/`, 2021.

[101] L. Aksoy, "Qatt query attack." Available: `https://github.com/leventaksoy/qatt`.

Acknowledgements

I would like to express my deepest gratitude to everyone who supported me throughout this PhD journey. First and foremost, I am profoundly grateful to my supervisors, Prof. Dr. Samuel Pagliarini and Dr. Levent Aksoy. Samuel introduced me to Hardware Security, and despite our disagreements and challenging moments, his extensive knowledge, guidance, and dedication made this research possible. Levent, thank you for your exceptional patience, valuable feedback, and unwavering friendship. Your technical insights and support have significantly shaped my work. I would also like to thank Prof. Dr. Jaan Raik for his support as my co-supervisor during part of this journey. Additionally, I extend my sincere thanks to all members of the Centre for Hardware Security and the Department of Computer Systems for their collaboration and encouragement.

I am grateful to Tallinn University of Technology for providing the essential resources, academic structure, and financial support required to complete this thesis. I acknowledge the generous funding from the ICT programme and the SAFEST project, both supported by the European Union, as well as the MOBERC35 project funded by ETAG.

Special thanks go to my colleagues Cesar, Tiago, Malik, and Zain, whose companionship, insightful discussions, and mutual support have made this journey memorable and enriching. I am particularly grateful to my therapist, Michele, whose professional guidance has been vital in navigating the emotional challenges of this PhD.

Heartfelt thanks to my friends from the Baku group—Paola, Felipe, Nemaila, Natalia, Marcelo, and Patrik—for their friendship, joy, and strength during challenging times. To Carlos, thank you for your unwavering support, constant presence, and genuine care throughout this journey. I also extend warm appreciation to my wonderful friends in Germany—Milena, Nicolas, and Esperanza—and to my cherished childhood friends, Claudia, Isabela, Carlete, Daniel, and Wendyel, whose friendship I deeply value.

Most importantly, I am profoundly thankful to my family—my mother, Zefinha, my sister, Fernanda, and my late father, Jaime. Their unconditional love, encouragement, and belief in me have been the foundation of my achievements. I dedicate this thesis to my father, Jaime, whose memory inspires and guides me.

Abstract

Advanced Hardware Protection Mechanisms: A Study on Logic Locking and Circuit Obfuscation Techniques

This thesis explores innovative protection methods for digital integrated circuits (ICs), focusing on advanced logic locking (LL) and obfuscation strategies to counteract unauthorized access and intellectual property theft. The study introduces several notable contributions. Firstly, a hybrid protection technique for digital filters combines hardware obfuscation and LL to enhance resilience against oracle-guided and oracle-less attacks, addressing direct and transposed filter forms vulnerable to traditional attack vectors. Experimental results demonstrate that this hybrid approach effectively prevents key exposure, achieving a 0% success rate for attackers, in contrast to traditional LL methods that suffer up to 100% key recovery. Secondly, the research develops a resynthesis-based attack model to exploit synthesis-stage vulnerabilities within the IC design flow, demonstrating how adversaries may bypass traditional defenses without relying on an oracle, a crucial aspect for practical attack scenarios. By examining synthesis manipulation and structural diversity, this work underlines potential risks and calls for improved practices in secure design, achieving key recovery rates ranging from approximately 47% to 73%. Finally, the thesis presents the RESAA framework—a sophisticated attack model that breaks compound logic locking (CLL), including configurations resilient to satisfiability attacks. RESAA strategically classifies designs locked by CLL, identifies critical gates (CGs), and executes various attacks to uncover secret keys. The approach partitions the design based on the CG, thereby exposing structural weaknesses within CLL mechanisms and achieving secret key recovery rates ranging from 68% to 97% across benchmark circuits. This research highlights the pressing need for evolving IC protection approaches to meet the growing complexity of security threats and ensure that defense mechanisms remain robust and flexible. Future research directions could explore the integration of artificial intelligence (AI) for adaptive responses, ultimately paving the way for more resilient and self-correcting security solutions in dynamic operational environments.

Kokkuvõte

Täiustatud riistvara kaitsemehhanismid: uuring loogikalukustamise ja hägustamise tehnikate kohta

Käesolev doktoritöö uurib uuenduslikke kaitsemeetodeid digitaalsete integraallülituste (IL) jaoks, keskendudes täiustatud loogikalukustamisele (LL) ja hägustamisstrateegiatele, et takistada volitamata juurdepääsu ja intellektuaalomandi vargust. Uuring pakub mitmeid märkimisväärseid uuendusi. Esiteks ühendab digitaalsete filtrite hübriidne kaitsetehnika riistvara hägustamise ja LL-i, et suurendada vastupidavust nii oraakliga juhitud kui ka oraaklita rünnakute vastu, käsitledes otseseid ja transponeeritud filtrivorme, mis on haavatavad traditsioonilistele ründevektoritele. Eksperimendid näitavad, et see hübriidne lähenemisviis takistab tõhusalt võtmete avalikustamist, viies rünnete edukuse 0%ni, samas kui traditsioonilised LL-meetodid kannatavad kuni 100% võtmete taastamise all. Teiseks arendab uurimus sünteesipõhise ründe mudeli, et ära kasutada sünteesietapi haavatavusi IL disainivoos, näidates, kuidas vastased võivad traditsioonilistest kaitsemehhanismidest mööda minna ilma oraaklit kasutamata, mis on praktiliste ründestsenaariumide jaoks oluline aspekt. Uurides sünteesimanipulatsiooni ja struktuurilist mitmekesisust, rõhutab see töö võimalikke riske ja kutsub üles parandama turvalise disaini praktikaid, saavutades võtmete taastamise määrad vahemikus 47-73%. Lõpetuseks esitleb doktoritöö RESAA raamistikku—keerukat ründe mudelit, mis murrab kombineeritud loogikalukustamise (KLL), sealhulgas konfiguratsioone, mis on vastupidavad SAT rünnakutele. RESAA klassifitseerib strateegiliselt KLL-iga lukustatud disainid, tuvastab kriitilised loogikalülitused ja viib läbi erinevaid rünnakuid, et paljastada salajasi võtmeid. Lähenemisviis jagab disaini kriitilise loogikalülituse alusel, paljastades seeläbi KLL-mehhanismide struktuursed nõrkused ja saavutades salajaste võtmete taastamise määrad vahemikus 68-97% üle võrdlusskeemide. Uurimus toob esile pakilise vajaduse arendada IL kaitsemeetodeid, et tulla toime turvaohtude kasvava keerukusega ja tagada, et kaitsemehhanismid jäävad tugevaks ja paindlikuks. Tulevased uurimissuunad võiksid uurida tehisintellekti integreerimist adaptiivsete reaktsioonide saavutamiseks, sillutades teed vastupidavamatele ja ennast parandavatele turvalahendustele dünaamilistes töökeskkondades.

Appendix A

[I]
L. Aksoy, Q. -L. Nguyen, F. Almeida, J. Raik, M. -L. Flottes, S. Dupuis, and S. Pagliarini, "Hybrid Protection of Digital FIR Filters," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 6, pp. 812-825, 2023.

Hybrid Protection of Digital FIR Filters

Levent Aksoy, *Member, IEEE,* Quang-Linh Nguyen, Felipe Almeida, Jaan Raik, *Member, IEEE,* Marie-Lise Flottes, *Member, IEEE,* Sophie Dupuis, *Member, IEEE,* and Samuel Pagliarini, *Member, IEEE*

***Abstract*—A digital Finite Impulse Response (FIR) filter is a ubiquitous block in digital signal processing applications and its behavior is determined by its coefficients. To protect filter coefficients from an adversary, efficient obfuscation techniques have been proposed, either by hiding them behind decoys or replacing them by key bits. In this article, we initially introduce a query attack that can discover the secret key of such obfuscated FIR filters, which could not be broken by existing prominent attacks. Then, we propose a first of its kind hybrid technique, including both hardware obfuscation and logic locking using a point function for the protection of parallel direct and transposed forms of digital FIR filters. Experimental results show that the hybrid protection technique can lead to FIR filters with higher security while maintaining the hardware complexity competitive or superior to those locked by prominent logic locking methods. It is also shown that the protected multiplier blocks and FIR filters are resilient to existing attacks. The results on different forms and realizations of FIR filters show that the parallel direct form FIR filter has a promising potential for a secure design.**

***Index Terms*—hardware obfuscation, logic locking, oracle-less and oracle-guided attacks, constant multiplications, FIR filters, direct and transposed forms.**

I. Introduction

Due to the increase in the design complexity of Integrated Circuits (ICs) and the rising costs of chip fabrication at advanced technology nodes, the IC supply chain has become heavily specialized and globalized [1]. Design houses have been combining their Intellectual Properties (IPs) with many others purchased from third-parties and resorting to *untrusted* foundries for fabrication. Although such globalization reduces the overall cost of producing an IC, it leads to serious security threats – especially for IPs – such as piracy, overuse, modification, and reverse engineering [2]. Over the years, IP protection has received a significant amount of interest and efficient methods, including watermarking [3], digital rights management [4], metering [5], and hardware obfuscation [6], have been introduced. Among these techniques, only hardware obfuscation can prevent IP theft, while the others are useful to prove the IP owner and reveal the IP owner's rights during a litigation process. Hardware obfuscation aims to make the design less clear and hard to understand for an adversary, by hiding the design content using structural transformations, locking the design functionality using additional logic with key bits, and exploiting camouflaged gates [6].

Digital filtering is frequently used in Digital Signal Processing (DSP) applications and Finite Impulse Response (FIR) filters are generally preferred due to their stability and linear phase property [7]. Since filter coefficients determine the filter behavior, they are actually an IP and need protection from reverse engineering by an adversary. Although there exist many efficient high-level and behavioral obfuscation methods proposed for protecting IPs [8]–[13], digital FIR filters require specialized obfuscation techniques, since they should behave according to their specifications, such as pass-band and stopband frequencies and ripples [14]. However, there exist only a limited number of techniques proposed to obfuscate DSP circuits and especially, digital filters [15]–[18]. The technique of [15] generates the desired filter and also its obfuscated versions, grouped in two categories as meaningful and unmeaningful in terms of filter behavior, using high-level transformations, and combines these realizations using a key-based finite state machine and a reconfigurator. To make the reverse engineering of coefficients harder for an end-user, adding input and output noises was proposed in [16]. Recently, we introduced a hardware obfuscation technique that hides the filter coefficients behind decoys [17], [18]. In [17], decoys can be selected based on their Hamming distance to reduce the hardware complexity or chosen randomly to increase the corruption at the filter output. Since an obfuscated FIR filter may still generate the desired behavior under a wrong key in [17], decoys are selected in such a way that the obfuscated filter presents the desired behavior only when the secret key is provided in [18]. To do so, the lower and upper bounds of each filter coefficient are found and decoys are selected beyond these bounds. In [17], [18], the folded design of an FIR filter is considered as a case study and its Time-Multiplexed Constant Multiplication (TMCM) block is obfuscated at Register-Transfer Level (RTL).

In this article, we initially introduce the query attack, which can discover the original filter coefficients hidden behind decoys [17], [18] or replaced by key bits [9]. Then, we propose a hybrid technique, which includes both hardware obfuscation and logic locking, for the protection of digital FIR filters. To do so, first, we describe a defense technique that obfuscates the multiplier blocks of parallel direct and transposed forms of an FIR filter, i.e., Constant Array Vector Multiplication (CAVM) and Multiple Constant Multiplication (MCM), respectively, using decoys. We also present their hardware-efficient realizations with and without multipliers. Second, we enhance this obfuscation technique by locking the obfuscated design using

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 952252 (SAFEST) and by the project MOOSIC ANR-18-CE39-0005 of the French National Research Agency (ANR).

L. Aksoy, F. Almeida, J. Raik, and S. Pagliarini are with the Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia (e-mail: {levent.aksoy, felipe.almeida, jaan.raik, samuel.pagliarini}@taltech.ee.)

Q.-L. Nguyen is with STMicroelectronics, Grenoble, France (e-mail: quanglinh.nguyen@st.com.)

M.-L. Flottes and S. Dupuis are with LIRMM, University of Montpellier, CNRS, Montpellier, France (e-mail: {marie-lise.flottes, sophie.dupuis}@lirmm.fr.)

a point function to make the protected design resilient to well-known attacks and by thwarting the query attack to determine the secret key. The hybrid protection technique works at RTL and can be easily adapted to any application including constant multiplications, such as image and video processing and neural networks. The main contributions of this article are as follows:

- Query attack developed for breaking designs generated by constant obfuscation techniques;
- Secure hybrid technique, consisting of hardware obfuscation and logic locking, developed for the protection of FIR filters with different forms and realizations;
- Comprehensive results on obfuscation and logic locking of FIR filters in terms of hardware complexity, attack resiliency, and filter behavior.

Experimental results clearly show that the proposed hybrid protection technique leads to FIR filter designs with higher security and competitive hardware complexity when compared to previously proposed hardware obfuscation and logic locking methods. As an interesting outcome of this work, we show that the parallel direct form filter has better resiliency properties than other FIR filter forms and realizations.

The remainder of this article is organized as follows: Section II presents background concepts. The query attack is described in Section III and the hybrid protection method is introduced in Section IV. Experimental results are presented in Section V. Further discussions on how other techniques may identify the original filter coefficients are given in Section VI. Finally, Section VII concludes the article.

II. Background

This section initially presents frequently used notations and then, gives details on digital FIR filters and multiplierless constant multiplications. Finally, it summarizes related work.

A. Notations

Table I presents notations of important parameters used in the description of obfuscation and logic locking techniques.

TABLE I
SUMMARY OF NOTATIONS

c	Constant/filter coefficient
C	Constant array/set
n	Number of constants/filter coefficients
mbw	Maximum bit-width of constants/filter coefficients
X	Input variable/filter input
ibw	Bit-width of the input variable/filter input
Y	Output variable/filter output
k	Key bit
$\boldsymbol{K}$	Secret key
p	Total number of key bits
v	Number of key bits for obfuscation
w	Number of key bits for logic locking

B. Digital FIR Filters

The FIR filter output $Y(j)$ is given as $\sum_{i=0}^{n-1} c_i \cdot X(j-i)$, where n is the filter length, c_i is the i^{th} filter coefficient, and $X(j-i)$ is the i^{th} previous filter input with $0 \leq i \leq n-1$. Fig. 1 shows the parallel and folded realizations of an FIR filter. Note that the filter output is obtained in a single clock cycle in a parallel design, as shown in the direct and transposed

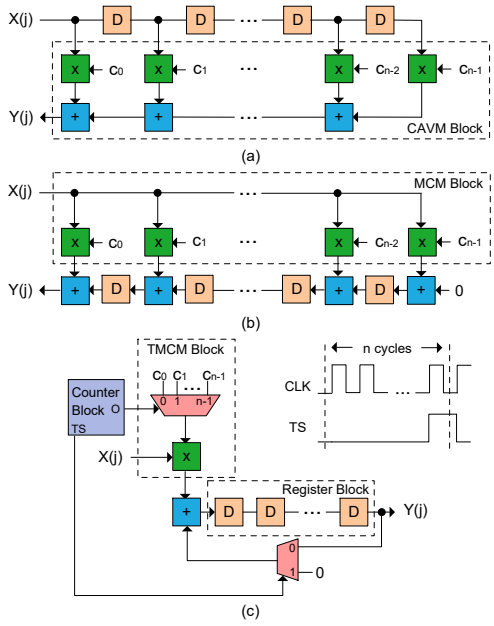


Fig. 1. Designs of an FIR filter: (a) parallel direct form; (b) parallel transposed form; (c) folded transposed form, where the counter counts from 0 to $n-1$.

forms in Figs. 1(a)-(b). On the other hand, the folded realization leads to a design with the least hardware complexity, since the common operations are re-used. However, it requires n clock cycles to compute the filter output, as shown in Fig. 1(c).

C. Multiplierless Design of Constant Multiplications

Multiplication of constant(s) by variable(s) is a ubiquitous and crucial operation in many DSP applications. Among others presented in [24], the CAVM, MCM, and TMCM blocks can be used in the design of a filter, as shown in Fig. 1. They are defined as follows:

1) The *CAVM operation* implements the multiplication of a $1 \times n$ constant array C by an $n \times 1$ input vector X, i.e., $Y = \sum_i c_i X_i$ with $1 \leq i \leq n$.
2) The *MCM operation* computes the multiplication of a set of n constants C by a single variable X, i.e., $Y_i = c_i X$ with $1 \leq i \leq n$.
3) The *TMCM operation* realizes the multiplication of a constant selected from a set of n constants C by a single variable X at a time, i.e., $Y = c_i X$ with $1 \leq i \leq n$.

Since the constants are determined beforehand, these constant multiplications can be realized using addition, subtraction, and shift operations under the shift-adds architecture. Note that parallel shifts can be implemented virtually for free in hardware using only wires. A straightforward shift-adds design technique, called the Digit-Based Recoding (DBR) [19], can realize constant multiplications in two steps: i) define the constants under a particular number representation, e.g., binary; ii) for the nonzero digits in the representation of constants, shift the input variables according to digit positions and add/subtract the shifted variables with respect to digit

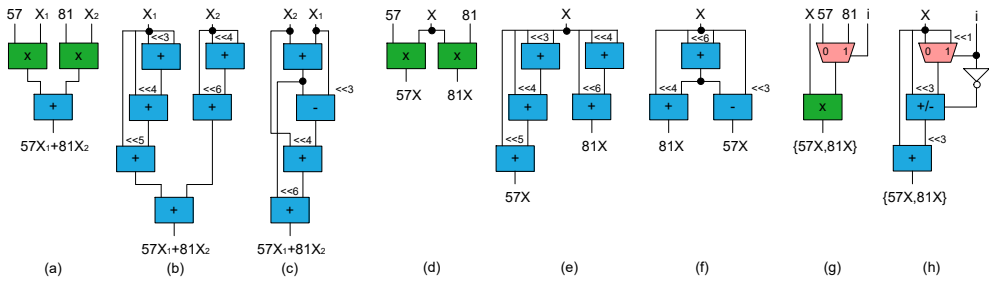


Fig. 2. Realizations of the CAVM (a-c), MCM (d-f), and TMCM (g-h) blocks including constants 57 and 81: (a) using multipliers; (b) the DBR method [19]; (c) the method of [20]; (d) using multipliers; (e) the DBR method [19]; (f) the method of [21]; (g) using a multiplier; (h) the method of [22].

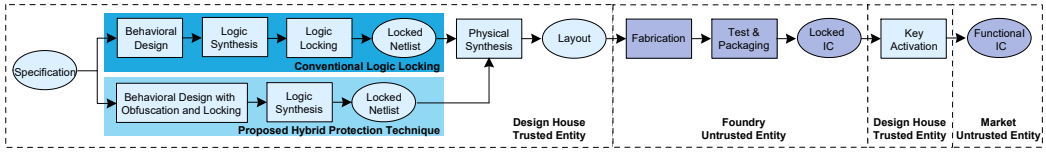


Fig. 3. Conventional logic locking and proposed hybrid protection technique in the IC design flow (adapted from [23]).

values. Furthermore, the number of operations can be reduced by maximizing the sharing of common subexpressions among constant multiplications [20]–[22], [25], [26].

As a simple example, consider the CAVM, MCM, and TMCM blocks realizing constant multiplications, where C includes $57 = (111001)_{bin}$ and $81 = (1010001)_{bin}$. These constant multiplications are shown in Fig. 2. Note that the adder/subtractor shown in Fig. 2(h) behaves as an adder or a subtractor when its select input is 0 or 1, respectively. Observe from Figs. 2(b)-(c) and (e)-(f) that the sharing of common subexpressions can lead to a significant reduction under the shift-adds architecture in terms of the number of operations with respect to the DBR method.

D. Related Work

Hardware obfuscation can take place at different stages in the IC design flow, e.g., high-level synthesis [11], RTL [9], gate-level [27], and layout level [28]. In hardware obfuscation, locking the design functionality is a common practice. Fig. 3 presents conventional logic locking applied at gate-level in the IC design flow. Note that after the layout of the locked netlist is shipped to the foundry without revealing the secret key, the locked IC is produced and delivered back to the design house. Then, values of the secret key are stored in a tamper-proof memory and the functional IC is sent to the market.

1) Defenses: Earlier logic locking methods have been applied at gate-level. After the introduction of the concept of Random Logic Locking (RLL) using XOR/XNOR gates in [27], many works focused on different types of key logic, such as AND/OR, multiplexors (MUXes), and look-up tables, taking into account the hardware complexity of the locked circuit [29]. However, the satisfiability (SAT)-based attack [30] overcame all the defenses existing at that time. To thwart the SAT-based attack and its variants, circuits have been locked using a point function that forces these attacks to explore an exponential number of queries [23], [31]–[35]. Moreover, the obfuscation of a locked design is considered in [36].

However, as mentioned in [8], at a higher level in the IC design flow, the selection of critical blocks of the design to be obfuscated gets easier, the exploration of tradeoffs between overhead and attack resiliency becomes more efficient, and the optimization of the obfuscated design is more effective. Recently, high-level and behavioral obfuscation techniques have been presented in [8]–[12]. Related to digital FIR filters including a large number of constants, filter coefficients are obfuscated by replacing their bits by key bits in [9], [11].

We note that our proposed hybrid protection technique works at one level higher than the gate-level, i.e., at RTL, as also shown in Fig. 3.

2) Attacks: In logic locking, there are generally two threat models, namely oracle-less (OL) and oracle-guided (OG). In the OL threat model, only the gate-level netlist of the locked circuit is available to an adversary. In the OG threat model, it is assumed that an adversary can also obtain the functional IC programmed with the secret key from the market and use it as an oracle to apply inputs and observe outputs. Hence, in this model, the adversary has both the netlist of the locked circuit and the functional IC.

Under the OL threat model, due to the limited information available to the adversary, patterns in the structure of the locked netlist are studied using statistical analysis, Automated Test Pattern Generation (ATPG), and machine learning [37]–[40]. Structural attacks, which identify and remove the logic inserted by a logic locking method, are proposed in [41]–[43].

Under the OG threat model, the ATPG-based attack of [44] leverages testing principles, such as justification and sensitization while finding the secret key. The SAT-based attack [30] iteratively finds Distinguishing Input Patterns (DIPs) that rule out wrong keys and achieves decryption as shown in Algorithm 1. It generates two locked circuits with the same inputs (X), but two different keys (K_1 and K_2) described in a Conjunctive Normal Form (CNF) formula in a SAT problem (line 2). Then, it finds a DIP, which generates different outputs on these circuits, using a SAT solver (line 4) and computes

Algorithm 1 The SAT-based attack [30]

Inputs: Locked circuit *LC* and *oracle*.
Output: Secret key $\boldsymbol{K}$.

1: $i := 1$ ▷ Number of iterations
2: $F_1 = LC(X, K_1, Y_1) \wedge LC(X, K_2, Y_2)$
3: **while** $sat[F_i \wedge (Y_1 \neq Y_2)]$ **do**
4: $\quad X_i^d := sat\_assignment_X[F_i \wedge (Y_1 \neq Y_2)]$
5: $\quad Y_i^d := oracle(X_i^d)$
6: $\quad F_{i+1} := F_i \wedge LC(X_i^d, K_1, Y_i^d) \wedge LC(X_i^d, K_2, Y_i^d)$
7: $\quad i := i + 1$
8: $\boldsymbol{K} := sat\_assignment_{K_1}(F_i)$

the output based on the found DIP using the oracle (line 5). It adds the Boolean equations including key bits into the SAT problem, which are obtained after inserting the values of these inputs and outputs into these circuits (line 6). This process is iterated until the SAT problem becomes unsatisfiable (line 3), meaning that there exists no DIP to distinguish wrong keys from the secret key. Finally, it determines the secret key as the one found in the last iteration (line 8).

In a similar fashion, the SAT-based attack of [45] eliminates at least 2 DIPs in a single iteration. A Satisfiability Modulo Theory (SMT) solver is used instead of a SAT solver, providing more flexibility while encoding the problem [46], [47]. The so-called approximate attack of [48] aims for approximate functional recovery. The SAT-based attack of [49] achieves sequential deobfuscation using dynamic simplifications of key conditions. The attack of [50] discovers the vulnerabilities of the SAT-resilient logic locking methods of [31], [32]. In [51], a generic framework is developed to attack compound locking techniques. A security diagnosis tool, which can evaluate the structural vulnerability of a design locked by a provably secure logic locking technique, is introduced in [52].

III. The Query Attack

The SAT-based attack [30] presented in Algorithm 1 guarantees that the found values of **all** key bits are equal to those of the secret key. To do so, it may use a large number of queries that are required to eliminate all the wrong keys. On the other hand, our query attack proves that the found value of a **single** key bit is equal to that of the associated one in the secret key. To do so, it uses a small number of queries that make each key bit observable at a primary output. Hence, it slightly increases the SAT problem size when compared to the SAT-based attack. Thus, it can easily cope with circuits including a large number of gates and key bits [53] and logic structures, such as a multiplier and a tree of AND gates [30], which the SAT-based attack generally finds hard to handle. In this section, we initially describe the proposed query attack and then, present its results on obfuscated designs.

A. Description

Our proposed OG SAT-based query attack is described in Algorithm 2. It initially finds queries using two strategies (line 1). In the first one, an ATPG tool is used to find the test patterns for the stuck-at-fault of each key bit on the locked circuit and the values of the related primary inputs are stored

Algorithm 2 The query attack

Inputs: Locked circuit *LC* and *oracle*.
Output: Proven values of the secret key $\boldsymbol{K}$.

1: $Q := find\_queries(LC)$
2: $F = LC(X, K, Y)$
3: **for** i := 1 to $2p$ **do**
4: $\quad Y_i := oracle(Q_i)$
5: $\quad F := F \wedge LC(Q_i, K, Y_i)$
6: $K := sat\_assignment_K(F)$
7: **for** i := 0 to $p - 1$ **do**
8: $\quad$ **if** $unsat[F \wedge \overline{K_i}]$ **then**
9: $\quad\quad \boldsymbol{K_i} = K_i$
10: **for** i := 0 to $p - 2$ **do**
11: $\quad$ **for** $j := i + 1$ to $p - 1$ **do**
12: $\quad\quad$ **if** $undefined(\boldsymbol{K_i})$ & $undefined(\boldsymbol{K_j})$ **then**
13: $\quad\quad\quad$ **if** $unsat[F \wedge (K_i \neq K_j)]$ **then**
14: $\quad\quad\quad\quad \boldsymbol{K_i} = \boldsymbol{K_j}$
15: $\quad\quad\quad$ **else if** $unsat[F \wedge (K_i \neq \overline{K_j})]$ **then**
16: $\quad\quad\quad\quad \boldsymbol{K_i} = \overline{\boldsymbol{K_j}}$

as queries. The aim of this strategy is to find input patterns that can propagate each key bit to a primary output, making it observable. In the second one, queries are obtained randomly. The aim of this strategy is to find input patterns that may make multiple key bits observable at primary outputs. In our experiments, we generate a total of $2p$ queries, where p denotes the total number of key bits.

Then, the locked circuit is described in a CNF formula $\mathbb{F}$ by expressing each gate in its CNF (line 2). For each query (lines 3-5), it is applied to the oracle and the values of primary outputs are obtained (line 4). Then, the related input and output values are assigned to the associated nets in the locked circuit, the constant values of these nets are propagated, and the Boolean equations including key bits are derived in a CNF formula and added into $\mathbb{F}$ (line 5).

After all the queries are considered, the SAT problem $\mathbb{F}$ is solved using a SAT solver and the values of key bits are determined (line 6). Note that the locked circuit with the found values of key bits behaves exactly the same as the oracle under the given queries, but not under all possible input values. Hence, the found key is not guaranteed to be the secret key.

However, the found value of a key bit can be proven correct by using the concept of *proof by contradiction*. To do so, for each key bit (lines 7-9), the complement of its found value is added into $\mathbb{F}$ and the SAT solver is run. If there exists no solution to $\mathbb{F}$, i.e., the SAT problem is unsatisfiable, the value of the related key bit in the secret key is proven to be the one in the found solution.

As an example, consider the majority circuit in Fig. 4(a) and suppose that it is locked using XOR/XNOR gates as given in Fig. 4(b). Assume that a query is found as $x_1x_2x_3 = 000$ and thus, the value of its output y is obtained as 0 using the oracle. After propagating these values on the locked circuit, a Boolean equation $\overline{k_0} \vee k_1 = 0$, i.e., $k_0 \wedge \overline{k_1}$ in CNF, is obtained as shown in Fig. 4(c). In the SAT solution, the key bit values are found as $k_1k_0 = 01$. Note also that these are the proven key values since a SAT solver guarantees that there exists no solution to the SAT problem $\mathbb{F}$ when it is extended by either the constraint $k_0 = 0$, i.e., $\overline{k_0}$ in CNF or $k_1 = 1$, i.e., k_1 in

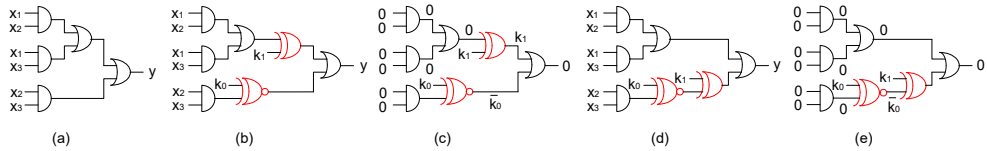


Fig. 4. Examples on the query attack: (a) majority circuit; (b)-(c) a locked majority circuit; (d)-(e) another locked majority circuit.

TABLE II
DETAILS ON FIR FILTERS AND THEIR TMCM BLOCKS.

Filter	Filter Details		TMCM Details	
	n	mbw	$\#in$	$\#out$
Mirzaei10a	71	15	39	47
LimYu07	121	14	39	46
Mirzaei10b	151	15	40	47

TABLE III
ATTACKS ON OBFUSCATED TMCM BLOCKS.

Filter	p	Architecture	Attacks			
			SAT	ATPG	Query	
			time	time	prv	time
Mirzaei10a	128	TMCM-MUL [17]	OoT	OoT	128	212
		TMCM-CRK [9]	OoT	OoT	128	341
LimYu07	128	TMCM-MUL [17]	OoT	OoT	128	112
		TMCM-CRK [9]	OoT	OoT	128	365
Mirzaei10b	256	TMCM-MUL [17]	OoT	OoT	256	809
		TMCM-CRK [9]	OoT	OoT	256	852

CNF, due to a conflict with the found Boolean equation, i.e., $k_0 \wedge \overline{k_1}$ in CNF.

We note that the query attack is also capable of proving if the value of a key bit, k_i, is equal to the value of another key bit, k_j, or its opposite (lines 10-16). To do so, we extend the SAT problem with $k_i \neq k_j$, i.e., $(k_i \vee k_j) \wedge (\overline{k_i} \vee \overline{k_j})$ in CNF, and $k_i \neq \overline{k_j}$, i.e., $(k_i \vee \overline{k_j}) \wedge (\overline{k_i} \vee k_j)$ in CNF, respectively, where $i < j$ and $0 \leq i, j \leq p-1$. We run the SAT solver and check if the SAT problem is unsatifiable. In this case, relations between two key bits are found independent of their values.

Returning back to our majority circuit, consider its another locked version given in Fig. 4(d). Assume that a query is again found as $x_1x_2x_3 = 000$ and hence, the output y is computed as 0. Thus, after the propagation of input and output values as shown in Fig. 4(e), a Boolean equation $\overline{k_0} \oplus k_1 = 0$, i.e., $(k_0 \vee k_1) \wedge (\overline{k_0} \vee \overline{k_1})$ in CNF, is found. In the SAT solution, the key bit values are found as $k_1k_0 = 10$. Although the actual values of key bits could not be proven, it is found that k_0 and k_1 have opposite values after the SAT problem is extended with the Boolean equation $k_0 \neq \overline{k_1}$ and it becomes unsatisfiable. Hence, the values of key bits $k_1k_0 = 10$ or $k_1k_0 = 01$ in the locked design lead to the original majority circuit.

B. Results

First, three FIR filters with a large number of coefficients and a large bit-width of filter input and coefficients were used to demonstrate the performance of the query attack. They were taken from [54]. Table II presents their details, where n and mbw are the number and maximum bit-width of coefficients, respectively. The folded realization of these filters was considered. Table II presents details on their TMCM blocks, where *#in* and *#out* are respectively their number of inputs and outputs when the bit-width of the input variable, i.e., ibw, is set to 32. These TMCM blocks were obfuscated using decoys under the architecture including MUXes and a multiplier [17],

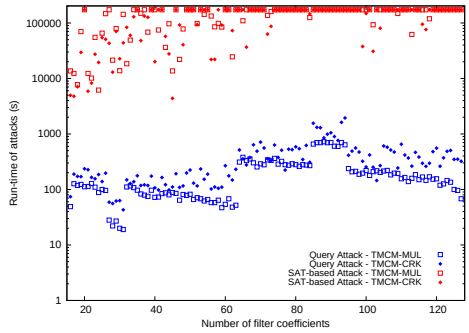


Fig. 5. Run-time of attacks on obfuscated TMCM blocks.

denoted as TMCM-MUL, and also obfuscated by replacing constants with key bits [9], denoted as TMCM-CRK.

Table III presents the number of key bits p and the results of the query attack along with the SAT- and ATPG-based attacks taken from [55]. In this table, *time* denotes the run-time of an attack in seconds and *prv* stands for the number of key bits, whose values are proven by the query attack. Also, *OoT* indicates that an attack could not find a solution due to the time limit, which was set to 2 days. The attacks were run on a computing cluster including Intel Xeon processing units at 2.4 GHz with 40 cores and 96 GB memory. The query attack was developed in Perl and equipped with the ATPG tool Atalanta [56] and the SAT solver CaDiCaL [57]. It is available at *https://github.com/Centre-for-Hardware-Security/*.

Observe from Table III that the query attack can easily find the secret key of obfuscated designs while it is hard for the well-known attacks to find a solution. The main reason is that the TMCM block includes a multiplier block, where one of its inputs is the 32-bit input variable, and the SAT-based attack is not effective on designs including a multiplier as mentioned in [30]. However, the query attack can deal with a small number of queries, which are sufficient to determine the value of each key bit, using a little computational effort.

Second, we generated a total of 112 FIR filters, where n ranges between 16 and 127 when mbw was set to 12, to find the impact of the number of constants and key bits on the performance of the query attack. Again, the folded design of these filters were considered and ibw was set to 32. The TMCM blocks were obfuscated using $2^{\lfloor log_2 n \rfloor + 1}$ key bits under the TMCM-MUL and TMCM-CRK architectures. The SAT-based [30] and query attacks were run on these obfuscated TMCM blocks, where the time limit was set to 2 days. Fig. 5 presents the run-time of these attacks.

Observe from Fig. 5 that as n and p increase, the run-time of the query attack increases slightly. Note that while the query attack can find the secret key of each instance, the SAT-based

Algorithm 3 Selection of decoys for original constants

Inputs: Original constants $C = \{c_1, c_2, \ldots, c_n\}$ and v key bits.
Output: Decoy set D.

1: $noi = 0$ ▷ Number of iterations
2: $nok = 0$ ▷ Number of used key bits
3: $D = \emptyset$ ▷ Set of n decoy constant arrays
4: **while** $nok < v$ **do**
5: $nod = 2^{noi}$ ▷ Number of decoys to be assigned
6: **for** $i = 1$ **to** n **do**
7: D_i = AssignDecoy(D_i, c_i, nod)
8: $nok = nok + 1$
9: **if** $nok == v$ **then**
10: **break**
11: $noi = noi + 1$

attack can find a solution on 39 and 43 instances under the TMCM-MUL and TMCM-CRK architectures, respectively. Observe that the query attack runs faster than the SAT-based attack on these instances. Note that Section V presents more results of the query attack on different multiplier blocks obfuscated and locked by different techniques.

IV. Proposed Hybrid Protection Technique

This section initially presents the obfuscation technique used to hide filter coefficients behind decoys in the CAVM and MCM blocks of parallel direct and transposed forms of FIR filters (cf. Section IV-A and Section IV-B, respectively). Then, it describes the logic locking method using a point function described at RTL (cf. Section IV-C). Finally, it introduces the hybrid protection technique including both of these methods (cf. Section IV-D).

The original constants can be obfuscated using decoys as described in [17]. The motivation behind such obfuscation is that the use of decoys enables us to control the tradeoff between hardware complexity, output corruption, and filter behavior [17], [18] when compared to logic locking. The obfuscation technique using decoys requires two main steps: i) given the number of key bits, determine decoys for each original constant; ii) realize the obfuscated design, where original constants are hidden behind decoys using MUXes and key bits. The selection of decoys for the original constants is done as shown in Algorithm 3. In its *AssignDecoy* function (line 7), decoy selection can be done based on a given criterion, namely hardware complexity, output corruption, and filter behavior. In these criteria, decoys are chosen to be unique to increase the obfuscation.

A. Hardware Obfuscation of the CAVM Block

Given $1 \times n$ original constant array $C = [c_1, c_2, \ldots, c_n]$ and the number of key bits for obfuscation, i.e., v, let D denote a set of n decoy constant arrays, i.e., $D = \{[d_1^1, \ldots, d_1^{nd_1}], [d_2^1, \ldots, d_2^{nd_2}], \ldots, [d_n^1, \ldots, d_n^{nd_n}]\}$, where nd_i is the number of decoy constants selected for the i^{th} original constant determined based on a given criterion with $1 \leq i \leq n$. Then, the set R, which includes each original constant and its decoys, i.e., $R_i = c_i \cup D_i = [c_i, d_i^1, \ldots, d_i^{nd_i}]$ with $1 \leq i \leq n$, is formed. Let $r_{i,j}$ denote the j^{th} constant in R_i with $1 \leq i \leq n$ and $1 \leq j \leq nd_i + 1$. Thus, the

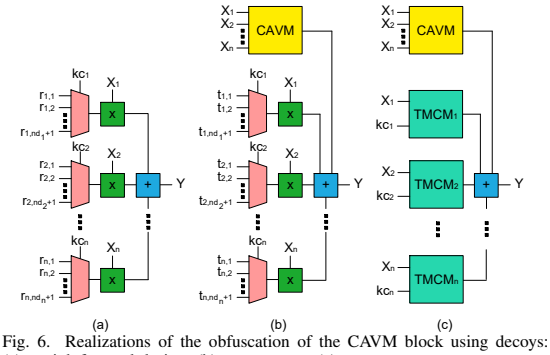


Fig. 6. Realizations of the obfuscation of the CAVM block using decoys: (a) straightforward design; (b) CAVM-MUL; (c) CAVM-SA.

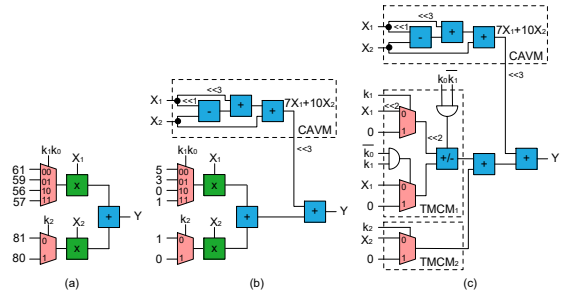


Fig. 7. Realizations of the obfuscation of the CAVM block including constants 57 and 81: (a) straightforward design; (b) CAVM-MUL; (c) CAVM-SA.

straightforward realization of the obfuscated CAVM block is given in Fig. 6(a). Note that the key bits determined for each constant, i.e., kc_i, have the size of $\lceil log_2(nd_i + 1) \rceil$ with $1 \leq i \leq n$. The secret key, which is formed as the concatenation of these key bits, is determined based on the location of the original constant in the constant array R_i.

Note that the size of a multiplier given in Fig. 6(a) is related to the bit-width of the original constant and its decoy(s). Hence, to reduce the hardware complexity of the straightforward design, the size of constants, which are inputs of MUXes, can be decreased. To do so, we implement a CAVM block, where each entry of its constant array S is an element of each R array, i.e., $S = [s_1, s_2, \ldots, s_n]$ with $s_i \in R_i = [c_i, d_i^1, \ldots, d_i^{nd_i}]$ and $1 \leq i \leq n$. Then, the original constant and its decoys at inputs of each MUX are computed as $T_i = R_i - s_i$ with $1 \leq i \leq n$. Fig. 6(b) presents the obfuscated design under the proposed architecture called CAVM-MUL. Note that the CAVM block realizes $s_1X_1 + s_2X_2 + \ldots + s_nX_n$ and is implemented under the shift-adds architecture using the algorithm of [20]. The constants to be in S are decided based on the hardware complexity of the CAVM block and the size of multipliers. This problem is formulated as a 0-1 Integer Linear Programming (ILP) problem.

To further reduce the hardware complexity of the design in Fig. 6(b), each multiplier with a MUX, which represents a TMCM block, is realized under the shift-adds architecture using the algorithm of [22]. Fig. 6(c) presents the obfuscated design under the proposed architecture called CAVM-SA.

Returning to our example in Fig. 2 with $C = [57, 81]$ and assuming that the number of key bits is 3, the set D, that includes decoys for each constant, is found as $D = \{[61, 59, 56], [80]\}$

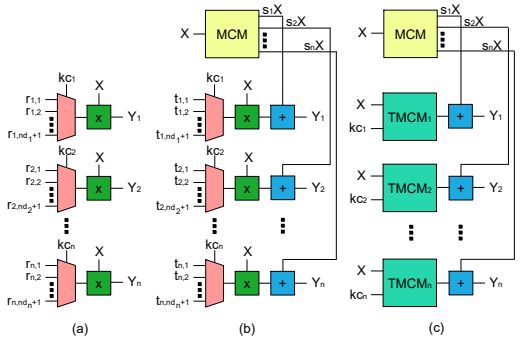


Fig. 8. Realizations of the obfuscation of the MCM block using decoys: (a) straightforward design; (b) MCM-MUL; (c) MCM-SA.

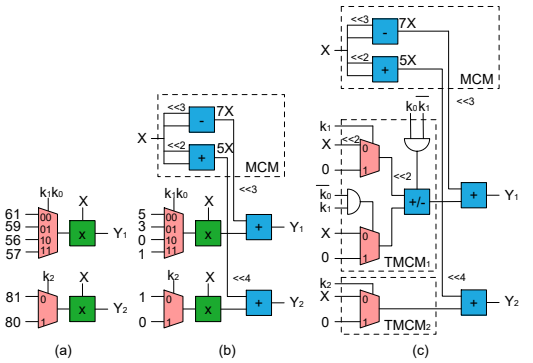


Fig. 9. Realizations of the obfuscation of the MCM block including constants 57 and 81: (a) straightforward design; (b) MCM-MUL; (c) MCM-SA.

based on the hardware complexity criterion. Thus, the set R is formed as $R = \{[61, 59, 56, 57], [81, 80]\}$. The straightforward realization of the obfuscated CAVM block using decoys is shown in Fig. 7(a), where the secret key is $\boldsymbol{K} = k_2k_1k_0 = 011$. Under the CAVM-MUL and CAVM-SA architectures, the constant array S is determined as $S = [56, 80]$. Thus, the set T is formed as $\{[5, 3, 0, 1], [1, 0]\}$ leading to multipliers with smaller sizes when compared to those given in Fig. 7(a). The realization of the obfuscated CAVM design under the CAVM-MUL architecture is given in Fig. 7(b). Furthermore, Fig. 7(c) presents the shift-adds realization of the TMCM blocks implementing the constant multiplications including those in the set T under the CAVM-SA architecture.

In addition to the obfuscation using decoys on the CAVM block, we also developed the constant obfuscation technique used in the ASSURE tool [9]. Given the number of key bits, constants in the original CAVM block are replaced by key bits under the architecture called CAVM-CRK.

B. Hardware Obfuscation of the MCM Block

Similarly, the MCM block can also be obfuscated using decoys. After decoys selected for each original constant are found based on the given criterion, and the set R is determined, the straightforward realization of the obfuscated design can be obtained as illustrated in Fig. 8(a). Moreover, the size of multipliers can be reduced by determining the set of constants

x_2 x_1 x_0	K^7 K^6 K^5 K^4	K^3	K^2 K^1 K^0
0 0 0	0 0 0 0	0	0 0 1
0 0 1	0 0 0 0	0	0 1 0
0 1 0	0 0 0 0	0	1 0 0
0 1 1	0 0 0 0	0	0 0 0
1 0 0	0 0 0 1	0	0 0 0
1 0 1	0 0 1 0	0	0 0 0
1 1 0	0 1 0 0	0	0 0 0
1 1 1	1 0 0 0	0	0 0 0

(a)

x_2 x_1 x_0	K^7 K^6 K^5 K^4	K^3	K^2 K^1 K^0
0 0 0	0 0 0 0	0	0 0 1
0 0 1	0 0 0 0	0	0 1 1
0 1 0	0 0 0 0	0	1 1 0
0 1 1	0 0 0 0	0	1 0 0
1 0 0	0 0 0 1	0	0 0 0
1 0 1	0 0 1 1	0	0 0 0
1 1 0	0 1 1 0	0	0 0 0
1 1 1	1 1 0 0	0	0 0 0

(b)

Fig. 10. (a) Behavior of a Boolean function locked by one-point function; (b) behavior of a Boolean function locked by relaxed one-point function.

S from the set R and the set T is computed accordingly as described in Section IV-A. The multiplications of constants in the set S by the variable X, i.e., $s_1X, s_2X, \ldots, s_nX$, are realized in an MCM block, which is implemented under the shift-adds architecture using the algorithm of [21]. Fig. 8(b) presents the obfuscated design under the proposed architecture called MCM-MUL. Furthermore, the multiplierless realization of the design obfuscated under the MCM-MUL architecture can be obtained by realizing the TMCM block under the shift-adds architecture using the algorithm of [22]. Fig. 8(c) shows the obfuscated design under the proposed architecture called MCM-SA.

Returning to our example, Fig. 9 presents the straightforward realization of the obfuscated MCM block and its designs under the MCM-MUL and MCM-SA architectures.

In addition to the obfuscation using decoys, constants in the original MCM block are replaced by key bits under the architecture called MCM-CRK.

C. Logic Locking with a Point Function

As shown in Sections III and V, the constant obfuscation techniques are vulnerable to the SAT-based attack and its variants. The motivation behind locking the obfuscated design with a point function is to make it resilient to these techniques. In order to increase the number of DIPs to be explored in a SAT-based attack, one can lock primary outputs of a multiplier block using a point function[1] at RTL as done at gate-level in [23], [31]–[33].

Suppose that a Boolean function $f : \mathbb{B}^q \rightarrow \mathbb{B}$ is locked using a one-point function with w key bits, where $w \leq q$, leading to a locked Boolean function $g : \mathbb{B}^{w+q} \rightarrow \mathbb{B}$ and let $\boldsymbol{K}$ denotes the secret key. Then, $f(X) = g(X, \boldsymbol{K})$ under all possible input values. Fig. 10(a) shows the behavior of the locked function g under each possible key value when $q = w = 3$ and $k_2k_1k_0 = 011$ is the secret key. In this figure, K^i stands for the assignment of the value i in binary to key bits, i.e., $k_{w-1} \ldots k_1k_0 = (i)_{bin}$ with $0 \leq i \leq 2^w - 1$. Also, the value of logic 0 (1) under each possible key value denotes that the locked function g is (not) equal to the original function f. Note that the locked function under the secret key, i.e., $\boldsymbol{K} = K^3$ highlighted in our example, always generates the same output as the original function for every input pattern. Observe from Fig. 10(a) that each input pattern eliminates at most one wrong key, leading to an exponential number of

[1]A one-point function is a Boolean function that evaluates to 1 at exactly one input pattern.

```
// One-point function at RTL
always @(*) begin
  if (X == K)
    if (K == K)
      g = f;
    else
      g = !f;
  else
    g = f;
end
//Relaxed one-point function at RTL
always @(*) begin
  if ((X - K) >= q'd0 && (X - K) <= q'dcv)
    if (K == K)
      g = f;
    else
      g = !f;
  else
    g = f;
end
```

Listing 1. Logic locking using a point function at RTL.

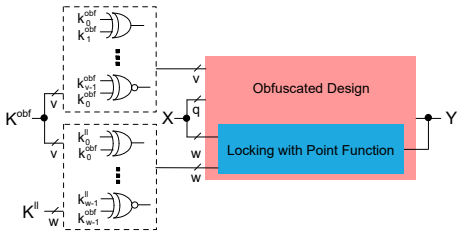


Fig. 11. Proposed hybrid protection technique.

DIPs to find the secret key, i.e., 2^w. Moreover, such a one-point function can be relaxed to increase the corruption at a primary output. For example, Fig. 10(b) presents the behavior of the locked function g, where each input pattern can eliminate at most 2 wrong keys. Observe that the exponential number of tries to find the secret key is still valid, i.e., 2^{w-1} in this case. Furthermore, multiple primary outputs can be locked using point functions with different key bits.

Listing 1 presents the Verilog code snippet, which describes logic locking using the one-point function at RTL. In this code, X is an array of primary inputs and K is an equally sized array of key bits. Moreover, the relaxed one-point function can also be described at RTL, as shown in the same listing. In its code, cv stands for the corruption value, which denotes the maximum number of wrong keys that can be distinguished by a single input pattern.

We note that since the point function is described at RTL, the synthesis tool shapes its circuit based on the given synthesis parameters. Thus, its realization does not have a regular structure like logic locking methods of [32], [33], [35].

D. Combination of Obfuscation and Logic Locking

The hybrid protection technique includes obfuscation and logic locking using a point function described in previous subsections. Initially, the obfuscation technique using decoys is applied and then, the obfuscated design is locked using a point function. Fig. 11 illustrates the hybrid protection technique, where X and Y denote the inputs and outputs of the original design and K^{obf} and K^{ll} stand for the key bits used for obfuscation and locking, respectively. Additionally, to thwart the structural attacks, the key bits used for logic locking, K^{ll}, are hidden among the key bits used for obfuscation, K^{obf}, using XOR/XNOR gates. In this scheme, an XOR (XNOR) gate, which has k_i^{ll} and k_j^{obf} as inputs, is generated if the value of k_j^{obf} in the secret key is equal to logic 0 (1) value, where $0 \leq i \leq w-1$ and $0 \leq j \leq v-1$. Then, the output of this gate is connected to the net, which would be driven by k_i^{ll}. Moreover, to thwart the query attack, each k_i^{obf} is hidden among another k_j^{obf} using an XOR/XNOR gate, where $i \neq j$ and $0 \leq i, j \leq v-1$. By doing so, each key bit is observed with other key bits at a primary output, making it harder for the query attack to prove the value of the related key bit.

TABLE IV
DETAILS ON THE FIR FILTER AND ITS MULTIPLIER BLOCKS.

Filter	n	mbw	Multiplier Block	$\#in$	$\#out$
Johansson08	30	10	CAVM	480	31
			MCM	16	733
			TMCM	21	26

We developed a Computer-Aided Design (CAD) tool to automate the design and verification process for the obfuscation and locking of the CAVM, MCM, and TMCM blocks and the parallel direct and transposed form and folded FIR filters. The CAD tool takes the filter coefficients, the number of key bits, the design architecture, and other design parameters as inputs and generates the description of the obfuscated design in Verilog, the testbench for verification, and synthesis and simulation scripts. Note that designs are described in a behavioral fashion at RTL.

V. Experimental Results

In this section, we introduce the gate-level synthesis results of multiplier blocks protected by the proposed hybrid technique, obfuscated by previously proposed techniques, and locked by prominent logic locking methods. We also provide the results of well-known logic locking attacks and the proposed query attack on these designs. Furthermore, we present the results of obfuscated and locked FIR filters and also, introduce the results of prominent attacks on these designs. Finally, we explore the impact of parameters used in the point function on the hardware complexity and resiliency to the SAT-based attack and present gate-level synthesis results of the obfuscated and locked CAVM block of the direct form filter, which has promising security properties.

A. Results of Obfuscated and Locked Multiplier Blocks

Based on our experimental observations, similar outcomes have been observed on FIR filters with a different number of coefficients and different bit-width of filter input and coefficients in comparison of design architectures, obfuscation and locking techniques, and attacks. Hence, in this experiment, a single FIR filter with a small number of coefficients and a small bit-width of filter input and coefficients was used to reveal the effectiveness of obfuscation and locking techniques. Table IV presents the details of this filter taken from [54], where *#in* and *#out* are respectively the number of inputs and outputs of the multiplier blocks when ibw is 16.

TABLE V
RESULTS OF OBFUSCATED AND PROTECTED MULTIPLIER BLOCKS.

Block	Architecture	Technique	Synthesis			Attacks							
						SAT	ATPG	AppSAT	DoubleDIP	Query		SCOPE	
			area	delay	power	time	time	time	time	prv	time	cdk/dk	time
CAVM	CAVM-MUL	Decoy [17]	15435	4616	4641	155143	OoT	OoT	OoT	32	9893	20/32	8
		Proposed Hybrid	15710	4611	4757	OoT	OoT	OoT	OoT	0	OoT	1/1	13
	CAVM-SA	Decoy [17]	15465	4611	4475	36083	4539	OoT	OoT	32	9944	20/32	8
		Proposed Hybrid	15704	4715	4497	OoT	OoT	OoT	OoT	0	29328	1/1	12
	CAVM-CRK	Constant [9]	18737	3982	4756	110	1446	OoT	OoT	32	897	21/27	11
		Proposed Hybrid	18976	4265	4809	OoT	OoT	OoT	OoT	0	1937	2/3	16
MCM	MCM-MUL	Decoy [17]	10949	3102	2839	106	OoT	324	243	32	176	27/32	7
		Proposed Hybrid	11173	3031	2897	OoT	OoT	OoT	OoT	0	197	1/1	11
	MCM-SA	Decoy [17]	10493	3112	2493	119	OoT	342	254	32	152	27/32	7
		Proposed Hybrid	10705	3159	2495	OoT	OoT	OoT	OoT	0	115	1/1	11
	MCM-CRK	Constant [9]	12799	2772	2412	159	OoT	415	300	32	459	18/32	7
		Proposed Hybrid	13038	2970	2456	OoT	OoT	OoT	OoT	0	759	1/1	11
TMCM	TMCM-MUL	Decoy [17]	1545	3517	610	241	6783	378	496	32	7	21/32	4
		Proposed Hybrid	1794	4278	639	OoT	OoT	OoT	OoT	0	17	2/3	2
	TMCM-SA	Decoy [17]	2043	4452	1037	738	963	935	OoT	32	37	17/32	4
		Proposed Hybrid	2245	4536	1065	OoT	OoT	OoT	OoT	0	42	1/2	3
	TMCM-CRK	Constant [9]	1566	2997	623	1035	15571	1032	OoT	32	29	20/32	4
		Proposed Hybrid	1776	3655	642	OoT	OoT	OoT	OoT	0	48	1/1	2

TABLE VI
RESULTS OF LOCKED MULTIPLIER BLOCKS.

Block	Logic Locking	Synthesis			Attacks							
					SAT	ATPG	AppSAT	DoubleDIP	Query		SCOPE	
		area	delay	power	time	time	time	time	prv	time	cdk/dk	time
CAVM	RLL	16411	3385	4183	171	OoT	205	2609	32	254	0/0	9
	RLL+AntiSAT	16492	3416	4127	OoT	OoT	OoT	OoT	16	364	0/0	14
	RLL+CASLock	16473	3502	4110	OoT	OoT	OoT	OoT	16	443	0/0	13
	RLL+SARLock	16512	3572	4191	OoT	OoT	48855	OoT	32	441	8/8	13
	RLL+SFLL	16506	3473	4205	339	829	39664	OoT	32	436	0/0	14
	RLL+SKGLock	16559	3894	4308	OoT	OoT	OoT	OoT	19	513	5/6	14
MCM	RLL	8090	2398	2039	86	122	293	371	32	89	0/0	6
	RLL+AntiSAT	8244	2434	2065	OoT	OoT	OoT	OoT	16	249	0/0	9
	RLL+CASLock	8121	2404	2024	OoT	OoT	1054	OoT	16	164	0/0	9
	RLL+SARLock	8209	2467	2041	OoT	OoT	45013	OoT	32	228	10/10	9
	RLL+SFLL	8166	2452	2019	576	7870	2626	OoT	32	243	0/0	9
	RLL+SKGLock	8252	2464	2056	OoT	OoT	OoT	OoT	19	160	5/5	9
TMCM	RLL	1587	3712	646	8	39	57	77	32	15	0/0	2
	RLL+AntiSAT	1659	3545	632	OoT	OoT	OoT	OoT	13	28	0/0	2
	RLL+CASLock	1653	3664	628	OoT	OoT	OoT	OoT	15	20	0/0	2
	RLL+SARLock	1659	3756	658	OoT	OoT	2662	OoT	31	36	6/6	3
	RLL+SFLL	1644	3800	653	1095	1089	5363	OoT	32	36	0/0	2
	RLL+SKGLock	1694	3739	676	OoT	OoT	OoT	OoT	18	34	10/10	2

Table V presents the synthesis results of the CAVM, MCM, and TMCM blocks of the FIR filter obfuscated by previously proposed methods [9], [17] and protected by the proposed hybrid technique. Note that the TMCM-SA architecture denotes the TMCM block obfuscated using decoys under the shift-adds architecture. Logic synthesis was performed by Cadence Genus using a commercial 65 nm cell library with the aim of area optimization. For this aim, a very high virtual clock period value, i.e., 80 ns, was used. The encrypted designs were validated by simulation using 10,000 randomly generated inputs, where the switching activity data of each node in the design were collected and stored in a Switching Activity Interchange Format (SAIF) file, which is later used by the synthesis tool while computing the power dissipation. For obfuscation, 32 key bits were used. There were 16 key bits for locking using the one-point function. Thus, a total of 48 key bits were used in designs protected by the hybrid technique. In this table, *area*, *delay*, and *power* stand for the total area in μm^2, delay in the critical path in ps, and total power dissipation in μW, respectively. This table also presents the results of OG attacks, namely SAT- and ATPG-based attacks, the approximate AppSAT attack taken from [55], and the DoubleDIP attack taken from [58], and the OL SCOPE attack taken from [59]. For the SCOPE attack, *cdk* and *dk* denote the number of correctly deciphered key bits and the number of deciphered key bits, respectively. The time limit given to the attacks was 2 days. In this table, designs, whose secret key has not been discovered by the given attacks, are highlighted.

1) Comments on Hardware Complexity: Observe from Table V that the hybrid protection technique increases the hardware complexity when compared to the obfuscation techniques simply due to the inclusion of the point function and logic for the obfuscation of key bits. Note that the increase of area in the CAVM, MCM, and TMCM blocks reaches up to 1.7%, 2%, and 13.8%, respectively. The obfuscation and hybrid protection of the CAVM and MCM blocks under the proposed architectures, i.e., CAVM-MUL, CAVM-SA, MCM-MUL, and MCM-SA, lead to designs with less area when compared to those realized under the CAVM-CRK and MCM-CRK architectures. Note that such a decrease reaches up to 17.6% and 18% in the CAVM and MCM blocks, respectively. This is simply because the proposed techniques exploit common

TABLE VII
RESULTS OF OBFUSCATED AND LOCKED FIR FILTERS.

Filter	Obfuscation and Hybrid Protection								Logic Locking							
	Architecture	Technique	Synthesis			Attacks			Technique	Synthesis			Attacks			
						KC2	SCOPE						KC2	SCOPE		
			area	delay	power	time	cdk/dk	time		area	delay	power	time	cdk/dk	time	
Direct	CAVM-MUL	Decoy [17]	19238	4907	3088	Failed	18/32	8	RLL	20233	3566	2906	Failed	15/18	10	
		Proposed Hybrid	19500	4828	3153	OoT	1/1	13	RLL+AntiSAT	20300	3512	2896	Failed	21/26	14	
	CAVM-SA	Decoy [17]	19243	4792	3012	Failed	19/32	8	RLL+CASLock	20324	3746	2928	OoT	11/15	14	
		Proposed Hybrid	19485	4798	3040	OoT	1/1	13	RLL+SARLock	20326	3630	2936	OoT	31/37	14	
	CAVM-CRK	Constant [9]	22551	4228	2734	Failed	14/32	10	RLL+SFLL	20307	3619	2892	Failed	21/28	14	
		Proposed Hybrid	22796	4244	2757	OoT	2/4	15	RLL+SKGLock	20380	4052	2994	Failed	16/24	15	
Trans.	MCM-MUL	Decoy [17]	25195	3470	3848	100347	25/32	11	RLL	22362	3093	3303	67811	16/21	9	
		Proposed Hybrid	25439	3540	3896	OoT	1/1	16	RLL+AntiSAT	22510	3218	3302	OoT	15/24	15	
	MCM-SA	Decoy [17]	24967	3322	3569	82952	26/32	10	RLL+CASLock	22461	3337	3320	OoT	7/8	14	
		Proposed Hybrid	25139	3346	3562	OoT	1/1	15	RLL+SARLock	22425	3183	3303	OoT	31/41	14	
	MCM-CRK	Constant [9]	27126	3240	3273	51973	21/32	11	RLL+SFLL	22389	3116	3311	OoT	19/29	14	
		Proposed Hybrid	27433	3256	3290	OoT	1/1	17	RLL+SKGLock	22514	3186	3329	OoT	17/24	15	
Folded	TMCM-MUL	Decoy [17]	9126	4785	869	7478	20/32	2	RLL	9183	4496	904	7845	15/18	3	
		Proposed Hybrid	9379	4665	933	OoT	2/2	3	RLL+AntiSAT	9225	4681	882	OoT	8/15	4	
	TMCM-SA	Decoy [17]	9791	5758	1168	11895	18/32	2	RLL+CASLock	9237	4675	926	OoT	8/13	4	
		Proposed Hybrid	9966	5646	1236	OoT	9/13	3	RLL+SARLock	9235	4761	911	OoT	31/31	4	
	TMCM-CRK	Constant [9]	9126	4328	870	5657	22/32	2	RLL+SFLL	9222	4570	892	OoT	16/20	4	
		Proposed Hybrid	9356	4602	894	OoT	1/1	3	RLL+SKGLock	9288	4434	910	OoT	18/31	4	

subexpressions shared in constant multiplications. On the other hand, the obfuscation and hybrid protection of the TMCM blocks under the TMCM-MUL and TMCM-CRK architectures lead to designs with less area with respect to those realized under the TMCM-SA architecture. Note that such a decrease reaches up to 24.3%. This is because a single multiplier is replaced by a large number of addition and subtraction operations under the TMCM-SA architecture. It is also observed that the minimum achievable delay values in the critical path of obfuscated and protected multiplier blocks are very close to each other, meaning that the inclusion of the point function and logic for the obfuscation of key bits does not have a significant impact while realizing the design with the smallest delay.

2) Comments on Attack Resiliency: Observe also from Table V that while the OG attacks can easily discover the secret key on the obfuscated designs, the OL attack can decipher all the key bits with high accuracy, except for the CAVM design obfuscated under the CAVM-CRK architecture. On the other hand, none of these attacks can break the defense built by the hybrid protection technique. Note that the designs protected by the hybrid technique were also applied to Fa-SAT [51] and the Valkyrie tool [52], but without any success due to the combination of both obfuscation and locking.

Moreover, these multiplier blocks under an architecture including a multiplier are locked by prominent logic locking methods, namely, RLL [27] and the SAT-resilient methods of AntiSAT [32], SARLock [31], SFLL [23], CASLock [33], and SKGLock [35]. In this case, the multiplier block described at RTL is initially synthesized and its gate-level netlist is obtained, and then, this netlist is locked. Note that while the RLL, AntiSAT, and SFLL methods were applied using the NEOS tool [60], the script for the SARLock method was provided by P. Subramanyan, and we implemented the CASLock and SKGLock methods. In the RLL method, 32 key bits are used, the same as the obfuscation techniques presented in Table V. Same as the hybrid protection method shown in Table V, there are a total of 48 key bits in the combination of RLL and a SAT-resilient method, while 16 key bits are designated to a SAT-resilient method. Note that due to the locking nature of AntiSAT, CASLock, and SKGLock, they require twice the number of designated key bits. Hence, a total of 32 key bits are used in these methods. Table VI presents the results of locked multiplier blocks. The locked designs, whose secret key has not been discovered by the given attacks, are also highlighted.

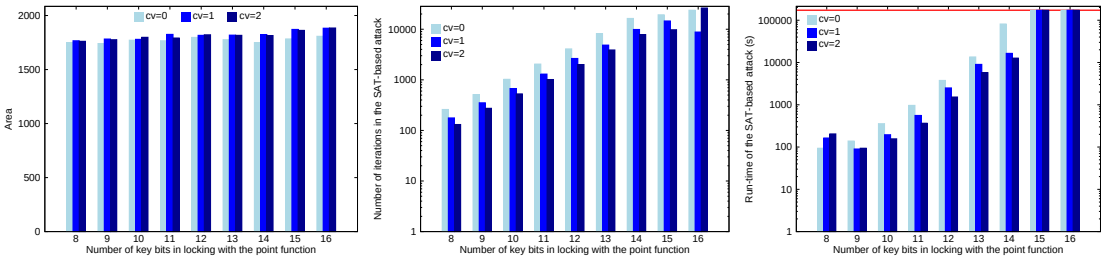
3) Comments on Hardware Complexity: Observe from Table VI that SAT-resilient methods with a combination of RLL lead to designs with hardware complexity very close to each other. When compared to the results of the hybrid protection technique given in Table V under the architectures using multiplier(s), the locked CAVM and MCM blocks have larger and smaller area, respectively and the locked TMCM blocks have competitive area. A locked MCM block has less hardware complexity than an obfuscated or protected MCM block because the logic locking is applied after the common subexpressions are exploited by the synthesis tool.

4) Comments on Attack Resiliency: Also, observe from Table VI that the secret key of designs locked by RLL, RLL+SARLock, and RLL+SFLL[2] can be found by the given attacks. Note also that the MCM block locked by RLL+CASlock could be broken by the AppSAT. While the query attack is also capable of proving the values of most of the RLL key bits in all logic locking methods and extra SKGLock key bits, the SCOPE attack can predict the values of some key bits of designs locked by RLL+SARLock and RLL+SKGLock with high accuracy.

B. Results of the Obfuscated and Locked FIR Filters

Table VII presents the synthesis results of parallel direct and transposed form and folded FIR filters, whose CAVM, MCM, and TMCM blocks are obfuscated by the previously proposed techniques [9], [17] and the hybrid protection technique,

[2]Confirmed by the developer of the SFLL method that when a small number of key bits are used and their values are biased towards all logic 0s or 1s in the SFLL method, the exponential growth in the number of iterations in the SAT-based attack is no longer valid.



(a) (b) (c)

Fig. 12. Impact of point function parameters: (a) area; (b) number of iterations; (c) run-time.

respectively. It also shows the synthesis results of FIR filters locked by prominent logic locking methods. It introduces the results of attacks that can be applied to sequential circuits namely, the OG KC2 attack, which was taken from [60], and the OL SCOPE attack. In this table, *failed* denotes that the found solution of the KC2 attack is actually a wrong key verified through simulation.

1) Comments on Hardware Complexity: Observe from Table VII that the direct form filter has less area and consumes less power, but has a higher delay when compared to the transposed form filter. On the other hand, the folded design has the smallest area, but the filter output is computed in 30 clock cycles, increasing the latency and energy consumption. The conclusions drawn based on the gate-level synthesis results on the obfuscated and locked multipliers blocks given in Tables V-VI are also valid on the obfuscated and locked FIR filters. However, due to the registers in the filter design, the overhead on the overall FIR filter design gets smaller. Note that the proposed hybrid technique achieves the maximum area reduction with respect to the logic locking methods on the parallel direct form FIR filters, i.e., 4.4%, obtained when the FIR filter under the CAVM-SA architecture is compared to the FIR filter locked by RLL+SKGLock.

2) Comments on Attack Resiliency: Observe also from Table VII that the KC2 attack is capable of discovering the secret key of the obfuscated FIR filters using previously proposed techniques, except for the direct form filters, but it is not successful on the FIR filters protected by the proposed hybrid technique. It can also find the secret key locked by RLL, except for the direct form filter, but fails on the filters locked by both RLL and a SAT-resilient method. Similarly, the SCOPE attack generally deciphers all the key bits on the obfuscated FIR filters with high accuracy, but it can only decipher a small number of key bits of the FIR filters protected by the hybrid technique. However, it is capable of deciphering more key bits on the locked FIR filters when compared to its results on the locked multiplier blocks. This is heavily due to the resynthesis of the FIR filter including the locked multiplier block. Note that the proposed hybrid technique increases the area and power dissipation of FIR filters when compared to the previously proposed obfuscation techniques [9], [17] in order to increase their resiliency to the existing attacks.

C. Analysis on the Point Function

To find the impact of the point function and its parameters in the hybrid protection technique on the hardware complexity, the number of iterations taken in the SAT-based attack [30], and its run-time, we used the TMCM block of our FIR filter under the TMCM-MUL architecture. Again, the TMCM block is obfuscated using decoys with 32 key bits when ibw is 16. In logic locking with the point function, the number of key bits, i.e., w, is determined to be between 8 and 16, the corruption value, i.e., cv, is set to be between 0 and 2, and a single primary output is locked. Fig. 12 presents the impact of point function parameters on the area of the protected TMCM block and the number of iterations and run-time of the SAT-based attack.

Observe from Fig. 12(a) that as the number of key bits used in logic locking, w, increases, the area of the protected TMCM block using the hybrid technique increases slightly. Note that as the corruption value, cv, increases, the design area increases simply due to the increased range of comparator logic given in Listing 1. Also, observe from Figs. 12(b)-(c) that as w increases, the number of iterations and run-time of the SAT-based attack increases. An exponential growth in the number of iterations and run-time can be observed till w is 15. As can be seen from Fig. 12(c), for the 15- and 16-bit keys in the point function, the SAT-based attack cannot find the secret key in the time limit, i.e., 2 days, denoted by the red line. Thus, the number of iterations given in Fig. 12(b) for these number of key bits, is the one obtained in the time limit. Note also that in all TMCM designs locked by the point function with the given parameters, the number of iterations increases exponentially, while it decreases as cv is increased, but still keeping the exponential growth.

To find the impact of locking an obfuscated design using a point function on hardware complexity and attack resiliency, we used the same 112 FIR filters presented in Section III-B, where n ranges between 16 and 127. In our experiments, the TMCM blocks of folded FIR filters were obfuscated using $2^{\lfloor log_2 n \rfloor + 1}$ key bits under the TMCM-MUL architecture when ibw was set to 16. For the point function, 16 key bits were used. Fig. 13 presents the run-time of the SAT-based attack on obfuscated and protected TMCM blocks when its time limit was 2 days.

Observe from Fig. 13 that locking an obfuscated design using a point function increases the SAT-based attack resiliency

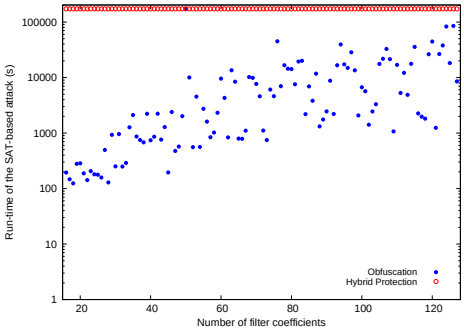


Fig. 13. Run-time of the SAT-based attack on TMCM blocks.

TABLE VIII
DETAILS ON THE PROTECTED AND LOCKED CAVM BLOCKS.

Filter	n	mbw	p	Hybrid			RLL+CASLock		
				area	delay	power	area	delay	power
Dempster02	25	12	41	11145	5903	7057	16557	3136	4238
Johansson08	30	10	46	14826	4662	4504	16461	3381	4101
Shi11_Y2	34	11	50	12270	3594	3567	12533	3163	3190
Shi11_A	59	10	75	21431	4858	5825	20993	3404	5390
Samueli89	60	13	76	24188	4130	7522	26696	3998	6819
Lim83	63	10	79	22932	4848	6652	24696	3661	6322
Yoshino09	64	13	80	25096	4964	7927	30414	3437	7927
Nielsen89	67	15	83	26187	5318	9645	34950	4071	9164
Maskell07	108	9	124	37306	4811	9049	37148	3604	9522
LimYu07	121	14	137	48354	4947	16862	59492	3888	15346

significantly. Note that the SAT-based attack can find a solution to all obfuscated TMCM blocks except one. However, the average area, delay, and power dissipation of the protected designs are increased by 10.7%, 7.1%, and 9.6%, respectively when compared to those of the obfuscated designs.

D. Analysis on the Direct Form FIR Filter

Among the parallel design of FIR filters, the direct form is a good candidate to be used in a secure implementation for several reasons based on the results obtained in this work. First, as shown in Table VII, its obfuscated hardware complexity in terms of area and power dissipation is significantly smaller than the transposed form filter. Second, it includes a large number of multiplication operations in chain, which make the SAT-based attack and its variants hard to discover the secret key. Third, the CAVM block of the direct form filter has a large number of inputs than the MCM block of the transposed form filter, which enables an increase in the number of key bits in the point function, improving the resiliency of the design protected by the hybrid technique as shown in Figs. 12(b)-(c). This last observation is also true when compared to the TMCM block used in folded FIR filter design.

To find the impact of the number of coefficients on the hardware complexity of the protected CAVM block of an FIR filter, 10 filters were taken from [54], where n ranges between 25 and 121 and mbw is between 9 and 15. In the design of CAVM blocks, ibw is set to 16. In the hybrid protection technique, these CAVM blocks were obfuscated using decoys with n key bits and locked using the point function with ibw key bits under the CAVM-MUL architecture using a total of $n + ibw$ key bits. These protected designs are also compared with those locked by both RLL and CASLock, where the number of RLL and CASLock key bits is $n - ibw$ and $2 * ibw$, respectively. This logic locking method was chosen because it generally generates a locked multiplier block with a small area as shown in Table VI. Table VIII presents the gate-level synthesis results of the CAVM designs protected by the hybrid technique and locked by both RLL and CASLock.

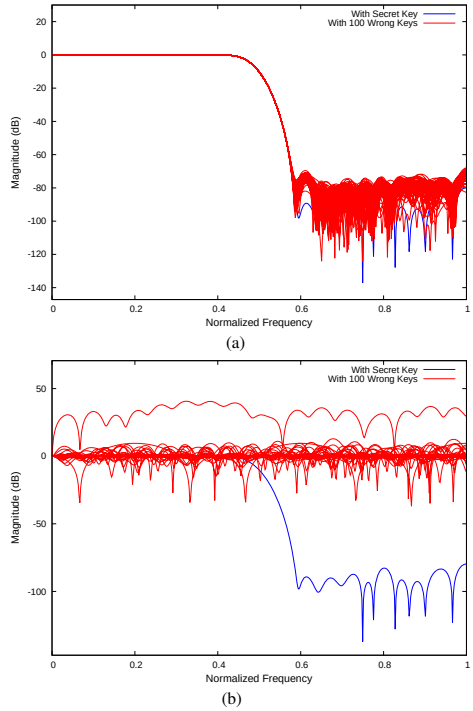


Fig. 14. Behavior of the FIR filter *Nielsen89*: (a) protected by the hybrid technique; (b) locked by RLL+CASLock.

Observe from Table VIII that as the number of coefficients, n, increases, the hardware complexity of the protected and locked CAVM blocks generally increases. The hybrid protection technique generally leads to a design with a smaller area when compared to the RLL+CASLock method, where the gain reaches up to 32.6%. Note that on filters *Shi11_A* and *Maskell07*, the RLL+CASLock method leads to a locked design with a smaller area, since the bit-width of coefficients is small, enabling the synthesis tool to optimize the logic.

To find the impact of obfuscation techniques on the filter behavior, the Zero-Phase Frequency Response (ZPFR) of the FIR filter *Nielsen89* is obtained when the secret key and 100 randomly generated wrong keys are applied. Fig. 14 presents ZPFRs of FIR filters protected by the hybrid technique and locked by RLL+CASlock.

Observe from Fig. 14 that both obfuscation techniques may lead to a filter behavior different from the original one when a random wrong key is applied. While the filter behavior of the protected design under a wrong key is meaningful, but out of desired filter specification, the locked design exhibits an unmeaningful behavior under a wrong key due to the logic related to RLL key bits. Thus, the hybrid protection technique

may make the adversary believe that the filter behavior under the wrong key is actually the desired one.

VI. Discussion

Other than the logic locking attacks used in this article, there exist Reverse Engineering (RE) and Side-Channel Analysis (SCA) techniques that can identify the filter coefficients in an obfuscated design. In [18], a machine learning tool that can determine the decoy selection method used in an obfuscated design was developed. The same work also proposed an RE technique that can identify filter coefficients hidden among decoys determined based on a decoy selection method. It was shown that if more than one decoy is used to obfuscate a filter coefficient, where the Hamming distance between each decoy and filter coefficient is 1, then the coefficient can be identified. In other cases, the RE technique was not capable of identifying original filter coefficients.

To the best of our knowledge, there exists no SCA technique proposed specifically to identify the original filter coefficients in an obfuscated filter design. The challenge for such a technique would be to understand how the synthesis tools embed the constants, i.e., filter coefficients and decoys, into the gate-level design using efficient methods, which optimize the hardware complexity of constant multiplications. This procedure would almost entail reverse engineering the algorithms used by the synthesis tools. In this case, it will be hard to reveal the filter coefficients from the power dissipation or delay values obtained from the obfuscated design since those data come from a logic combining both filter coefficients and decoys. Studying SCA and its efficiency to overcome obfuscation methods remains a formidable path for future research.

VII. Conclusions

This article focused on the obfuscation of digital FIR filters. Initially, it showed that the techniques previously proposed for the obfuscation of FIR filters are vulnerable to our SAT-based query attack, which applies several queries and proves that the found key bit value is the actual value of the related key bit in the secret key. Then, to secure an FIR filter design, it proposed the hybrid protection technique, which includes both obfuscation and locking with a point function. The proposed technique is applied to parallel direct and transposed forms of an FIR filter and its folded implementation. Experimental results clearly showed that the hybrid protection technique is competitive to prominent logic locking techniques in terms of hardware complexity and leads to obfuscated designs resilient to well-known attacks. It is also shown that the direct form FIR filter is a good candidate for secure filter implementation.

Acknowledgment

The authors would like to thank Nimisha Limaye and Satwik Patnaik for running our obfuscated designs on their tools and Mohammad Yasin, Leon Li, and Christian Pilato for fruitful discussions. The attacks were carried out in the High Performance Computing Centre of TalTech.

References

[1] Defence Science Board Task Force. (2015, February) On High Performance Microchip Supply Chain. [Online]. Available: https://dsb.cto.mil/reports/2000s/ADA435563.pdf

[2] S. Amir, B. Shakya, D. Forte, M. Tehranipoor, and S. Bhunia, "Comparative Analysis of Hardware Obfuscation for IP Protection," in *GLSVLSI*, 2017, pp. 363–368.

[3] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *DAC*, 1998, pp. 776–781.

[4] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote Activation of ICs for Piracy Prevention and Digital Right Management," in *ICCAD*, 2007, pp. 674–677.

[5] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, 2012.

[6] T. Hoque, R. S. Chakraborty, and S. Bhunia, "Hardware Obfuscation and Logic Locking: A Tutorial Introduction," *IEEE Design & Test*, vol. 37, no. 3, pp. 59–77, 2020.

[7] L. Wanhammar, *DSP Integrated Circuits*. Academic Press, 1999.

[8] M. R. Muttaki, R. Mohammadivojdan, M. Tehranipoor, and F. Farahmandi, "HLock: Locking IPs at the High-Level Language," in *DAC*, 2021, pp. 79–84.

[9] C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg, and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, vol. 29, no. 7, pp. 1306–1318, 2021.

[10] S. A. Islam, L. K. Sah, and S. Katkoori, "High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging," *ACM TODAES*, vol. 26, no. 1, 2020.

[11] C. Pilato, F. Regazzoni, R. Karri, and S. Garg, "TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis," in *DAC*, 2018, pp. 1–6.

[12] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 467–476, 2017.

[13] R. S. Chakraborty and S. Bhunia, "RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation," in *International Conference on VLSI Design*, 2010, pp. 405–410.

[14] Y. J. Yu and Y. C. Lim, "Optimization of Linear Phase FIR Filters in Dynamically Expanding Subexpression Space," *Circuits, Systems, and Signal Processing*, vol. 29, no. 1, pp. 65–80, 2010.

[15] Y. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE TVLSI*, vol. 23, no. 5, pp. 819–830, 2015.

[16] G. Bottegal, F. Farokhi, and I. Shames, "Preserving Privacy of Finite Impulse Response Systems," *IEEE Control Systems Letters*, vol. 1, no. 1, pp. 128–133, 2017.

[17] L. Aksoy, Q.-L. Nguyen, F. Almeida, J. Raik, M.-L. Flottes, S. Dupuis, and S. Pagliarini, "High-Level Intellectual Property Obfuscation via Decoy Constants," in *IOLTS*, 2021, pp. 1–7.

[18] L. Aksoy, A. Hepp, J. Baehr, and S. Pagliarini, "Hardware Obfuscation of Digital FIR Filters," in *DDECS*, 2022, pp. 68–73.

[19] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.

[20] L. Aksoy, P. Flores, and J. Monteiro, "ECHO: A Novel Method for the Multiplierless Design of Constant Array Vector Multiplication," in *ISCAS*, 2014, pp. 1456–1459.

[21] L. Aksoy, E. O. Güneş, and P. Flores, "Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate," *Elsevier MICPRO*, vol. 34, no. 5, pp. 151–162, 2010.

[22] L. Aksoy, P. Flores, and J. Monteiro, "Multiplierless Design of Folded DSP Blocks," *ACM TODAES*, vol. 20, no. 1, 2014.

[23] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *ACM CCS*, 2017, pp. 1601–1618.

[24] L. Aksoy, P. Flores, and J. Monteiro, "A Tutorial on Multiplierless Design of FIR Filters: Algorithms and Architectures," *Circuits, Systems, and Signal Processing*, vol. 33, no. 6, pp. 1689–1719, 2014.

[25] Y. Voronenko and M. Püschel, "Multiplierless Multiple Constant Multiplication," *ACM Transactions on Algorithms*, vol. 3, no. 2, 2007.

[26] P. Tummeltshammer, J. Hoe, and M. Püschel, "Time-Multiplexed Multiple-Constant Multiplication," *IEEE TCAD*, vol. 26, no. 9, pp. 1551–1563, 2007.

[27] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *DATE*, 2008, pp. 1069–1074.

[28] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," in *ACM CCS*, 2013, pp. 709–720.

[29] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *Journal of Electronic Testing*, vol. 35, pp. 273–291, 2019.

[30] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *HOST*, 2015, pp. 137–143.

[31] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *HOST*, 2016, pp. 236–241.

[32] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE TCAD*, vol. 38, no. 2, pp. 199–207, 2019.

[33] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on CHES*, vol. 2020, no. 1, pp. 175–202, 2019.

[34] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-Based Perspective," *IEEE TCAD*, vol. 39, no. 12, pp. 4439–4452, 2020.

[35] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *ISVLSI*, 2021, pp. 114–119.

[36] H. Zhou, A. Rezaei, and Y. Shen, "Resolving the Trilemma in Logic Encryption," in *ICCAD*, 2019, pp. 1–8.

[37] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," in *Asian HOST*, 2018, pp. 56–61.

[38] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *DATE*, 2019, pp. 540–545.

[39] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE TVLSI*, vol. 29, no. 8, pp. 1529–1542, 2021.

[40] L. Alrahis, S. Patnaik, F. Khalid, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "GNNUnlock: Graph Neural Networks-based Oracle-less Unlocking Scheme for Provably Secure Logic Locking," in *DATE*, 2021, pp. 780–785.

[41] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," in *DATE*, 2019, pp. 936–939.

[42] F. Yang, M. Tang, and O. Sinanoglu, "Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) - Unlocked," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2778–2786, 2019.

[43] Z. Han, M. Yasin, and J. J. Rajendran, "Does Logic Locking Work with EDA Tools?" in *USENIX Security Symposium*, 2021, pp. 1055–1072.

[44] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *DAC*, 2012, pp. 83–89.

[45] Y. Shen and H. Zhou, "Double DIP: Re-Evaluating Security of Logic Encryption Algorithms," in *GLSVLSI*, 2017, pp. 179–184.

[46] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks," *IACR Transactions on CHES*, vol. 2019, no. 1, pp. 97–122, 2018.

[47] C. Karfa, R. Chouksey, C. Pilato, S. Garg, and R. Karri, "Is Register Transfer Level Locking Secure?" in *DATE*, 2020, p. 550?555.

[48] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *HOST*, 2017, pp. 95–100.

[49] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-Condition Crunching for Fast Sequential Circuit Deobfuscation," in *DATE*, 2019, pp. 534–539.

[50] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," in *Cryptographic Hardware and Embedded Systems*, vol. 10529, 2017, pp. 189–210.

[51] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based Attack on Compound Logic Locking Techniques," in *DATE*, 2021, pp. 1166–1171.

[52] ——, "Valkyrie: Vulnerability Assessment Tool and Attack for Provably-Secure Logic Locking Techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[53] B. Tan *et al.*, "Benchmarking at the Frontier of Hardware Security: Lessons from Logic Locking," 2020. [Online]. Available: https://arxiv.org/abs/2006.06806

[54] FIRSuite. (2009) Suite of Constant Coefficient FIR Filters. [Online]. Available: https://www.firsuite.net/

[55] P. Subramanyan. HOST'15 Code Material. [Online]. Available: https://drive.google.com/file/d/19gYMK5lVaynzPhCNJlvzmI0eJmF6kVU0/view

[56] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, Tech. Rep. 12-93, 1993.

[57] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[58] Y. Shen. DoubleDIP Code Material. [Online]. Available: https://github.com/YuanqiShen/Double_DIP

[59] A. Alaql. SCOPE Code Material. [Online]. Available: https://github.com/alaql89/SCOPE

[60] Kaveh Shamsi. Netlist Encryption and Obfuscation Suite. [Online]. Available: https://bitbucket.org/kavehshm/neos/src/master/

Levent Aksoy received his Ph.D. degree in electronics engineering from Istanbul Technical University (ITU), Istanbul, Türkiye, in 2009. He worked as a researcher at ITU and INESC-ID, Lisbon, Portugal. He also worked at Dialog Semiconductor, Istanbul, Türkiye, as a senior digital design engineer. Currently, he is a post-doctoral researcher at the Centre for Hardware Security at Tallinn University of Technology (TalTech), Tallinn, Estonia. His research interests include hardware security and CAD for VLSI circuits with emphasis on solving EDA problems using SAT models and optimization techniques.

Quang-Linh Nguyen currently works as a Design-for-Test engineer at STMicroelectronics, Grenoble, France. He received a M.S. degree in Integrated Circuits and Systems from University of Paris-Saclay, Paris, France, in 2018 and a PhD degree in Micro-Electronics from University of Montpellier, Montpellier, France, in 2022. His research interests include VLSI Design, Design-for-Trust, Design-for-Test and Hardware Security.

Felipe Almeida received his bachelor's degree in Computer Engineering from the Pernambuco University and a master's degree in Microelectronics from the Federal University of Rio Grande do Sul. He is currently affiliated with the Centre for Hardware Security at Tallinn University of Technology (TalTech) as a Ph.D. student. His research interests are on Hardware Security and Radiation Tolerant Circuits.

Jaan Raik is a professor of digital systems' verification at the Department of Computer Systems and the head of the Centre for Dependable Computing Systems of TalTech University, Estonia. Prof. Raik received his M.Sc. and Ph.D. degrees at TalTech in 1997 and in 2001, respectively. He has co-authored more than 200 peer-reviewed scientific publications. His research interests cover a wide area in electrical engineering and computer science domains including hardware test, functional verification, fault-tolerance and security as well as emerging computer architectures.

Marie-Lise Flottes is a researcher for the French National Research Center (CNRS). Since 1990, she has been conducting research at LIRMM, Montpellier, France. She received her Ph.D. degree in 1990 from the University of Montpellier. Her interests include design for testability, testability and dependability of secure circuits, test data compression and test management for SoC and SiP.

Sophie Dupuis has been an Associate Professor with LIRMM, Montpellier, France, since 2011. She received her M.Sc. and Ph.D. degrees in microelectronics and design on integrated circuits from the Pierre & Marie Curie University, Paris, France, in 2004 and 2009 respectively. Her current research interests are oriented towards hardware trust, the design of trusted circuits despite potential untrustworthy design steps in particular.

Samuel Pagliarini received the PhD degree from Telecom ParisTech, Paris, France, in 2013. He has held research positions with the University of Bristol, Bristol, UK, and with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor at Tallinn University of Technology (TalTech) in Tallinn, Estonia where he leads the Centre for Hardware Security. His current research interests include many facets of digital circuit design, with a focus on circuit reliability, dependability, and hardware trustworthiness.

Appendix B

[II]
F. Almeida, L. Aksoy, Q. -L. Nguyen, S. Dupuis, M. -L. Flottes, and S. Pagliarini, "Resynthesis-based Attacks Against Logic Locking," in 24th International Symposium on Quality Electronic Design (ISQED), pp. 1-8, 2023.

Resynthesis-based Attacks Against Logic Locking

Felipe Almeida[†], Levent Aksoy[†], Quang-Linh Nguyen[‡], Sophie Dupuis[‡], Marie-Lise Flottes[‡], and Samuel Pagliarini[†]
[†]Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[‡]LIRMM, University of Montpellier, Montpellier, France
Email: [†] {felipe.almeida, levent.aksoy, samuel.pagliarini}@taltech.ee
Email: [‡] {quang-linh.nguyen, sophie.dupuis, marie-lise.flottes}@lirmm.fr

***Abstract*—Logic locking has been a promising solution to many hardware security threats, such as intellectual property infringement and overproduction. Due to the increased attention that threats have received, many efficient specialized attacks against logic locking have been introduced over the years. However, the ability of an adversary to manipulate a locked netlist prior to mounting an attack has not been investigated thoroughly. This paper introduces a resynthesis-based strategy that utilizes the strength of a commercial electronic design automation (EDA) tool to reveal the vulnerabilities of a locked circuit. To do so, in a pre-attack step, a locked netlist is resynthesized using different synthesis parameters in a systematic way, leading to a large number of functionally equivalent but structurally different locked circuits. Then, under the oracle-less threat model, where it is assumed that the adversary only possesses the locked circuit, not the original circuit to query, a prominent attack is applied to these generated netlists collectively, from which a large number of key bits are deciphered. Nevertheless, this paper also describes how the proposed oracle-less attack can be integrated with an oracle-guided attack. The feasibility of the proposed approach is demonstrated for several benchmarks, including remarkable results for breaking a recently proposed provably secure logic locking method and deciphering values of a large number of key bits of the CSAW'19 circuits with very high accuracy.**

***Index Terms*—Logic locking, resynthesis, EDA tools, oracle-less and oracle-guided attacks.**

I. Introduction

Due to the globalized integrated circuit (IC) supply chain, serious security threats, such as hardware Trojans, piracy, overbuilding, reverse engineering, and counterfeiting, have emerged [1]. Many defense techniques, such as watermarking [2], digital rights management [3], metering [4], and logic locking [5], have been introduced over the years to deal with these threats. Among those, logic locking stands out by being a well-established technique and by offering protection against a diverse array of adversaries [6]. Logic locking inserts additional logic driven by key bits so that the circuit behaves as expected only when the secret key is applied.

On the other hand, many efficient attacks have been introduced to overcome the defenses built by logic locking [7]. However, the impact of an electronic design automation (EDA) tool on the manipulation of the locked netlist **before** performing an attack has not been investigated thoroughly. In this work, we explore if EDA tools can be used to make a locked circuit vulnerable to existing logic locking attacks. Thus, the main contributions of this work are three-fold: (i) we introduce a resynthesis procedure that is a **pre-attack** step, where functionally equivalent but structurally different locked circuits are generated by resynthesizing the original locked circuit using different optimization parameters and delay constraints in order to create structural vulnerabilities that can be exploited by existing attacks; (ii) we present an oracle-less (OL) **resynthesis-based attack**, which applies the prominent SCOPE attack [8] to these resynthesized circuits and gathers all its solutions to discover the secret key; (iii) we show that our OL attack can be **combined** with a traditional oracle-guided (OG) attack for further improving the number of correctly deciphered key bits. The last contribution is essential since we consider circuits from the CSAW'19 contest – these circuits compound the use of two logic locking techniques at the same time.

The main finding of this work is that the use of many resynthesized locked circuits enables us to discover values of more key bits, and even the whole key, when compared to a single attack mounted on the original locked netlist.

The remainder of this paper is organized as follows: Section II presents the background concepts and related work. The resynthesis process and the proposed attacks are described in Section III. Experimental results are given in Section IV. Finally, Section V concludes the paper.

II. Background

A. Logic Locking and Threat Models

The procedure of logic locking is applied at the gate-level in the IC design flow, as shown in Fig. 1. Note that the layout of the locked circuit is sent to the foundry without revealing the secret key. After the locked IC is produced and delivered to the design house, the values of the secret key are stored in a tamper-proof memory, before the functional IC is sent to the market.

It is assumed that the gate-level netlist of the locked circuit can be obtained directly by an untrusted foundry or by reverse-engineering a functional IC obtained from the open market. An adversary can also use the functional IC

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 952252 (SAFEST).

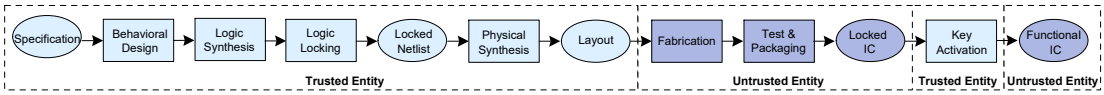


Fig. 1. Conventional logic locking in the IC design flow (adapted from [6]).

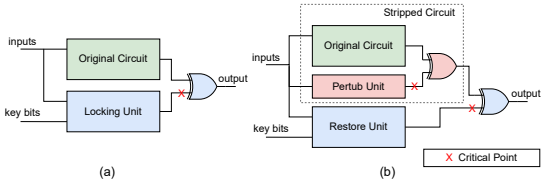


Fig. 2. SAT-resilient logic locking methods: (a) SFLT; (b) DFLT.

programmed with the secret key as an oracle to apply inputs and observe outputs. Thus, in logic locking, there are generally two threat models: OL and OG. In the OL threat model, only the gate-level netlist of the locked circuit is available to the adversary. The adversary has both the netlist of the locked circuit and the functional IC in the OG threat model.

B. Related Work

After the introduction of random logic locking (RLL) using XOR/XNOR gates in [9], earlier work focused on different types of key gates, such as AND/OR, multiplexors, and look-up tables, taking into account the hardware complexity of the locked circuit [5]. However, the OG satisfiability (SAT)-based attack [10] overcame all the defenses existing at that time. Note that the SAT-based attack iteratively finds distinguishing input patterns (DIPs) that rule out wrong keys. To thwart the SAT-based attack and its variants, circuits are locked using a point function that sets a limit on the number of wrong keys which a DIP can eliminate, forcing these attacks to explore an exponential number of queries [6], [11]–[14].

The SAT-resilient methods can be categorized into two groups: single-flip locking technique (SFLT) and double-flip locking technique (DFLT), as shown in Fig. 2. An SFLT has only one critical point, which is responsible to corrupt a protected output under a specific input pattern. Under this category, SARLock [15] adds a comparator and a masking circuit connected with the original netlist in a way that it generates a corruption on one input pattern. Anti-SAT [11] utilizes two complementary AND gate trees, whose output is merged with the original circuit. CASLock [12] is based on the same concept of Anti-SAT, however it uses both AND and OR gates. SKG-Lock [14] uses decoy key bits and provides a tunable output corruption. Note that SFLTs are susceptible to removal attacks [16]–[18]. If an attacker can identify this single critical point, he/she can split the design into a recovered netlist (original) and the locking unit.

A DFLT has two critical points, one that connects the original netlist with a perturbation unit and another one that connects the output of the stripped circuit with the restore unit. Under this category, stripped functionality logic locking (SFLL) [6], [13] initially corrupts an output based on an input combination in the perturbation unit and then, corrects this output only when the secret key is applied in the restore unit. Note that a removal attack becomes inefficient for a DFLT since the original circuit is mixed with the perturbation unit, even though it can easily identify the restore unit. However, there exist efficient structural attacks developed for DFLTs [19]–[22].

Alternative locking techniques have also been introduced. In [23], a technique, which has more than two critical points, called the multi-flip locking technique (MFLT), was proposed. However, it leads to a significant increase in area, power dissipation, and delay when compared to other techniques. Compound logic locking techniques were proposed to overcome the main drawback of a SAT-resilient technique, i.e., its low output corruptibility as can be observed in Fig. 2, by locking a design using both low and high output corruptibility techniques, such as SFLL and RLL, respectively [24]. Recently, efficient attacks have also been introduced against compound logic locking [25], [26].

Moreover, the OL attacks explore patterns in the structure of a locked netlist using statistical analysis [8], [27], [28]. For example, the SCOPE attack [8] is an unsupervised constant propagation technique, which analyzes each key bit of the locked design for critical features that can reveal its correct value after it is assigned to logic 0 and 1 value. These critical features include area, power dissipation, delay, and many other circuit characteristics obtained by a synthesis tool. These features are analyzed using linear regression and machine learning based clustering.

III. Proposed Resynthesis-based Attack

This section describes our resynthesis-based attack in detail. We assume a scenario, where the design house is the only trusted entity. An attacker has all possible reverse engineering, synthesis, and computing tools and has an access to the gate-level locked netlist and the functional IC. In this section, we initially introduce the pre-attack stage, where the locked circuit is resynthesized using different synthesis parameters, leading to a large number of structurally different netlists with the same functionality. Then, we present the OL attack that utilizes these resynthesized netlists in order to find the secret key. Finally, in order to handle the compound logic locking efficiently, we present its modified version, where our proposed OL attack cooperates with an OG attack.

A. The Pre-attack Step: Resynthesis of the Locked Netlist

The locked circuit is synthesized multiple times using a different script each time, where the synthesis parameters are explored in a systematic way. We use the following parameters to increase the number of resynthesized locked circuits:

Synthesis Effort: In a synthesis tool, logic optimizations can be applied with different efforts at different synthesis stages. This flexibility enables a designer to explore the trade-off between the quality of results and run time. The following efforts are considered at the given synthesis stage: low, medium, and high at generic transformations (*syn_gen*); low, medium, and high at mapping (*syn_map*); and low, medium, high, and extreme at optimization (*syn_opt*).

Delay Constraint: To meet performance targets, delay constraints are used to guide the synthesis tool. We initially resynthesize the locked circuit without a delay constraint and find the delay of its critical path, i.e., dcp. Then, in an interval between 0 and dcp, $d-1$ points, which are computed as $(dcp/d)i$ with $1 \leq i \leq d-1$, are set as delay constraints. Note that d is set to 5 in order to generate a large number of resynthesized circuits. Even though some delay constraints are impossible to meet, the synthesis tool *always* generates a netlist equivalent to the original one in terms of functionality.

Maximum Transition: The transition time of a net in a circuit is defined as the longest time required for its driving pin to change its logic value. The maximum transition value was chosen to be 5%, 10%, and 15% of the delay constraint for all the nets in the locked circuit to explore different resynthesized circuits.

Key Constraints: To direct the synthesis tool to work intensively on the paths that include the keyed logic, a delay constraint, which is impossible to be satisfied, can also be used. In this case, we force the delay between all key bits and all primary outputs to be 1 ps.

Thus, the combination of parameters given above generates $3 \times 3 \times 4 \times 5 \times 3 \times 2 = 1080$ netlists. We eliminate the resynthesized circuits with identical characteristics and keep only the unique ones. Additionally, we prevent the use of XOR/XNOR gates, which can be problematic for the SCOPE attack, during technology mapping. Note that our resynthesis methodology aims to generate different versions of the locked circuit, making it more vulnerable to existing attacks. Thus, any existing attack, either OL or OG, may potentially benefit from this pre-attack strategy to discover the secret key. The resynthesis process is automated for a commercial synthesis tool in a Perl script. This script, which can be modified for other synthesis tools, is available at *https://github.com/Centre-for-Hardware-Security/*.

B. Attacks on the Resynthesized Netlists

Time-efficient attacks are chosen in order to handle a large number of resynthesized circuits. In our OL resynthesis-based attack, SCOPE [8] is used to predict the values of key bits. In its modified version developed for compound logic locking, a query attack is used to find the values of key bits in a deterministic way.

1) Proposed OL Attack: SCOPE is applied to each resynthesized locked circuit and a solution is found. Note that this solution may return a logic 0, 1, or an unknown value for a key bit. Then, the values of key bits deciphered for each netlist are merged into a single solution that represents the overall guess. To do so, for each key bit, k_i with $1 \leq i \leq p$, where p denotes the number of key bits, we initially count the number of solutions, where k_i is deciphered as logic 0 and 1, denoted as dk_i^0 and dk_i^1, respectively. Then, if $dk_i^0 > dk_i^1$ or $dk_i^1 > dk_i^0$, the value of k_i is determined to be 0 or 1, respectively. Otherwise, in the case of a tie, the value of k_i is decided to be unknown.

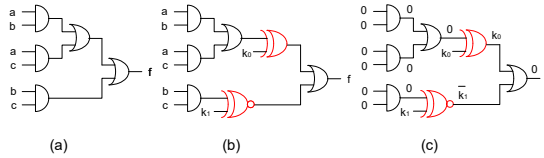


Fig. 3. (a) Majority circuit; (b) Locked majority circuit; (c) Constant propagation on the locked majority circuit.

2) Proposed OG Attack: In order to handle a large number of resynthesized netlists efficiently, we introduce a SAT-based query attack, which can determine the actual values of individual key bits. Note that traditional SAT-based attacks rather attempt to find the whole secret key, which increases the computational effort significantly. In this attack, we initially find queries, i.e., values of inputs of the oracle circuit, using two techniques. The first technique uses the ATPG tool Atalanta [29] to find test patterns for the stuck-at-fault of each key bit on the locked circuit and stores the values of the related primary inputs as queries. The aim is to find input patterns that can propagate each key bit to a primary output, making it observable. The second technique finds queries randomly. The aim is to find input patterns that may make multiple key bits observable at primary outputs. In our experiments, we generate a total of $2p$ queries, where p denotes the number of key bits.

Then, we describe the locked circuit in a conjunctive normal form (CNF) formula $\mathbb{C}$ by expressing each gate in its CNF. Each query is applied to the oracle and the values of primary outputs are obtained. Then, the related input and output values are assigned to the associated nets in the locked circuit, the constant values of these nets are propagated, and the Boolean equations including key bits are derived in a CNF formula $\mathbb{E}$. The SAT problem including the locked circuit in CNF, i.e., $\mathbb{C}$, is augmented with these equations, i.e., $\mathbb{C} = \mathbb{C} \wedge \mathbb{E}$. After all the queries are considered, the SAT problem $\mathbb{C}$ is solved using a SAT solver and the values of key bits are determined. Note that the locked circuit with the found values of key bits behaves exactly the same as the oracle under the given queries, but not under all possible input values. Hence, these key values are not guaranteed to be the values of the secret key.

However, the value found for a key bit can be proved if it is indeed equal to the actual value of the related bit in the secret key using the concept of *proof by contradiction*. To do so, for each key bit, the complement of its found value is added into $\mathbb{C}$ and the SAT solver is run. If there exists no solution to $\mathbb{C}$, i.e., the SAT problem is unsatisfiable, the value of the related key bit is proven to be the one in the found solution.

As a simple example, consider the majority circuit in Fig. 3(a) and suppose that it is locked using XOR/XNOR

TABLE I
DETAILS OF THE ISCAS'85 CIRCUITS.

Circuit	Original Netlist			Locked Netlist				
				p	Anti-SAT	CASLock	SFLL	SKG-Lock
	#in	#out	#gates		#gates	#gates	#gates	#gates
c2670	157	64	1193	64	1321	1320	1421	1401
c3540	50	22	1669	32	1733	1732	1783	1773
c5315	178	123	2307	64	2435	2434	2523	2514
c6288	32	32	2416	32	2480	2479	2531	2516
c7552	206	105	3512	64	3640	3639	3729	3713

gates as given in Fig. 3(b). Assume that a query is found as $abc = 000$ and thus, the value of its output f is obtained as 0 using the oracle. After propagating these values on the locked circuit as shown in Fig. 3(c), a Boolean equation $k_0 \vee \overline{k_1} = 0$, i.e., $\overline{k_0} \wedge k_1$ in CNF, is obtained. In the SAT solution, the key bit values are found as $k_0 k_1 = 01$. Note that these are the proven key values since a SAT solver guarantees that there exists no solution to the SAT problem $\mathbb{C}$, which is extended by either $k_0 = 1$, i.e., k_0 in CNF, or $k_1 = 0$, i.e., $\overline{k_1}$ in CNF, due to a conflict with the found Boolean equation, i.e., $\overline{k_0} \wedge k_1$ in CNF.

The query attack is run on all the resynthesized circuits and the proven values of key bits in each netlist are combined into a single solution. It is developed in Perl and is equipped with the incremental SAT solver CaDiCaL [30]. It is also available at *https://github.com/Centre-for-Hardware-Security/*.

Finally, the solution of the OG resynthesis-based attack is determined after merging the solution of the SCOPE attack over all resynthesized circuits into that of the query attack on all resynthesized circuits without changing the proven values of key bits.

IV. EXPERIMENTAL RESULTS

This section initially presents the results of the proposed OL resynthesis-based attack on the ISCAS'85 circuits [31] and then, those of the OG resynthesis-based attack on the CSAW'19 circuits [24] including compound logic locking.

A. Results on the ISCAS'85 Circuits

As the first experiment set, five ISCAS'85 circuits were considered. Table I presents their details. For our experiments, these circuits were locked by the Anti-SAT [11], CASlock [12], SFLL [6], and SKG-Lock [14] techniques. Note that while Anti-SAT and SFLL were taken from the NEOS tool [32], we implemented CASLock and SKG-Lock. Table I also presents details of the locked circuits. Note that the number of keys, i.e., p, was determined based on the number of inputs and overhead of the locking technique, and circuit characteristics, i.e., the number of inputs, outputs, and gates, were taken from the gate-level netlist.

Observe from Table I that all logic locking techniques lead to circuits with a number of gates close to each other, whereas the one locked by SFLL has a slightly large number of gates. Besides, the overhead on the number of gates in circuits locked by SFLL varies from 4.7% to 19.1% when compared to original circuits.

In the following subsections, we present the results of the resynthesis process and OL resynthesis-based attack, analyze the impact of synthesis parameters on the performance of the resynthesis process and SCOPE attack, and introduce improvements to the run-time of the resynthesis process.

1) Resynthesis of the Locked ISCAS'85 Circuits: The resynthesis is performed by Cadence Genus with a commercial 65 nm standard cell library. Table II presents the resynthesis results of locked circuits. In this table, *unique* denotes the number of unique locked netlists out of 1080 generated netlists and *area*, *delay*, and *power* stand respectively for the average values of the total area in μm^2, delay in the critical path in ps, and total power dissipation in μW on the unique locked netlists. Finally, *time* is the total run-time of the resynthesis process. The resynthesized netlists were generated on a computing server with Intel Xeon processing units at 3.9 GHz and a total of 1 TB memory.

Observe from Table II that the number of unique netlists is less than half of the total number of generated netlists, i.e., 540, except the *c3540* circuit locked by SKG-Lock. Note that Anti-SAT, CASLock, and SFLL lead to fewer unique netlists when compared to SKG-Lock, which is mainly because the logic added by these techniques is more compact than that added by SKG-Lock, which uses a chain of AND gates. We note that the synthesis tool consumes a large amount of time to fulfill a delay constraint that is impossible to meet, such as strict delay constraints and key constraints described in Section III-A. Hence, the run-time of the resynthesis process depends on the locked circuit and the logic locking technique, and more importantly, if there exists enough room for the synthesis tool to satisfy the constraints.

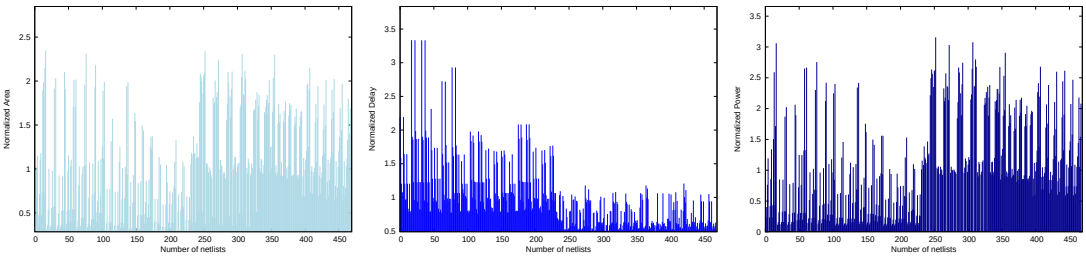
In order to illustrate the diversity of resynthesized netlists, the *c2670* circuit locked by SFLL is considered. Fig. 4 presents the area, delay, and power dissipation of each unique netlist, normalized by their average values given in Table II. Observe that resynthesis generates circuits significantly different from each other in terms of hardware complexity. The standard deviation on area, delay, and power dissipation values of all these netlists are computed as 1578, 235, and 4964, respectively. Note also that in this figure, the netlists after instance number 232 have a distinct profile, since they are generated using key constraints described in Section III-A.

In order to illustrate the differences in the structure of generated netlists, the *c2670* circuit locked by SKG-Lock is considered. Fig. 5 presents the graphs of two netlists resynthesized using the same synthesis parameters, except for the delay constraint. In this figure, red, green, and blue circles denote the inputs, key bits, and outputs, respectively; the gray triangles represent the gates. Observe that a small change in the delay constraint can lead to a structurally different netlist, where the difference between the number of gates and logic levels is 599 and 12, respectively.

2) Attacks on the Locked ISCAS'85 Circuits: Table III presents the results of the SCOPE attack on the original locked netlists and those of OL resynthesis-based attack on the unique locked netlists generated in the resynthesis process. In this

TABLE II
RESULTS OF RESYNTHESIZED LOCKED ISCAS'85 CIRCUITS.

Technique	Details	c2670	c3540	c5315	c6288	c7552
Anti-SAT	unique	480	537	464	498	439
	area	2357	2803	4112	7265	5387
	delay	504	818	663	2144	694
	power	5518	4934	4297	9403	7479
	time	17h14m51s	1d05h56m12s	1d09h56m22s	3d20h50m46s	1d16h01m13s
CASLock	unique	473	449	488	410	479
	area	2359	3112	4173	7739	5337
	delay	513	874	650	2146	676
	power	5170	3304	3852	10693	6765
	time	15h29m56s	1d11h02m52s	1d06h52m54s	4d03h12m29s	1d16h06m52s
SFLL	unique	468	484	477	523	504
	area	2817	3444	4326	7646	5340
	delay	481	870	697	2144	604
	power	6189	6337	9053	12115	11320
	time	13h13m23s	1d47m51s	21h57m07s	2d22h15m07s	22h40m29s
SKG-Lock	unique	521	541	507	527	521
	area	2673	2773	4646	6293	4774
	delay	936	986	782	2093	874
	power	3881	3831	8160	7201	7822
	time	22h22m01s	1d08h8m27s	1d03h56m15s	2d14h29m32s	1d04h19m



(a) (b) (c)

Fig. 4. Normalized complexity of resynthesized netlists of the c2670 circuit locked by SFLL: (a) area; (b) delay; (c) power.

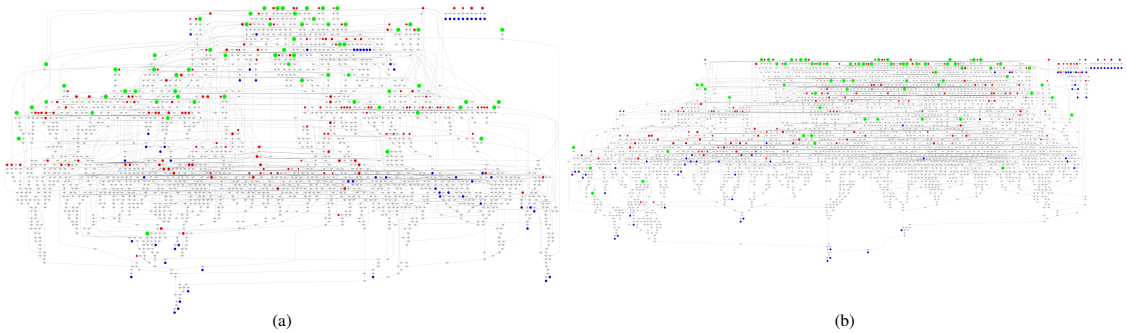
table, *cdk* and *dk* stand respectively for the number of correctly deciphered key bits and the total number of deciphered key bits and *time* is the total time required for the attack. The attacks were also run on the same server used to resynthesize the locked netlists.

Observe from Table III that the SCOPE attack is not entirely successful on any of the original locked netlists. However, the use of resynthesized netlists enables us to decipher the values of a large number of key bits, and even the whole key, e.g., for the *c2670* and *c3540* circuits locked by SKG-Lock. Note that the SCOPE attack can decipher almost all of the key bits using the resynthesized netlists locked by SKG-Lock. While the results on the netlists locked by SKG-Lock are all correct, the ones on the netlists locked by Anti-SAT, CASLock, and SFLL are slightly better than a random guess. The run-time of the SCOPE attack and our resynthesis-based attack depends mainly on the number of gates and keys in the locked design.

To find the SAT resiliency of resynthesized locked circuits, the SAT-based attack of [10] was run on 541 netlists of the *c3540* circuit locked by SKG-Lock with a time limit of 2 days. This circuit was chosen since it has the smallest number of key bits. Note that the SAT-based attack was not able to find the secret key of any resynthesized locked netlists. This experiment indicates that the resynthesis changes only the structure of the circuit as shown in Fig. 5, but maintains its SAT resiliency.

3) Redundant Synthesis Runs: Observe from Tables II and III that the total run-time of the proposed attack is dominated by the resynthesis process. However, it is possible to reduce the time required to resynthesize the locked netlist by removing redundant synthesis runs without sacrificing any unique netlists. For example, it is observed that the *high* value of the *syn_gen* parameter given in Section III-A can be removed from the parameter list, since all possible synthesis scripts including this parameter generate the same circuit when this parameter is *low* or *medium*. Thus, the number of generated circuits, i.e., 1080, reduces to 720.

4) Convergence on the Number of Deciphered Keys: It is also observed that the number of key bits deciphered by the SCOPE attack on all unique resynthesized netlists can actually be obtained using a small number of netlists. Fig. 6 presents the number of deciphered key bits along the unique resynthesized netlists of the *c2670* circuit locked by SKG-Lock. Observe from this figure that although a large number of unique netlists increases the quality of the SCOPE attack, actually a small number of unique netlists, 147 in this



(a) (b)

Fig. 5. Graphs of resynthesized netlists generated using a difference in the delay constraint dc: (a) dc is 990 ps; (b) dc is 496 ps.

TABLE III
RESULTS OF OL ATTACKS ON THE LOCKED ISCAS'85 CIRCUITS.

Circuit	Anti-SAT				CASLock				SFLL				SKG-Lock			
	SCOPE		Resynthesis		SCOPE		Resynthesis		SCOPE		Resynthesis		SCOPE		Resynthesis	
	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time
c2670	0/0	4s	37/64	34m18s	0/0	4s	35/64	33m47s	0/0	4s	34/64	37m32s	32/32	4s	64/64	44m37s
c3540	0/0	3s	17/32	21m27s	0/0	3s	17/32	18m12s	0/0	2s	19/32	21m29s	17/17	2s	32/32	24m30s
c5315	0/0	5s	38/64	42m34s	0/0	5s	30/64	43m54s	0/0	5s	33/64	46m23s	32/32	5s	62/62	52m06s
c6288	0/0	3s	18/32	29m08s	0/0	3s	16/32	27m18s	0/0	3s	16/31	33m19s	16/16	3s	31/31	34m24s
c7552	0/0	6s	38/64	45m31s	0/0	6s	47/64	49m13s	0/0	6s	38/63	52m26s	32/32	6s	61/61	56m45s

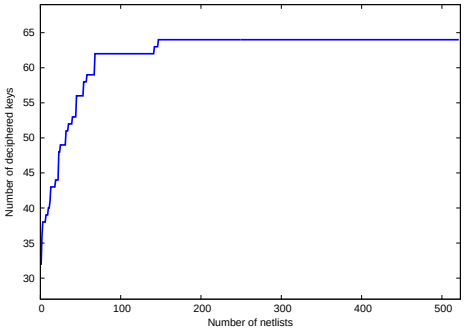


Fig. 6. Convergence on the number of deciphered keys over the number of resynthesized netlists in the SCOPE attack.

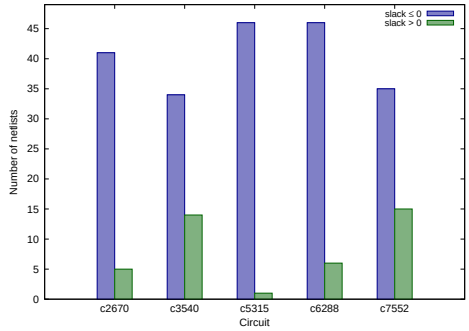


Fig. 7. Classification of resynthesized netlists based on their slack values on promising solutions of SCOPE attack.

case, is sufficient to achieve the same result as when all 521 unique netlists are considered. We note that a similar situation was also observed on circuits locked by other techniques.

5) Promising Resynthesized Netlists: Moreover, it is observed that the SCOPE attack is more successful on specific resynthesized netlists. To find a set of synthesis parameters that enables the SCOPE attack to decipher more key values, we initially define two categories of netlists based on the slack time of the design, i.e., the difference between the required and arrived time in the critical path, as follows: i) netlists with a slack value less than or equal to 0; ii) netlists with a slack value greater than 0. The slack value of a design gives indeed a rough idea of the effort put in by the synthesis tool; for the netlists in the first category, the synthesis tool works extremely hard to meet the delay constraint, trying many logic transformations and optimization techniques.

Then, the solutions of the SCOPE attack on all possible 1080 netlists are obtained and sorted based on the number of deciphered key bits in descending order. The top 10% of these sorted netlists are categorized based on their slack values.

Fig. 7 presents the results of this experiment on the circuits locked by SKG-Lock. Observe that the netlists that enable the SCOPE attack to decipher more key values generally have a slack value less than or equal to 0. Thus, to generate such circuits, one can easily add strict delay constraints or key constraints as described in Section III-A. We note that a similar result was also observed on resynthesized netlists locked by other techniques.

6) Structural Analysis: In order to improve the performance of the resynthesis process, the logic cone, which is the locking technique is applied on, can be extracted and resynthesized. Note that the output of this logic cone is a single primary

TABLE IV
RESULTS OF THE RESYNTHESIS PROCESS AND OL RESYNTHESIS-BASED ATTACK ON ENTIRE CIRCUIT AND LOGIC CONE.

Circuit	Entire Circuit			Logic Cone		
	unique	time	cdk/dk	unique	time	cdk/dk
c2670	468	13h13m23s	34/64	319	7h46m26s	34/64
c3540	484	1d47m51s	19/32	320	6h29m35s	16/32
c5315	477	21h57m14s	33/64	313	7h6m16s	32/64
c6288	523	2d22h15m7s	16/31	302	6h20m57s	19/32
c7552	504	22h40m29s	38/63	279	6h57m14s	38/63

TABLE V
DETAILS OF THE LOCKED CSAW'19 CIRCUITS.

Circuit	Details			Number of key bits		
	#in	#out	#gates	RLL	SFLL-rem	Total
small	522	512	15995	40	40	80
medium	767	757	24008	60	60	120
large	1452	1445	36584	80	80	160
bonus	892	1746	23004	128	128	256

output, while its inputs are primary inputs, but not necessarily all the primary inputs of the locked design. Thus, the run-time of the resynthesis process can be decreased, since the logic cone has a small number of inputs, outputs, and gates when compared to the whole locked circuit.

Table IV presents details on the resynthesis process on entire locked circuits and logic cones when the circuits locked by SFLL are used. Observe that the resynthesis process on a logic cone generates less number of unique designs and takes significantly less time without a significant loss on the solution quality when compared to the resynthesis process on the entire circuit. We note that similar results were also observed on circuits locked by other techniques.

B. Results on the CSAW'19 Circuits

As the second experiment set, we used the state-of-the-art locked circuits from the CSAW'19 contest [24]. Details of these circuits are given in Table V. Note that two logic locking techniques – RLL [9] and SFLL-rem [13] – are applied together to lock a circuit.

In the following two subsections, we present the results of the resynthesis process and the resynthesis-based attack.

1) Resynthesis of the Locked CSAW'19 Circuits: Table VI presents the resynthesis results of locked circuits. Observe that the number of unique resynthesized netlists is larger than half of the total number of generated netlists, i.e., 540. As the hardware complexity of designs increases, the run-time of the resynthesis process increases. We note that diverse netlists in terms of complexity are obtained, e.g., the standard deviation on area, delay, and power dissipation values of all the locked netlists of the *small* circuit are computed as 8526, 1029, and 20074, respectively.

2) Attacks on the Locked CSAW'19 Circuits: Table VII presents results of the attacks obtained, after they are applied to the original locked netlist, denoted as *OLN*, and all unique resynthesized netlists, denoted as *URNs*. In this table, *prv* stands for the number of proven values of key bits. Note that since the secret key is **not publicly available**, the *cdk* values are omitted for the SCOPE and resynthesis-based attacks.

TABLE VI
RESULTS OF RESYNTHESIZED LOCKED CSAW'19 CIRCUITS.

Circuit	unique	area	delay	power	time
small	557	18935	1631	23571	5d3h22m28s
medium	569	26080	1745	31284	6d12h24m16s
large	567	31348	1798	24610	5d21h42m10s
bonus	560	20643	1758	19090	4d14h44m29s

TABLE VII
RESULTS OF ATTACKS ON THE LOCKED CSAW'19 CIRCUITS.

Circuit-Netlist	SCOPE		Query		Resynthesis	
	dk	time	prv	time	dk	time
small - OLN	19	20s	39	1m21s	40	1m41s
small - URNs	77	4h10m42s	40	1d10h4m37s	79	1d14h15m19s
medium - OLN	32	41s	58	6m37s	59	7m18s
medium - URNs	117	8h33m56s	58	3d19h12m13s	120	4d3h46m9s
large - OLN	30	1m7s	79	6m19s	79	7m26s
large - URNs	152	12h56m15s	80	3d2h52m11s	159	3d15h48m26s
bonus - OLN	64	1m46s	118	3m2s	120	4m48s
bonus - URNs	233	16h7m17s	125	1d20h29m22s	252	2d12h36m39s

Observe from Table VII that the original SCOPE attack could only decipher a small number of key bits, all of which belong to RLL, and the query attack can prove the values of a large number of key bits, all of which again belong to RLL, on the original locked circuits. Thus, the resynthesis-based attack could only decipher the RLL key bits on the original locked circuits. However, the use of resynthesized circuits makes the SCOPE attack decipher more key bits that also belong to SFLL-rem and makes the query attack prove the values of more key bits that belong to RLL. Thus, the resynthesis-based attack could decipher almost all the values of the secret key, proving almost all the values of the key bits of RLL. Note that all the unknown key bits belong to SFLL-rem. Observe that the run-time of attacks increases, as the number of gates and key bits increases.

After the values of key bits of the CSAW'19 circuits were determined, they were sent to the contest organizers for evaluation. Table VIII presents the results of the resynthesis-based attack along with those of other techniques which participated in the contest.

Observe from Table VIII that our proposed attack can determine all the key bits of RLL correctly, even though there are unproven key bits in the *medium* and *bonus* circuits as shown in Table VII. This observation implies that the guesses of the SCOPE attack on those key bits are actually correct. Moreover, the proposed technique can decipher the key bits of SFLL-rem with a number of deciphered key bits greater than any other OL technique with high accuracy.

V. CONCLUSIONS

This work has shown that EDA tools can be used to generate variants of locked circuits that may be vulnerable to existing logic locking attacks and such circuits can be generated using a specific set of synthesis parameters. It was shown that the run-time of the proposed technique can be improved using a small number of resynthesized netlists without diminishing its solution quality. Experimental results clearly indicated that the use of many resynthesized circuits enables existing attacks to decipher values of a large number of key bits with high

TABLE VIII
RESULTS OF ATTACKS ON THE LOCKED CSAW'19 CIRCUITS.

Approach	Attack Scenario	Circuit							
		small (40+40)		medium (60+60)		large (80+80)		bonus (128+128)	
		RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem
Key sensitization [33]	OG	40/40	—	60/60	—	80/80	—	—	—
Hamming distance-based attack [24]	OG	30/30	—	50/50	—	72/72	—	—	—
Automated analysis + SAT [24]	OG	11/18	—	31/50	—	10/34	—	—	—
Sub-circuit SAT [24]	OG	17/17	—	29/29	—	—	—	—	—
Redundancy-based [27]	OL	28/28	4/12	35/35	23/28	45/45	0/51	66/66	8/27
Bit-flipping attack [34]	OG	40/40	—	60/60	—	80/80	—	—	—
Topology guided attack [28]	OL	15/32	—	19/50	—	36/73	—	75/108	—
Resynthesis-based attack	OG	40/40	20/39	60/60	29/60	80/80	35/79	128/128	55/124

accuracy. Hence, the resynthesis of a locked circuit can be utilized as a pre-attack step for many existing attacks in order to improve their success rate. As future work, we plan to consider other synthesis parameters, such as fanout, capacitance limits, and wire loads, which enable synthesis tools to generate different circuits. Also, we aim to incorporate other commercial and open source EDA tools into the resynthesis process to generate different unique netlists.

ACKNOWLEDGMENT

The authors thank Nimisha Limaye for evaluating the keys found by the proposed technique on the CSAW'19 benchmarks.

REFERENCES

[1] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[2] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *DAC*, 1998, pp. 776–781.

[3] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote Activation of ICs for Piracy Prevention and Digital Right Management," in *ICCAD*, 2007, pp. 674–677.

[4] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, 2012.

[5] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *J. Electron. Test.*, vol. 35, no. 3, pp. 273–291, 2019.

[6] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *ACM CCS*, 2017, pp. 1601–1618.

[7] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Threats on Logic Locking: A Decade Later," in *GLVLSI*, 2019, pp. 471–476.

[8] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE TVLSI*, vol. 29, no. 8, pp. 1529–1542, 2021.

[9] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *DATE*, 2008, pp. 1069–1074.

[10] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *HOST*, 2015, pp. 137–143.

[11] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE TCAD*, vol. 38, no. 2, pp. 199–207, 2019.

[12] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 175–202, 2019.

[13] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-Based Perspective," *IEEE TCAD*, vol. 39, no. 12, pp. 4439–4452, 2020.

[14] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *ISVLSI*, 2021, pp. 114–119.

[15] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *HOST*, 2016, pp. 236–241.

[16] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," in *Cryptographic Hardware and Embedded Systems*, 2017, pp. 189–210.

[17] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[18] A. Sengupta, N. Limaye, and O. Sinanoglu, "Breaking CAS-Lock and Its Variants by Exploiting Structural Traces," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, p. 418–440, 2021.

[19] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," in *DATE*, 2019, pp. 936–939.

[20] F. Yang, M. Tang, and O. Sinanoglu, "Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) – Unlocked," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2778–2786, 2019.

[21] Z. Han, M. Yasin, and J. Rajendran, "Does Logic Locking Work with EDA Tools?" in *USENIX Security Symposium*, 2021, pp. 1055–1072.

[22] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability Assessment Tool and Attack for Provably-Secure Logic Locking Techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[23] Y. Liu, M. Zuzak, Y. Xie, A. Chakraborty, and A. Srivastava, "Strong Anti-SAT: Secure and Effective Logic Locking," in *ISQED*, 2020, pp. 199–205.

[24] B. Tan *et al.*, "Benchmarking at the Frontier of Hardware Security: Lessons from Logic Locking," 2020. [Online]. Available: https://arxiv.org/abs/2006.06806

[25] M. John, A. Hoda, R. Chouksey, and C. Karfa, "SAT Based Partial Attack on Compound Logic Locking," in *Asian Hardware Oriented Security and Trust Symposium*, 2020, pp. 1–6.

[26] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based Attack on Compound Logic Locking Techniques," in *DATE*, 2021, pp. 1166–1171.

[27] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *DATE*, 2019, pp. 540–545.

[28] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "TGA: An Oracle-Less and Topology-Guided Attack on Logic Locking," in *ASHES*, 2019, p. 75–83.

[29] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Tech. Rep. 12-93, 1993.

[30] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[31] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," in *ISCAS*, 1985, pp. 663–698.

[32] K. Shamsi, "Netlist Encryption and Obfuscation Suite," 2021. [Online]. Available: https://bitbucket.org/kavehshm/neos/src/master/

[33] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *DAC*, 2012, pp. 83–89.

[34] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based Bit-Flipping Attack on Logic Encryptions," in *DATE*, 2018, pp. 629–632.

Appendix C

[III]
F. Almeida, L. Aksoy, and S. Pagliarini, "RESAA: A Removal and Structural Analysis Attack Against Compound Logic Locking," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2025. Accepted for publication.

RESAA: A Removal and Structural Analysis Attack Against Compound Logic Locking

Felipe Almeida[1], Levent Aksoy[1] and Samuel Pagliarini[1 2]
[1]Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[2]ECE Department, Carnegie Mellon University, Pittsburgh - PA, USA
Email: [1]{felipe.almeida, levent.aksoy, samuel.pagliarini}@taltech.ee
Email: [2]pagliarini@cmu.edu

***Abstract*—The semiconductor industry's paradigm shift towards fabless integrated circuit (IC) manufacturing has introduced security threats, including piracy, counterfeiting, hardware Trojans, and overproduction. In response to these challenges, various countermeasures, including logic locking (LL), have been proposed to protect designs and mitigate security risks. LL is likely the most researched form of intellectual property protection for ICs. A significant advance has been made with the introduction of compound logic locking (CLL), where more than one LL technique is concurrently utilized for improved resiliency against attacks. However, the vulnerabilities of LL techniques, particularly CLL, need to be explored further. This paper presents a novel framework, RESAA, developed to classify designs locked by CLL, identify critical gates, and execute various attacks to uncover secret keys. RESAA is agnostic to specific LL techniques, offering comprehensive insights into CLL's security scenarios. Experimental results demonstrate RESAA's efficacy in identifying critical gates, distinguishing segments corresponding to different LL techniques, and determining associated keys based on different threat models. In particular, for the oracle-less threat model, RESAA can achieve up to 92.6% accuracy on a relatively complex ITC'99 benchmark circuit. The results reported in this paper emphasize the significance of evaluation and thoughtful selection of LL techniques, as all studied CLL variants demonstrated vulnerability to our framework. RESAA is also open-sourced for the community at large.**

***Index Terms*—Compound logic locking, oracle-less attacks, oracle-guided attacks, electronic design automation.**

I. Introduction

The evolution of the semiconductor industry and the migration to a fabless integrated circuit (IC) ecosystem has been revolutionary [1]. Outsourcing IC fabrication to third-party foundries and incorporating third-party intellectual properties (IPs) has significantly transformed the security dynamics in chip design. This shift has brought forth a spectrum of security threats, such as IC counterfeiting, IP piracy, IC overproduction, and the insertion of hardware Trojans. All these threats undermine the integrity of the IC supply chain [2].

Counterfeiting leads to the unauthorized replication of ICs, which can be associated with a loss in both quality and reliability [3]. Piracy involves the illicit use of IP fueling the production of counterfeit ICs [2]. Overproduction increases the proliferation of counterfeit products by manufacturing beyond authorized quantities [4]. Hardware Trojans are pieces of malicious logic inserted into a design, potentially compromising its functionality and/or reliability [5].

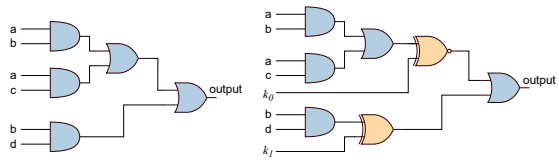


Fig. 1: (a) Original circuit; (b) Locked circuit where the secret key is $k_0k_1 = 10$.

As chip design and fabrication grow increasingly complex, maintaining the integrity of ICs and their IPs has emerged as a preeminent concern. In response to these emerging threats, researchers have proposed diverse countermeasures, such as split manufacturing, hardware metering, watermarking, and logic locking (LL). In split manufacturing design, the metal stack is divided across different foundries to mitigate security risks [6]. Hardware metering involves real-time monitoring of resource usage within IC against piracy using mechanisms designed to track and regulate the allocation of hardware resources, ensuring efficient utilization while maintaining security [7]. Watermarking embeds signatures into designs, without altering functionality, to detect IP theft and misuse [8]. LL stands out as likely the most researched technique; however, it offers only *potential* protection against several security threats [9]–[13]. The principle behind LL is the insertion of additional logic driven by key bits, such that the locked circuit behaves like the original circuit only when the secret key is provided, as shown in Fig. 1.

Over nearly two decades, researchers have strived to improve LL techniques by developing strategies that promote output corruption, deliver resilience against attacks, and reduce associated overheads [14]. Specific LL methodologies, like random logic locking (RLL), have been fine-tuned to simultaneously reduce area overhead and increase output corruption. Fig. 1 shows an example of RLL, which involves the insertion of additional gates controlled by key inputs, ensuring proper functionality solely when the secret key is applied [9]. However, its security has been compromised, primarily by attacks under the oracle-guided (OG) threat model [15]. The most prominent of these attacks is known as the SAT-based attack [15].

To counter the SAT attack, SAT-resilient techniques have

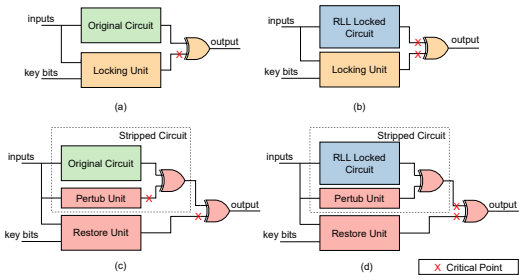


Fig. 2: High-level architecture of (a) SFLT (b) RLL + SFLT (c) DFLT, and (d) RLL + DFLT in a CLL scheme. The critical signals are indicated by "X" in red.

been developed under the name of provably secure logic locking (PSLL) [16]. A single-flip locking technique (SFLT) [10], [17], [18] incorporates a point function that introduces an additional block (locking unit), activated by key bits, to secure a specific output of the IC as shown in Fig. 2(a). Conversely, a double-flip locking technique (DFLT) [19], [20] employs a point function but enhances security by inserting a perturb unit and a restore unit, followed by output correction, as shown in Fig. 2(c). Although these techniques aim to bolster resilience against SAT specifically, they remain vulnerable to various attacks, under both OG and oracle-less (OL) threat models. Additionally, certain variants may be susceptible to removal and structural analysis attacks [21], [22].

Approaches to address the perceived weakness of LL techniques have taken many forms, including the deliberate insertion of cyclic logic [23], the use of emerging materials [24], and look-up table based obfuscation [25], [26]. A hybrid approach has emerged, too, termed compound logic locking (CLL), which combines high-output corruptibility and SAT-resilient LL techniques to address vulnerabilities that are present in obfuscated circuits when either technique is utilized alone [27], [28]. As a general trend, there is limited knowledge about attacks that are CLL-aware. In [27], authors have explored the combined use of SAT-based and structural attacks against CLL. It is worth noting that current CLL research is limited to a single combination of techniques, which highlights the need for a more comprehensive exploration.

A. Scope of this work

This work explores attacks on CLL, an advanced IP protection approach characterized by a multi-layered application of existing LL techniques. We examine the effects of combining two LL techniques to enhance the security of digital circuits. In this scenario, we first lock the original circuit using RLL and then, apply a PSLL technique. Although CLL is generally described as a more resilient approach, our study reveals that it may not necessarily mitigate known vulnerabilities. Instead, CLL could perpetuate or even worsen weaknesses inherent to the individual LL techniques. In some cases, combining SAT-resilient techniques with traditional LL has still left designs vulnerable to removal attacks, where attackers can bypass the locking mechanisms without finding the secret key. We aim to clarify how, in specific scenarios, CLL may fail to address these vulnerabilities, potentially fully exposing the original circuit.

We put forward RESAA, a comprehensive framework for: (i) identifying critical gates (CGs), (ii) classifying locked designs based on LL techniques, (iii) partitioning the design to apply well-established attacks, and (iv) potentially finding the secret key using prominent LL techniques. Our investigations reveal potential security vulnerabilities inherent in CLL compared to using a singular LL technique. These findings emphasize the intricate complexities and associated pitfalls of CLL strategies, highlighting the importance of thorough evaluation and careful use of LL techniques.

In contrast to previous studies [27], [28], our methodology takes a practical approach by relying solely on commercial synthesis tools[1]. In other words, RESAA is a framework for security analysis that can be readily used in existing design flows. Furthermore, all the circuits we study in this work are mapped to a commercial cell library for the sake of realism. This industry-minded approach to our CLL analysis provides insights that are directly applicable to real-world situations, ensuring that our findings address the challenges and limitations encountered by design houses attempting to protect their commercial IP.

B. Contributions

1) We present the RESAA framework, which identifies CGs in CLL designs and differentiates between RLL and PSLL techniques.
2) RESAA effectively divides designs into two parts, enabling the use of OL and OG attacks to uncover the secret key, exposing vulnerabilities in multiple CLL techniques.
3) RESAA is integrated within an industry-grade logic synthesis tool to ensure a realistic setting.
4) We open source scripts and strategies utilized in RESAA[2]
5) We expose the keys for large CLL-locked combinational circuits, even under the difficult OL setting, with a high accuracy.

It is also important to emphasize that our approach does not assume that the adversary is aware of the combination of techniques that make up the chosen CLL scheme. Instead, we analyze and classify netlists in order to automatically identify the SFLT/DFLT technique being employed. This aspect significantly *distinguishes* our work from existing methodologies that rely on assumptions.

The subsequent sections of this paper are structured as follows: Section II elaborates more on the foundational concepts surrounding (C)LL. The RESAA framework is described in full detail in Section III. Our experimental results are thor-

[1]Without loss of generality, our methodology can be easily adapted to open-source logic synthesis tools.

[2]See https://github.com/Centre-for-Hardware-Security/CLL_attack

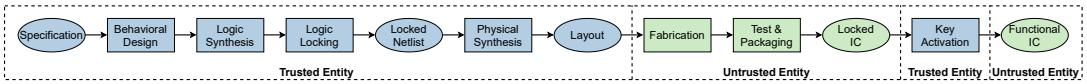


Fig. 3: Conventional logic locking in the IC design flow (adapted from [16]).

oughly covered in Section IV. Finally, Section V summarizes our findings.

II. Background and Related Work

This section presents details on LL, the OG and OL threat models, and existing defenses and attacks, indicating the never-ending cat-and-mouse game.

A. Logic Locking and Threat Models

Fig. 3 presents the conventional IC design flow augmented by using LL. The locking process is a security measure typically implemented at the gate level. LL aims to encode the circuit's functionality using key inputs, ensuring that unauthorized parties cannot access or replicate the design's core functionality without the corresponding secret key. Notably, implementing LL allows the design house to protect its proprietary information while engaging with third-party foundries for manufacturing. Once the locked IC design is finalized, the layout is sent to the foundry for fabrication. Upon completion of manufacturing, the values of the secret key are securely stored within a memory that is designed to resist unauthorized access and tampering attempts. Subsequently, once the functional ICs are prepared for market distribution, the locked design and its associated secret keys remain securely protected within the confines of the design house, thereby mitigating the risks associated with IP piracy and unauthorized overproduction.

LL is a countermeasure extensively researched to protect IP against several threats in the semiconductor industry. This technique involves strategically inserting extra logic gates into the design [9]–[13]. These added gates operate under the control of key inputs, ensuring that the locked design behaves the same as the original one if the secret key is applied. Otherwise, it generates a wrong (corrupted) output. LL has displayed both effectiveness and vulnerability against various attack methodologies, prompting the development of dozens of variants of the technique [14], [29].

Threat models play a critical role in identifying potential adversaries and their capabilities, which are essential for securing hardware designs against attacks. The existing literature considers two distinct threat models: the OG and OL threat models. Under both threat models, the adversary is assumed to have the locked netlist. Moreover, under the OG threat model, the adversary has a functional IC, which is used as an oracle to apply inputs and observe outputs.

In this work, we exercise adversarial capabilities in **both OG and OL threat models**. We assume that an adversary with access to a modern logic synthesis tool. We also assume the adversary can trivially differentiate key inputs from primary inputs[3]. However, the adversary does not know *a priori* which key input corresponds to which LL technique in CLL. Finally, we assume that the adversary is familiar with existing attacks and can apply them freely, including the following attacks: SAT-based [15], AppSAT [30], SCOPE [31], DoubleDIP [32], and query-based [33] methods. The attack itself can be carried out on a standard configuration laptop or desktop, with no need for specialized computational resources or servers. Additionally, the adversary is assumed to have access to logic synthesis tools and standard cell libraries if resynthesis is sought[4].

TABLE I: SA, KA, and OA attacks.

Attack	SA	KA	OA
SAT-based [15]	✓	×	×
AppSAT [30]	✓	×	×
Double DIP [32]	✓	×	×
Fa-SAT [28]	✓	×	×
Valkyrie [35]	✓	✓	×
This Work (RESAA)	✓	✓	✓

Furthermore, a more nuanced approach to determining adversarial knowledge is the classification of specific adversaries (SA), knowledgeable adversaries (KA), and oblivious adversaries (OA). As the names imply, there are varying levels of awareness regarding the LL techniques employed. Table I compares various tools/attacks for exploring vulnerable LL designs. The first tools were developed for specific LL techniques, with the adversary designing the attack exclusively for those techniques (therefore classified as SA). For example, SAT-based attack [15] and AppSAT [30] were initially aimed at RLL, while Double DIP specifically targeted SARLock designs. However, these attacks have been adapted and can now be applied to a broader range of LL techniques. Fa-SAT [28] was one of the first studies to explore CLL attacks, but only for a few specific combinations of LL techniques, such as bilateral logic encryption (BLE) [34] and Anti-SAT coupled with RLL. In contrast, Valkyrie [35] proposed a framework to attack fifteen LL techniques, which yielded satisfactory results, but the adversary knew the techniques employed in advance (therefore classified as a KA tool). Our study explores a more generic tool capable of achieving remarkable results without *a priori* knowledge about the CLL techniques employed (therefore, RESAA is classified as OA). This setting often has been employed in logic locking competitions.

B. Defenses

The first attempts at obfuscating a design were of the RLL form, an attempt to improve security using XOR/XNOR gates driven by key inputs and internal signals [9]. RLL can easily

[3]This is considered possible by tracing the key inputs back to the tamper-proof memory.

[4]This assumption is reasonable if the adversary is located at the foundry.

achieve high output corruption if enough key gates are inserted. Researchers have then modified the original RLL technique to strengthen its resilience against various attacks. For example, fault analysis-based methods [36] focus on analyzing the susceptibility of the locked circuit to fault attacks, and strong-interference-based LL approaches [37] are designed to create interference that is strong enough to confuse or mislead an attacker attempt, ensuring that adversaries cannot easily determine the correct keys by creating complex dependencies between them. However, the emergence of SAT attacks, which iteratively find distinguishing input patterns (DIPs) that rule out wrong keys, has highlighted vulnerabilities in these methods [15]. A DIP is an input combination that produces different outputs for two distinct key values. Consequently, a DIP provides an unequivocally incorrect key value that can be eliminated. This has led to the exploration of many other alternative LL techniques.

Figs. 2(a) and 2(c) show how the SAT-resilient LL techniques are divided into two main categories: SFLT and DFLT, respectively. SFLT uses a single critical signal which corrupts the original circuit. SFLT techniques use a point function in their locking unit, thus forcing a SAT-based attack to explore an exponential number of DIPs. The use of a point function makes key recovery increasingly challenging as the SAT problem size grows with each iteration [19]. However, the characteristic of having only one critical signal can be exploited for the potential recovery of the original design via a removal attack aimed precisely at this crucial signal [38]. In contrast, DFLT techniques introduce an additional critical signal in an attempt to be resilient against removal attacks. DFLT combines a restore and a perturb unit to secure circuits. While the perturb unit disrupts the correct operation on a single input pattern, the restore unit ensures that the correct functionality is restored using the secret key. This dual-phase approach enhances security and resists reverse engineering and SAT-based attacks. Even if the restore unit is removed in a DFLT, the original circuit design remains unrecoverable because of the perturb unit, as shown in Fig. 2(c). Consequently, DFLT provides enhanced protection by introducing an added layer of security, thwarting unauthorized access, and preserving the locked circuit's integrity.

In the SFLT category, the Anti-SAT [17] method incorporates a locking unit that executes a complementary function. This unit is structured around two distinct functions, denoted as g and $\bar{g}$, which are ANDed to generate a single critical signal. Subsequently, this critical signal is integrated with the original design by an XOR gate that drives the output of the original design. The Anti-SAT-DTL [39] approach has a diversified tree logic (DTL), where some AND gates in the AND-tree are replaced by OR/NAND/XOR gates. In CASLock [18], the g block is constructed using cascaded AND-OR gates, which serve as its distinctive feature. The effectiveness of CASLock lies in its incorporation of OR gates within the g and $\bar{g}$ blocks. This design attribute enables CASLock to effectively counter-bypass attacks by strategically altering the placement and quantity of AND/OR gates within the g and $\bar{g}$ blocks. SARLock [10] introduces a comparator and a masking circuit connected to the original netlist, causing corruption on a specific input pattern.

DFLT techniques, such as stripped functionality logic locking (SFLL), strip some functionality from the original design, corrupting its output corresponding to protected input patterns [16], [19]. SFLL includes a comparator as a restore unit for particular input vectors, with the ability to utilize a configurable hamming distance (HD) for protecting multiple input vectors [16]. Note that tenacious and traceless logic locking (TTLock) is equivalent to SFLL when HD is equal to 0 [19].

Figs. 2(b) and 2(d) exemplify CLL which integrates multiple LL techniques to enhance the security of ICs. Note that RLL is always used as it delivers the important feature of (high) output corruption. By combining RLL with other techniques, CLL endeavors to leverage their respective strengths while mitigating individual weaknesses. This integration strategy aims to fortify security by exploiting the complementary aspects of diverse LL techniques, selecting corruption levels and SAT resilience tailored to optimize the desired trade-off between security and corruptibility.

C. Attacks

LL has received much attention for being implementable at the front-end stage (i.e., as direct modifications to a netlist) without requiring layout modifications or foundry collaboration (as is the case with split manufacturing). However, its security has been challenged by various emerging attacks that have repeatedly succeeded at exposing secret keys. OG attacks involve comparing a locked design with an activated device, using the activated circuit as an oracle. This allows an adversary to examine the differences between the locked and unlocked designs, enabling them to compare the outputs of the original circuit with those of the locked circuit [15], [40]. The most well-known OG attack, the SAT-based attack, iteratively finds DIPs to break LL techniques [15], [40]. AppSAT [30] reduces the time to breach LL by finding an approximate solution instead of fully solving the SAT problem. At the same time, Double DIP [32] enhances the SAT-based attack by finding two DIPs in each iteration. Query attacks are another type of SAT-based attack, where each query is applied to the oracle, and the values of primary outputs are obtained [33]. Each query q corresponds to a distinct input or a set of inputs furnished by the attacker to the oracle. Then, the query attack finds the values of secret key based on these queries and proves the value of each found secret key bit is the same as in the original secret key using the concept of proof by contradiction.

In contrast, OL attacks focus on extracting sensitive information from a locked IC without direct access to an oracle. Adversaries in OL attacks possess only the locked design netlist. Techniques such as machine learning, constant propagation analysis, and resynthesis-based approaches are commonly employed in the OL setting to extract information from the locked IC [31], [41], [42]. Note that, by definition, OL is a much more difficult setting for the adversary than

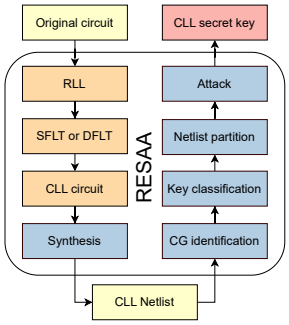


Fig. 4: Overview of the RESAA framework: Pre-processing step to lock using CLL and obtaining the mapped netlist (left) and attack on the CLL netlist (right).

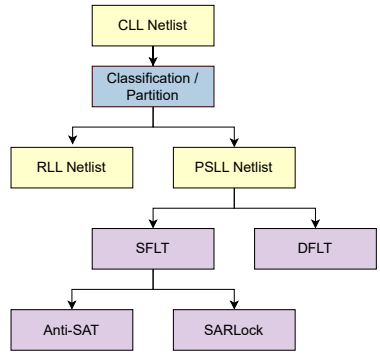


Fig. 5: Classification of techniques employed in a CLL netlist.

OG is. The SCOPE attack, which is a prime example of an OL attack, compromises the locked design by leveraging synthesis-based constant propagation. Unlike traditional SAT-based attacks that need an oracle to compare circuits, SCOPE analyzes and simplifies circuits during synthesis. SCOPE identifies and propagates constant values through the circuit, effectively reducing the complexity of the LL and ultimately producing a guess of the correct key with a relatively high degree of certainty.

Yet, both OG and OL attacks have shown effectiveness against specific LL techniques [14]. We will consider both of them when introducing our framework RESAA.

III. Proposed Methodology

This section describes our attack strategy explicitly aimed at CLL-protected circuits. The scenario begins with locking the original design, initially with an RLL technique and subsequently with a PSLL technique, which can be either SFLT or DFLT. The attacker – equipped with reverse engineering capabilities to access the gate-level locked netlist mapped to a commercial library, the functional IC, and EDA tools – then utilizes RESAA under either the OG or OL scenarios.

We detail our practical classification analysis and the partitioning of the CLL circuit into two distinct netlists facilitated by CG identification. Each netlist includes all the necessary inputs related to a particular LL technique. Our method involves implementing the RESAA attack on these divided netlists, effectively revealing the secret key.

A. Classification and Partition

Fig. 4 shows our pre-processing method in the left portion of the RESAA framework, which converts original circuits into locked CLL circuits (in bench format) and then mapping them into gate-level netlists (in Verilog) using a commercial logic synthesis tool. This step is crucial to make the CLL circuits suitable for analysis during the CG identification and key classification phase. The steps drawn in orange in Fig. 4 are, conceptually, executed by a defender. They are included here for the sake of completeness. Then, RESAA performs (re)synthesis on the locked design. When doing this, RESAA utilizes a commercial synthesis tool. However, RESAA introduces one additional constraint: only 2-input gates are allowed during mapping, thus simplifying the work of RESAA as CGs become easier to identify. Under this restriction, there is always a critical gate where all keys from one technique converge on one input while the other input links to keys from the other technique.

As seen in Fig. 4, the CG acts as a common path for all key inputs derived from the RLL and PSLL techniques. By utilizing the internal graph representation provided by the EDA tool, the process was scripted to analyze each key input and group them based on the primary output (PO) they reach. We identify the CG as the first gate where all paths from the key inputs converge before reaching a PO. This gate is crucial for classifying the LL techniques. One input of the CG is solely associated with an RLL key, while the other is associated with PSLL keys.

Our study has identified three distinct behaviors of the key inputs leading to a PO. The first group, which is associated with the use of RLL alone, shows no discernible patterns in the number of reachable outputs. This is predictable, since RLL is supposed to be random. The second group, related to the use of PSLL, is recognized when all key inputs lead to the same number of POs. The third group, a CLL circuit utilizing both RLL and PSLL techniques, emerges when both behaviors are present within a design.

The relationship between key inputs and POs reached by them is a feature that is leveraged by RESAA to perform netlist partition. Fig. 5 presents our classification methodology, where RESAA initially partitions the design into two distinct netlists: one containing exclusively RLL key inputs and the other consisting of PSLL key inputs. The PSLL netlist is further categorized into SFLT and DFLT techniques. Within the SFLT category, further classification is performed to distinguish between Anti-SAT-like techniques with a complementary function and SARLock-like techniques without a complementary

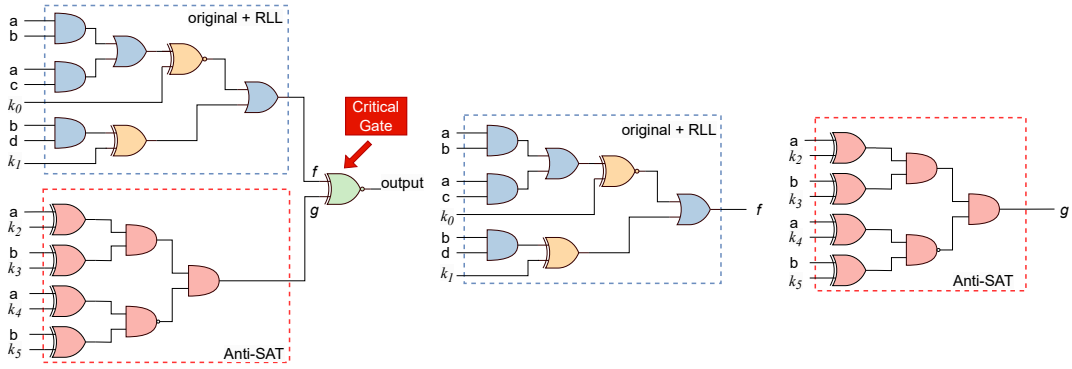


Fig. 6: A circuit locked with RLL and Anti-SAT. The CG identification and partition process by RESAA are highlighted.

function. It is important to ensure that the classification of key inputs is accurate to apply a successful attack. The following high-level steps outline the classification process:

- **Step 1: Generate Key Input Graphs**
  Create a directed graph for each key input k_i that shows all paths from k_i to the POs.
- **Step 2: Identify Number of POs for Each Key**
  Determine how many POs each key input k_i can reach through the paths.
- **Step 3: Group Key Inputs**
  Group key inputs together if they have the same number of reachable POs, suggesting they belong to the same LL technique.
- **Step 4: Identify the CG**
  Find the first gate where all paths from the grouped key inputs meet before reaching a PO. This gate is the CG.
- **Step 5: Final Classification**
  Check the classification by analyzing the paths through the CG. If a key is misclassified, move to the next gate and try reclassifying. For instance, if an RLL key was wrongly classified as a PSLL key because its paths appeared in both inputs of the CG, the process moves to the next gate. If no key paths match the new gate, the key is correctly reclassified as RLL.

RESAA takes the advantage of the fact that a logic synthesis tool already has, internally, a graph representation of the circuit and its connections that is efficient and can be queried at will. Therefore, Step 1, as previously outlined, requires no effort. With additional scripting, all other steps can be executed within the environment of the synthesis tool itself. This is a key feature of RESAA and contributes to its scalability.

To summarize the inner workings of RESAA, Fig. 6 presents an example of a circuit locked by RLL + Anti-SAT. In this example, the CG consolidates all paths from RLL key inputs into one input of the XNOR gate, while all paths from Anti-SAT key inputs are connected to the other input. Following this, RESAA removes this CG, resulting in the generation of two netlists. A first netlist is created by eliminating the CG and directly connecting the RLL partition to the design's PO. Similarly, the second netlist undergoes this process by taking the other input of the CG and connecting it directly to the PO, thereby creating a second netlist containing only the locking/restore unit.

In a CLL-locked design, the identification of this CG serves as an anchor for splitting the netlist and applying further attacks. In other words, the presence of a CG that consolidates all the various paths from RLL and from the PSLL is a vulnerability. Identifying this critical node yields valuable insights into the structural organization of the locked netlist, enabling the systematic division of the design into two distinct netlists. By identifying the CG, partitioning the netlist into two, and employing appropriate attack(s), an adversary can more easily uncover the secret key with respect to dealing with the whole CLL-locked circuit.

As shown in the right portion of Fig. 4, with the keys now accurately classified into RLL and PSLL categories and the CLL netlist partitioned into two distinct segments, RESAA can proceed to apply specific attacks. In scenarios, where the PSLL netlist includes an SFLT, the tool divides the CLL netlist, separating the original netlist with RLL keys and a logic unit from the PSLL technique, as illustrated in Fig. 2(b). Conversely, if the PSLL consists of a DFLT, RESAA separates the stripped function along with the RLL portion, where the stripped function includes an original circuit along with a perturbation module, as shown in Fig. 2(c).

B. Attack Under the OG Threat Model

Fig. 7 shows how attacks under the OG threat model can be applied. After classification/partition, we have the netlist consisting of RLL + original/stripped and locking/restore unit. Three attacks are used for this flow: the quantified boolean formula (QBF)-based attack [22] is applied to the locking/restore unit, the SAT-based attack [15] or the query attack [33] is applied to the RLL + original/stripped netlist, where an oracle is used to obtain input/output patterns.

We note that quantified Boolean formula (QBF) is the generalization of the SAT problem, in which both existential ($\exists$) and universal ($\forall$) quantifiers can be applied to each variable.

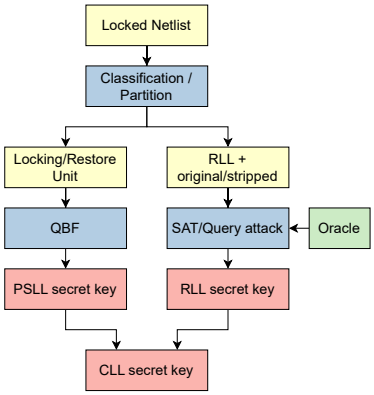


Fig. 7: RESAA attacks under the OG threat model: Netlists are highlighted in yellow, attacks are shown in blue, the oracle is indicated in green, and the resulting secret keys are in red.

Thus, the QBF attack, which targets the locking/restore unit, operates in two main steps. First, it constructs a QBF problem by combining the conjunctive normal form (CNF) formulas of individual gates. Secondly, it generates two distinct QBF problems: one for when the output of the locking/restore unit is equal to 0 and another for when this output is 1 for **all** possible input combinations. Subsequently, a QBF solver is employed to find the secret key on these problems. If a solution exists for either of these QBF problems, it signifies that the secret key of the partitioned circuit is found. It is important to highlight that formulating two distinct QBF problems allows for the evaluation of all possible output values. We note that the QBF solver can easily find a solution on the locking unit of an SFLT since it is designed to have a constant value under the secret key for all input combinations [22]. Thus, when the QBF attack identifies a solution, it confirms the presence of an SFLT within the CLL. In this case, a SAT-based attack is applied to the netlist containing the RLL + original configuration. However, the query attack becomes necessary if the netlist is composed of RLL + stripped. The stripped functionality allows many incorrect keys to produce correct outputs for many inputs, significantly hindering the SAT solver's ability to converge on the proper key. As a result, RESAA can identify the PSLL secret key using the QBF attack and the RLL secret key using the SAT-based attack, thus revealing the entire CLL secret key. In cases, where the CLL is composed of RLL + DFLT, QBF alone cannot identify the PSLL secret key, but the query attack can identify the majority of RLL keys.

A combined strategy of SAT-based and query-based attacks was implemented to enhance the effectiveness of RLL key extraction. The process begins by converting the locked netlist, consisting of the RLL + original/stripped configuration, into CNF to facilitate SAT solver analysis. The SAT solver then identifies DIPs that differentiate potential key values. These DIPs are applied to a functional IC, serving as an oracle that provides the correct output for each input. This correct output is then used to refine the CNF formula, progressively eliminating incorrect key candidates. The SAT solver iteratively performs this process until no further DIPs can be found, ensuring that only the correct key values remain.

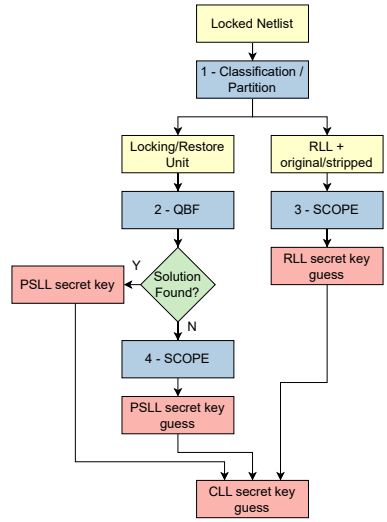


Fig. 8: RESAA steps under the OL threat model: Netlists are highlighted in yellow, attacks are shown in blue, and the resulting secret keys are in red.

Subsequently, the flow transitions to a query-based attack phase in cases where no PSLL secret key was found, which uses QBF attack, where specific, strategically crafted queries are directed at the oracle to extract additional insights about the correct key. Each query is analyzed to deduce logical inferences, further refining the CNF formula and pruning the search space. Our combined approach systematically narrows down the possible key values with increased efficiency and precision by targeted queries. Integrating these two attack methodologies ensures a comprehensive and robust evaluation of the (C)LL scheme, ultimately enhancing the likelihood of successful key recovery.

C. Attack Under the OL Threat Model

Figure 8 shows our attack under the OL threat model, which shares its initial steps with the OG threat model and begins with the same CLL netlist as an input. This is followed by a partitioning and classification process that can be adjusted to suit the specific needs of the task. The netlist, now classified and partitioned in a manner identical to the OG model, is subjected to the QBF attack on the locking/restore unit. If this attack is successful, our tool returns the PSLL key, reaffirming the previous classification as SFLT. In the event of an unsuccessful QBF attack, then the OL SCOPE attack is run on this netlist.

As can be observed from Fig. 8, the SCOPE attack is applied to the locking/restore unit to guess the PSLL secret key if the QBF attack fails to determine the PSLL secret key and to the partition containing both RLL + original/stripped and locking/restore unit to guess the RLL secret key. It is important to note that the SCOPE solution may yield a logical value of 0, 1, or an undetermined value for a given key input.

To bolster our methodology's reliability and ensure the developed tool's correctness, we introduced an additional step involving a logic equivalence checking (LEC) tool, specifically for internal verification. Therefore, this step is not shown in the graphical representation of the attack flow given in Figures 7 and 8. This integration occurred at two crucial junctures within our flow. First, following partitioning, we employed the LEC tool to compare the RLL with the original design, particularly in scenarios involving SFLT and the preceding netlist locked solely with the RLL we generated. Secondly, after uncovering the secret key for CLL, we utilized the LEC tool to ensure the integrity of our process. This involved comparing the modified CLL netlist, which includes the recovered secret key, against the original netlist. The LEC tool checks for logical equivalence between these two netlists, verifying that the CLL netlist with the secret key produces the same outputs as the original netlist for all possible inputs. We solely utilized this step for verification purposes. An adversary **does not** possess this capability since, by definition, the adversary does not possess the original circuit.

IV. Results

Our methodology utilizes a combination of Perl, Python, and TCL scripts to interface with the commercial logic synthesis tool Cadence Genus [43]. Synthesis is conducted using a commercial 65 nm standard cell library. More details about the logic synthesis process and settings can be found in our open-sourced scripts. All experiments were performed on a 32-core Intel Xeon processor running at 3.60 GHz with a RAM capacity of 1 TB. Yet, no attacks are multithreaded, and the results presented here can be generalized to a personal computer.

A. Experiments on Small-and Moderate-Size Circuits

Our first study involved the analysis of CLL circuits utilizing a set of ten benchmarks: *c2670*, *c3540*, *c5315*, *c6288*, and *c7552* from the ISCAS'85 benchmark suite [44], along with *b14_C*, *b15_C*, *b20_C*, *b21_C*, and *b22_C* from the ITC'99 benchmark suite [45], which are the combinational logic parts of the related sequential circuits. Table II presents a comprehensive description of the benchmarks, including the number of primary inputs (#in) and outputs (#out) alongside pertinent metrics such as area (μm^2), power consumption (mW), and delay (ps) for each benchmark. It also presents the area, power, and delay values after the benchmarks were locked by the RLL technique using the Neos tool [46]. The methodology for determining the number of RLL key inputs (p) was guided by considering both the number of inputs and the overhead associated with the LL technique. In this experiment, we used the same number of key inputs in the RLL and PSLL techniques. For the ISCAS'85 circuits, p was chosen as 32 or 64 since the number of key inputs in a PSLL technique is limited to the number of primary inputs of the original design. It was chosen as 128 for the ITC'99 circuits since they are generally larger than ISCAS'85 circuits.

TABLE II: Details of ISCAS'85 and ITC'99 circuits.

Circuit	Original Netlist					p	RLL Locked Netlist		
	#in	#out	area	power	delay		area	power	delay
c2670	157	64	1046	3.36	1264	64	1424	5.22	1742
c3540	50	22	1518	6.58	1977	32	1655	7.48	2091
c5315	178	123	2460	9.9	1864	64	2864	1.21	1982
c6288	32	32	3133	8.48	4621	32	3303	9.09	5160
c7552	206	105	2702	1.32	1663	64	3209	1.61	2015
b14_C	275	245	8326	3.59	4882	128	8872	4.34	4864
b15_C	485	449	12416	3.11	4809	128	12938	3.78	5188
b20_C	522	512	17210	8.95	5536	128	17731	9.93	5511
b21_C	522	512	17685	9.24	5075	128	18248	9.94	5061
b22_C	767	757	26416	1.33	5358	128	26941	1.39	5282

Analysis of the data presented in Table II reveals a consistent trend across all benchmarks subjected to an RLL scheme. There is a modest increase in area and power consumption, averaging approximately 4.5%, alongside an average delay increment of about 10.5%. This trend aligns with the characteristics of the RLL technique (insertion of XOR/XNOR gates), which minimally affects area and power consumption. Nevertheless, incorporating additional logic gates along the critical path contributes to a slight rise in delay.

After the RLL locking phase, the second locking technique was introduced by utilizing one out of five distinct PSLL techniques, namely Anti-SAT, Anti-SAT-DTL, CASLock, and SARLock as SFLTs and TTLock as a DFLT. The implementation of Anti-SAT, Anti-SAT-DTL, and TTLock was facilitated using the Neos tool [46]. Meanwhile, SARLock and CASLock were implemented using a Python and Perl script developed by P. Subramanyan and L. Aksoy, respectively.

Table III presents the total number of key inputs (k) alongside associated metrics, including area, power, and delay across all considered CLL benchmarks. The area exhibited an average increase of 4.50% compared to the original version. An average increase of 10% in the CLL of SFLT and around 11.6% when composed with DFLT. Notably, power consumption remains relatively stable compared to benchmarks locked with only RLL. Moreover, the observed delay overheads are approximately 5% for SFLT and around 8% for RLL + TTLock configurations when compared to the netlist locked with only RLL.

Initially, we performed several attacks documented in the existing literature to compare against our developed methodology, expecting all CLL circuits to withstand SAT-based attacks and their variants. All benchmarks locked in the CLL scheme were submitted to four attacks: the SAT-based attack (sat) [15], AppSAT (appsat) [30], Double-DIP (dp) [32], and query attack (qatt) [33]. Table IV shows the runtime to find a solution, with "out-of-time" (OoT) indicating instances where no solution could be found within the allowed 48-hour time limit. As observed from Table IV, the SAT-based and Double-DIP attacks

TABLE III: Details of locked ISCAS'85 and ITC'99 circuits.

Circuit	k	CLL Locked Netlist														
		RLL+Anti-SAT			RLL+Anti-SAT-DTL			RLL+CASLock			RLL+SARLock			RLL+TTLock		
		area	power	delay	area	power	delay	area	power	delay	area	power	delay	area	power	delay
c2670	128	1792	0.63	1727	1789	0.64	1836	1793	0.65	1754	1842	0.65	1842	1748	0.64	1788
c3540	64	1813	0.80	2083	1826	0.81	2052	1846	0.83	2042	1886	0.82	2087	1840	0.81	2096
c5315	128	3226	1.33	1954	3213	1.32	2070	3244	1.35	1980	3280	1.34	2081	3198	1.32	2005
c6288	64	3471	9.15	5233	3446	9.13	5119	3477	9.15	5167	3524	9.16	5250	3508	9.23	5106
c7552	128	3861	1.84	2109	3577	1.73	2011	3378	1.68	2004	3623	1.75	1993	3542	1.77	1972
b14_C	256	9557	4.58	4600	9526	4.56	4640	9637	4.62	4862	9820	4.63	4893	10052	4.65	4949
b15_C	256	13608	4.00	5148	13591	4.00	5144	13695	4.04	5020	13897	4.07	5036	14476	4.43	5067
b20_C	256	18379	10.14	5497	18379	10.12	5470	18468	10.20	5609	18641	10.22	5488	20023	10.91	5728
b21_C	256	18904	10.17	4922	18893	10.17	4954	18968	10.21	5030	19167	10.23	5045	20069	10.98	5127
b22_C	256	27559	14.02	5376	27578	14.03	5367	27703	14.19	5317	27803	14.10	5290	30007	14.51	5421

TABLE IV: Details of OG attacks on locked ISCAS'85 and ITC'99 circuits.

Circuit	Locked Netlist																					
	RLL+Anti-SAT					RLL+Anti-SAT-DTL					RLL+CASLock					RLL+SARLock					RLL+TTLock				
	sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt		sat	appsat	dp	qatt	
	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time
c2670	OoT	998	OoT	52	48	OoT	170	OoT	50	45	OoT	238	OoT	50	25.4	OoT	114	OoT	55	59	OoT	1055	OoT	48	48
c3540	OoT	766	OoT	30	20	OoT	203	OoT	32	19	OoT	66	OoT	30	8	OoT	91	5	31	22	48656	4499	2	31	19
c5315	OoT	239	OoT	62	50	OoT	9949	OoT	62	52	OoT	131	OoT	62	42	OoT	64	OoT	62	72	OoT	2768	OoT	62	60
c6288	OoT	OoT	OoT	32	77	OoT	58470	OoT	32	69	OoT	3936	OoT	32	90	OoT	935	OoT	32	119	OoT	6270	OoT	32	93
c7552	OoT	172	OoT	55	96	OoT	543	OoT	55	105	OoT	279	OoT	55	59	OoT	247	OoT	55	108	OoT	51	OoT	55	79
b14_C	OoT	OoT	OoT	109	1822	OoT	OoT	OoT	112	1383	OoT	OoT	OoT	106	2082	OoT	OoT	OoT	109	1335	OoT	OoT	OoT	111	1783
b15_C	OoT	OoT	OoT	98	466	OoT	OoT	OoT	97	635	OoT	OoT	OoT	96	836	OoT	OoT	OoT	80	513	OoT	OoT	OoT	87	671
b20_C	OoT	OoT	OoT	115	1562	OoT	OoT	OoT	115	1647	OoT	OoT	OoT	109	2645	OoT	OoT	OoT	115	1645	OoT	OoT	OoT	113	2187
b21_C	OoT	OoT	OoT	113	1119	OoT	OoT	OoT	114	1337	OoT	OoT	OoT	113	1935	OoT	OoT	OoT	110	1385	OoT	OoT	OoT	109	1738
b22_C	OoT	OoT	OoT	113	1034	OoT	OoT	OoT	113	1214	OoT	OoT	OoT	108	1489	OoT	OoT	OoT	105	938	OoT	OoT	OoT	111	1342

exhibit low efficiency in deciphering key inputs, as expected. The SAT-based attack only found a solution for one single RLL + TTLock case in the *c5315* circuit. The AppSAT attack showed promising results for small circuits but demanded significant execution time compared to other attacks. While the approach demonstrated near 100% efficiency for ISCAS'85 circuits, it failed to break all the locked ITC'99 circuits. This limitation arises due to the increase in hardware complexity of locked circuits, making it computationally prohibitive to determine the secret key. Finally, the query attack [33] displayed varying execution times and degrees of success in deciphering key inputs, as shown in Table IV. The number of proven key inputs prv discovered ranged just over 41%, with execution times spanning from 8 to 2645 seconds.

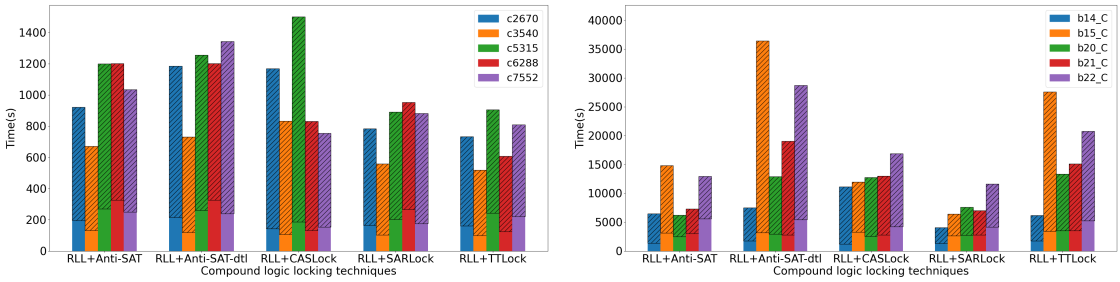
It is important to note that these attacks are available from the literature, and when designed, initially targeted circuits that are locked with a specific LL technique. Interestingly, according to Table IV, these attacks did not obtain significant results in uncovering key inputs in experiments carried out with CLL circuits. The AppSAT attack, despite its potential to find solutions quickly, presents significant challenges. In other words, available attacks often prove highly inaccurate and ultimately ineffective, mainly when dealing with intricate CLL structures. In this context, we emphasize that a tool like RESAA is invaluable. Next, we present the results of our OG and OL attack strategies.

1) Results of RESAA under the OG Threat Model: Under the OG threat model, two well-known attacks were considered. Specifically, the QBF attack outlined in [22] was employed to decipher PSLL keys, along with the SAT-based attack [15] and the query attack [22] to manage RLL key inputs.

Fig. 9 presents the classification and execution times. In this context, "classification time" refers to the duration required for categorizing the LL technique utilized in the CLL design, depicted in the lower section of the graph. Conversely, "attack time" is indicated by the hatched portion of the graph, while 'execution time' represents the total time, including both classification/partition time and the subsequent attack on each circuit, depicted by both sections in the graph. We note that our methodology could find the critical gates in both CLL techniques (i.e., RLL+SFLT and RLL+DFLT) with 100% accuracy.

Upon observation, it is evident that when a CLL design is classified as RLL + SFLT, a solution emerges during the QBF attack of the locking/restore unit. Subsequently, the netlist consisting of RLL + original becomes vulnerable to the SAT-based attack. RESAA successfully deciphered all key inputs for CLL circuits with SFLT. In cases where no solution is found by the QBF attack, the second netlist is mandatorily classified as RLL + stripped. In this scenario, a query attack is applied to the RLL + stripped netlist, resulting in proven RLL key inputs.

In each case, the complete set of key inputs was successfully exposed by implementing the partitioning approach, achieving 100% discovery when CLL included both RLL and SFLT. However, when a DFLT was introduced as a second technique, initial attempts to uncover PSLL key inputs were unsuccessful. It was only after employing a query attack that some of the RLL keys were eventually disclosed. A time limit of 1-hour was set for this query attack, as further execution did not yield improved results despite prolonging the runtime. We do hypothesize that this could be improved by changing the query



(a) ISCAS'85 benchmark

(b) ITC'99 benchmark

Fig. 9: Classification and execution times (seconds) for attacking ISCAS'85 and ITC'99 benchmarks in the CLL scheme. Bottom: Classification and partition time. Hatched: Attack time. Combined: Total execution time.

strategy.

Our classification and partitioning step involves processing a CLL netlist as input, where a timing analysis[5] is conducted to distinguish between RLL and PSLL key inputs, as illustrated in Fig. 7. The size and complexity of the CLL design directly influence the duration of both the classification and execution phases. For instance, the maximum execution time for ISCAS'85 benchmarks was approximately *1400* seconds, whereas it reached around *36000* seconds for the larger ITC'99 benchmarks. The classification time typically accounts for less than *1/3* of the total execution time.

The validation process, using the LEC tool, was crucial in cases where all the keys were revealed to validate RESAA. In each instance, the netlist composed of the RLL + original portion was equivalent to the netlist previously locked with RLL. Additionally, the CLL netlist, with the added secret key inputs, was confirmed to coincide in functionality with the original design, thereby certifying the accuracy of RESAA in the partitioning processes. This result confirms a high level of confidence in RESAA's outcomes.

2) Results of RESAA under the OL Threat Model: The attack strategy under the OL threat model is more restrictive than the OG model because it does not use an oracle. The SCOPE attack, described in [47], and the QBF method from [22] were employed. Specifically, in the OL threat model, the SCOPE attack was exclusively used to estimate key inputs for both RLL and PSLL. This approach was necessary when finding a solution in the netlist composed of the locking/restore unit was not feasible, as illustrated in Fig. 8.

Table V presents the results of the SCOPE attack conducted on both the entire CLL design and the partitioned netlist generated by RESAA. In this table, *cdk* and *dk* represent the number of correctly deciphered key inputs and the total number of deciphered key inputs, respectively while *time* indicates the runtime of the attack.

Observe from Table V that the SCOPE attack does not succeed in breaking the CLL designs locked by Anti-SAT, Anti-SAT-DTL, and CASLock. However, using RESAA, many key inputs associated with the RLL + original netlists, and even the entire RLL key, can be uncovered. That is exemplified by the *c3540* circuit locked by Anti-SAT-DTL. It is worth noting that the SCOPE attack can reveal over 48% of key bits using the partition netlists locked with any LL technique. The SCOPE attack on RLL + SARLock circuits resulted in the *dk*/*k* ratio of 50%, where *k* denotes the total number of key inputs, with an average *cdk*/*dk* ratio of 56%. In contrast, RESAA demonstrated a higher *dk*/*k* ratio, with cases such as *c5315* achieving a *cdk*/*k* ratio of 78%. Furthermore, when a circuit is locked using DFLT, as in RLL + TTLock, the SCOPE attack showed a higher *dk*/*k* ratio, but the *cdk*/*dk* ratio remained lower compared to that achieved by RESAA. In other words, SCOPE alone makes more incorrect guesses in the entire locked design than in the partitions obtained by RESAA. We note that the success of SCOPE in partitions obtained by RESAA is also related to the resynthesis of the locked design before the attack, which was also observed in [42], since it changes the structure of the locked netlist.

The runtime of the SCOPE attack, whether executed on the entire design or the partitioned design, significantly depends on the number of gates and keys present in the locked design. Consequently, the runtime for the netlist generated by RESAA after the partition step is notably shorter, as it contains only a fraction of the key inputs and gates. This reduction in complexity results in a smaller runtime, making the attack more efficient. Moreover, the runtime is typically smaller for the partitioned netlist due to the reduced number of gates, as it represents only a portion of the entire design.

The RESAA results for the ITC'99 *b22_C* circuit locked with RLL + CASLock stand out, achieving 190 *cdk* out of 205 *dk*, a 92.6% accuracy. These results are very encouraging, given the size and complexity of the circuit paired with the restrictive OL setting. Success in this case demonstrates RESAA's ability to analyze and navigate complex CLL designs effectively. This highlights RESAA's superior performance in accurately recovering keys, even in more complex CLL schemes.

[5]Timing analysis here is the static timing analysis (STA) performed by the logic synthesis tool. By performing STA, inherently a graph is built that can be used to query whether an input i has a path to an output o.

TABLE V: Results of OL Attacks on the locked ISCAS'85 and ITC'99 circuits.

Circuit	RLL+Anti-SAT				RLL+Anti-SAT-DTL				RLL+CASLock				RLL+SARLock				RLL+TTLock			
	SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA	
	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time
c2670	0/0	14	73/104	4	0/0	13	80/104	4	0/0	9	84/104	4	32/64	8	97/105	4	16/106	10	68/105	9
c3540	0/0	6	37/43	2	0/0	6	40/64	2	0/0	4	40/45	2	15/32	4	41/43	2	8/42	7	32/45	4
c5315	0/0	15	77/107	5	0/0	16	78/107	5	0/0	12	81/107	5	32/64	10	100/107	5	30/107	12	69/109	11
c6288	0/0	7	41/56	3	0/0	7	42/56	2	0/0	5	42/56	3	17/32	5	49/56	3	4/53	6	40/56	5
c7552	0/0	17	78/108	5	0/0	17	79/105	5	0/0	8	81/108	4	40/64	11	98/107	6	28/107	13	82/109	12
b14_C	0/0	91	160/210	67	0/0	91	168/215	69	0/0	67	180/213	44	68/128	71	168/200	46	36/203	87	38/72	58
b15_C	0/0	117	166/210	87	0/0	120	190/214	90	0/0	88	180/210	58	72/128	89	186/214	59	49/202	109	58/82	72
b20_C	0/0	155	172/203	115	0/0	156	182/200	117	0/0	116	191/211	77	67/128	118	188/209	77	44/201	136	62/86	93
b21_C	0/0	159	183/210	119	0/0	164	185/212	122	0/0	119	173/213	78	77/128	121	178/200	82	52/196	142	60/70	97
b22_C	0/0	231	180/209	172	0/0	233	185/212	175	0/0	177	190/205	116	86/128	175	189/199	116	48/192	193	56/70	139

TABLE VI: Details of large-size ITC'99 circuits.

Circuit	Original Netlist					p	RLL Locked Netlist		
	#in	#out	area	power	delay		area	power	delay
b17_C	1452	1446	40033	10.4	6261	256	40734	10.9	6341
b18_C	3357	3343	115197	60.0	7830	256	114014	57.7	7414
b19_C	6666	6670	230786	131.0	7957	256	227491	128.0	7490

B. Experiments on Large-Size Circuits

To demonstrate the scalability of our RESAA framework, we conducted experiments on three large-size circuits from the ITC'99 benchmark suite [45], namely *b17_C*, *b18_C*, and *b19_C*. We used a larger number of key inputs than those used in small- and medium-sized circuits given in Section IV-A. In this case, the number of RLL and PSLL key inputs is 256 with a total number of key inputs set to 512.

Table VI presents the details of large-size circuits on the number of primary inputs and outputs and also, gate-level synthesis results. It also shows the gate-level synthesis results of these circuits when locked by RLL using 256 key inputs. Observe that the hardware complexity of the locked designs changed slightly with respect to the original design, leading to 1.8% increase in the area of the *b17_C* and 1.0% and 1.4% reduction in the area of the *b18_C* and *b19_C*, respectively. These numbers are within the marging of noise of logic synthesis tools.

Table VII presents the gate-level synthesis results of circuits locked by RLL when they are subsequently locked by a PSLL technique. Observe that while the use of SFLT techniques increases the hardware complexity slightly, the use of the DFLT technique leads to locked designs with slightly less hardware complexity with respect to those of circuits locked by the RLL technique.

Table VIII presents the result of conventional attacks, i.e., SAT-based, AppSAT, Double DIP, and query attack, on the circuits locked by CLL techniques. The time limit for these attacks was set to 5 days due to the large-size locked circuits and large number of key inputs. Observe that none of these attacks could recover the secret key within the time limit, highlighting the inherent robustness of CLL schemes against these attacks. Only the query attack demonstrated limited success. The minimum (maximum) number of key inputs' values found by the query attack was 169 (201) obtained on the *b17_C* (*b18_C*) circuit when locked by RLL+TTLock (RLL + Anti-SAT). Also, observe from Tables IV and VIII that as the hardware complexity of the locked design increases, the SAT-based, AppSAT, and Double DIP attacks cannot find the secret key and the run-time of the query attack increases significantly. These results underline the strong security properties of large-size circuits locked by CLL techniques against these attacks.

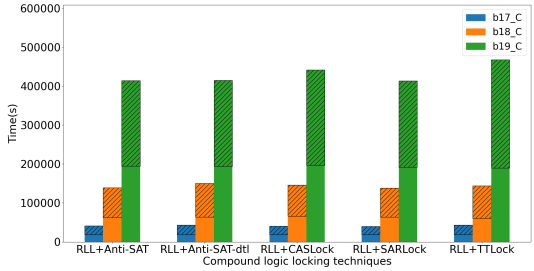


Fig. 10: Classification and execution times (seconds) for attacking the large-size ITC'99 circuits in the CLL scheme. Bottom: Classification and partition time. Hatched: Attack time. Combined: Total execution time.

Figure 10 presents the classification and execution times of RESAA on the large-size circuits locked by CLL techniques under the OG threat model. Observe that RESAA can complete its tasks in the given time limit, i.e., 5 days, demonstrating its effectiveness in handling large-size circuits while classifying the locking techniques and recovering the secret key, with respect to the conventional attacks as shown in Table VIII. Similar to the small- and medium-size circuits locked by CLL techniques, RESAA discovered all the secret keys of designs locked with the RLL + SFLT technique. However, as can be observed from Figs. 9 and 10, the run-time of RESAA increases significantly as the hardware complexity of the locked design and the number of key inputs increase.

Table IX presents the results of SCOPE and RESAA under the OL threat model. Observe that SCOPE alone was ineffective in discovering the key inputs' values in the circuits locked by RLL + Anti-SAT, RLL + Anti-SAT-DTL, and RLL + CASLock. However, it managed to uncover values of some key inputs in circuits locked by RLL + SARLock and RLL + TTLock. In contrast, RESAA could decipher a large number of key inputs' values, such as 417 in the *b17_C* circuit, 422 in the *b18_C* circuit, and 434 in the *b19_C* circuit when they are locked by RLL + TTLock. Note also that while the maximum

TABLE VII: Details of large-size locked ITC'99 circuits.

Circuit	k	CLL Locked Netlist											
		RLL+Anti-SAT			RLL+Anti-SAT-DTL			RLL+CASLock			RLL+SARLock			RLL+TTLock		
		area	power	delay	area	power	delay	area	power	delay	area	power	delay	area	power	delay
b17_C	512	42086	11.2	6576	42280	11.3	6447	42198	11.4	6767	42379	11.4	6278	40979	10.9	6225
b18_C	512	115364	57.8	7442	115512	57.8	7409	115485	58.2	7649	115641	58.3	7628	105462	58.9	7328
b19_C	512	228943	128.0	7392	229004	129.0	7724	228951	129.0	7558	229097	129.0	7641	206309	127.0	7150

TABLE VIII: Details of OG attacks on large-size locked ITC'99 circuits.

Circuit	Locked Netlist																			
	RLL+Anti-SAT					RLL+Anti-SAT-DTL					RLL+CASLock					RLL+SARLock					RLL+TTLock				
	sat	appsat	dp		qatt	sat	appsat	dp		qatt	sat	appsat	dp		qatt	sat	appsat	dp		qatt	sat	appsat	dp		qatt
	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time	time	time	time	prv	time
b17_C	OoT	OoT	OoT	193	4449	OoT	OoT	OoT	193	5002	OoT	OoT	OoT	187	7531	OoT	OoT	OoT	184	3058	OoT	OoT	OoT	169	2824
b18_C	OoT	OoT	OoT	201	20174	OoT	OoT	OoT	192	20608	OoT	OoT	OoT	190	13218	OoT	OoT	OoT	182	13586	OoT	OoT	OoT	178	9946
b19_C	OoT	OoT	OoT	187	65014	OoT	OoT	OoT	188	63350	OoT	OoT	OoT	192	43588	OoT	OoT	OoT	192	42733	OoT	OoT	OoT	190	33007

TABLE IX: Results of OL Attacks on the large-size locked ITC'99 circuits.

Circuit	RLL+Anti-SAT				RLL+Anti-SAT-DTL				RLL+CASLock				RLL+SARLock				RLL+TTLock			
	SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA		SCOPE		RESAA	
	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time
b17_C	0/0	356	302/416	188	0/0	345	301/416	188	0/0	353	280/406	171	106/256	368	285/406	173	69/250	278	312/417	167
b18_C	0/0	1156	370/417	545	0/0	780	360/416	577	0/0	1093	330/403	547	126/256	1201	319/403	842	80/286	795	315/422	497
b19_C	0/0	2225	377/430	1045	0/0	2256	372/428	1062	0/0	2114	359/424	1105	140/256	2149	350/424	1011	76/272	1005	376/434	894

cdk/k ratio in the SCOPE attack is around 27%, this value in RESAA reaches up to 73%, where k is the total number of key inputs.

These results demonstrate that the RESAA framework scales effectively to handle large-size circuits. By efficiently partitioning designs and leveraging advanced attack strategies, it improves the accuracy of key recovery and maintains practical execution times, making it a powerful tool for the evaluation of large-size circuits locked by CLL techniques.

V. Conclusions

The semiconductor industry's shift towards a fabless model has necessitated advanced security measures to combat emerging threats such as piracy and counterfeiting. CLL, which integrates multiple LL techniques, has been proposed by researchers as a robust solution to these security challenges. However, the security of CLL itself has not been extensively analyzed until now.

Our RESAA framework addresses this gap by systematically classifying locked designs, identifying critical gates, and executing attacks to uncover secret keys. Unlike previous methods, RESAA is agnostic to specific LL techniques, making it a versatile tool for evaluating a wide range of CLL implementations. Through our detailed methodology, which includes classification, partitioning, and applying both OG and OL attack strategies, we demonstrated the framework's ability to expose vulnerabilities in CLL-protected circuits.

Experimental results using ISCAS'85 and ITC'99 benchmark suites highlighted the efficacy of RESAA. The framework successfully identified critical points within the CLL designs, enabling attacks that revealed the secret keys. Our findings underscore the necessity of careful evaluation and selection of LL techniques to ensure the security of IC. The results indicated that even advanced CLL strategies are susceptible to the RESAA framework's targeted attacks.

References

[1] M. J. Shaw and M. A. Schilling, "The evolution of the semiconductor industry: A focus on fabless firms," *California Management Review*, 2000.

[2] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[3] G. Zarrinchian, "A chip activation protocol for preventing ic recycling," *Microprocessors and Microsystems*, vol. 101, p. 104872, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933123001187

[4] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[5] T. M. Supon, M. Seyedbarhagh, R. Rashidzadeh, and R. Muscedere, "A method to prevent hardware trojans limiting access to layout resources," *Microelectronics Reliability*, vol. 124, p. 114212, 2021.

[6] T. D. Perez and S. Pagliarini, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184 013–184 035, 2020.

[7] F. Koushanfar and G. Qu, "Hardware metering," in *38th Annual Design Automation Conference (DAC)*. New York, NY, USA: Association for Computing Machinery, 2001, p. 490–493.

[8] G. Qu and L. Yuan, *Secure Hardware IPs by Digital Watermark*. New York, NY: Springer New York, 2012, pp. 123–141.

[9] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2008, pp. 1069–1074.

[10] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[11] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021, pp. 114–119.

[12] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *J. Electron. Test.*, vol. 35, no. 3, pp. 273–291, 2019.

[13] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Threats on Logic Locking: A Decade Later," in *Great Lakes Symposium on VLSI (GLVLSI)*, 2019, pp. 471–476.

[14] H. M. Kamali, K. Z. Azar, F. Farahmandi, and M. Tehranipoor, "Advances in logic locking: Past, present, and prospects," *Cryptology ePrint Archive*, 2022.

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

[16] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[17] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 2, pp. 199–207, 2019.

[18] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2020, no. 1, pp. 175–202, 2019.

[19] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "TTLock: Tenacious and traceless logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 166–166.

[20] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-Based Perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 12, pp. 4439–4452, 2020.

[21] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 8, no. 2, pp. 517–532, 2020.

[22] L. Aksoy, M. Yasin, and S. Pagliarini, "Kratt: Qbf-assisted removal and structural analysis attack against logic locking," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[23] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 85–90.

[24] D. Divyanshu, R. Kumar, D. Khan, S. Amara, and Y. Massoud, "Logic locking using emerging 2t/3t magnetic tunnel junctions for hardware security," *IEEE Access*, vol. 10, pp. 102 386–102 395, 2022.

[25] 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 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 405–410.

[26] Z. U. Abideen, S. Gokulanathan, M. J. Aljafar, and S. Pagliarini, "An overview of fpga-inspired obfuscation techniques," *ACM Comput. Surv.*, jul 2024.

[27] M. John, A. Hoda, R. Chouksey, and C. Karfa, "Sat based partial attack on compound logic locking," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2020, pp. 1–6.

[28] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-sat: Fault-aided sat-based attack on compound logic locking techniques," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1166–1171.

[29] J. Mellor, A. Shelton, M. Yue, and F. Tehranipoor, "Attacks on logic locking obfuscation techniques," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2021, pp. 1–6.

[30] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[31] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE Transactions on Very Large Scale Integration VLSI Systems (TVLSI)*, vol. 29, no. 8, pp. 1529–1542, 2021.

[32] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Great Lakes Symposium on VLSI (GLSVLSI)*, ser. GLSVLSI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 179–184.

[33] L. Aksoy, Q.-L. Nguyen, F. Almeida, J. Raik, M.-L. Flottes, S. Dupuis, and S. Pagliarini, "Hybrid protection of digital fir filters," *IEEE Transactions on Very Large Scale Integration VLSI Systems (TVLSI)*, vol. 31, no. 6, pp. 812–825, 2023.

[34] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-sat era by locking & obfuscation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 13–18.

[35] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 17, pp. 744–759, 2022.

[36] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers (TC)*, vol. 64, no. 2, pp. 410–424, 2015.

[37] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *Design Automation Conference (DAC)*, 2012, pp. 83–89.

[38] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 8, no. 2, pp. 517–532, 2020.

[39] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the approximation resiliency of logic locking and ic camouflaging schemes," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 2, pp. 347–359, 2019.

[40] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based Bit-Flipping Attack on Logic Encryptions," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 629–632.

[41] A. Raj, N. Avula, P. Das, D. Sisejkovic, F. Merchant, and A. Acharyya, "Deepattack: A deep learning based oracle-less attack on logic locking," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[42] F. Almeida, L. Aksoy, Q.-L. Nguyen, S. Dupuis, M.-L. Flottes, and S. Pagliarini, "Resynthesis-based attacks against logic locking," in *24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–8.

[43] Cadence Design Systems, Inc., *Genus Synthesis Solution*, 2024, san Jose, CA, USA. [Online]. Available: https://www.cadence.com

[44] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1985, pp. 663–698.

[45] F. Corno, M. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.

[46] K. Shamsi, "Netlist Encryption and Obfuscation Suite," 2021. [Online]. Available: https://bitbucket.org/kavehshm/neos/src/master/

[47] A. Alaql, "Scope," Available: https://github.com/alaql89/SCOPE.

Felipe Almeida received his bachelor's degree in Computer Engineering from the Pernambuco University and a master's degree in Microelectronics from the Federal University of Rio Grande do Sul. He is currently affiliated with the Centre for Hardware Security at Tallinn University of Technology (TalTech) as a Ph.D. student. His research interests are hardware security and radiation tolerance circuits.

Levent Aksoy received his Ph.D. degree in electronics engineering from Istanbul Technical University (ITU), Istanbul, Türkiye, in 2009. He worked as a researcher at ITU and INESC-ID, Lisbon, Portugal. He also worked at Dialog Semiconductor, Istanbul, Türkiye, as a senior digital design engineer. Currently, he is a senior researcher at the Centre for Hardware Security at Tallinn University of Technology (TalTech), Tallinn, Estonia. He is the recipient of the best paper awards at EUC'15 and DDECS'22 and the first prize in HeLLO CTF'22. His research interests include hardware security and CAD for VLSI circuits with emphasis on solving EDA problems using SAT models and optimization techniques.

Samuel Pagliarini received the PhD degree from Telecom ParisTech, Paris, France, in 2013. He has held postdoctoral research positions at the University of Bristol, Bristol, UK, and at Carnegie Mellon University, Pittsburgh, PA, USA. He led the Centre for Hardware Security at Tallinn University of Technology, Tallinn, Estonia, from 2019 to 2023. His current research interests include many facets of digital circuit design, with a focus on circuit reliability, dependability, and security.

Curriculum Vitae

1. Personal data

Name	Antonio Felipe Costa de Almeida
Date and place of birth	12 November 1984, Recife, Brazil
Nationality	Brazillian

2. Contact information

Phone	+49 0152 35897889
E-mail	lipinhw@gmail.com

3. Education

2019–present	Tallinn University of Technology (Taltech) School of Information Technologies Information and Communication Technology, PhD studies
2010–2012	Federal University of Rio Grande do Sul (UFRGS) Graduate Program in Microelectronics Microeletronics Engineering, MSc
2009–2010	Federal University of Rio Grande do Sul (UFRGS) IC Digital Design Training
2003–2008	University of Pernambuco (UPE) Department of Computer Engineering Computer Engineering, BSc

4. Language competence

Portuguese	native
English	fluent
Italian	advanced

5. Professional employment

2023–2024	Elmos Semiconductor, Backend Engineer
2014–2019	Politecnico di Torino, Researcher Assistant
2019–2020	CPqD, Backend Team Leader
2014–2019	NSCAD Microelectronic, Physical Design Engineer and Instructor
2019–2020	PortoChip, Digital Design Engineer

6. Computer skills

- Operating systems: GNU/Linux and Windows
- Document preparation: Emacs and LaTeX
- Programming languages: C/C++, C#, Java and Python
- Hardware description languages: VHDL, Verilog and System Verilog

7. Honours and awards

- 2022, 1st Place Award, Attacks on Logic Locking, Obfuscation, Fine-grain Hardware Redaction, & Routing Table Configuration, HeLLO CTF 2022
- 2022, 3rd Place Award, Security Closure of Physical Layouts Design Contest, International Symposium on Physical Design (ISPD)

8. Defended theses

- 2012, Evaluating Placement Constraints and Majority Voter Insertion Techniques in Triple Modular Redundancy, Prof. Dr.Fernanda Kastensmidt Federal University of Rio Grande do Sul, Graduate Program in Microelectronics
- 2008, An Approach to Control Unwanted Messages in Networks using Mail-pot, Prof. Dr. Wellington dos Santos. Universty of Pernambuco (UPE), Department of Computer Engineering

9. Field of research

- Hardware security
- Hardware trojans
- Integrated circuits design

Curriculum Vitae (Estonian)

1. Isikuandmed

Nimi	Antonio Felipe Costa de Almeida
Sünniaeg ja -koht	12. november 1984, Recife, Brasiilia
Kodakondsus	Brasiilia

2. Kontaktandmed

Telefon	+49 0152 35897889
E-post	lipinhw@gmail.com

3. Haridus

2019–praegu	Tallinna Tehnikaülikool (Taltech)
	Infotehnoloogia teaduskond
	Infotehnoloogia ja kommunikatsioonitehnoloogia, doktorantuur
2010–2012	Rio Grande do Suli Föderaalülikool (UFRGS)
	Mikroelektroonika magistriõppe programm
	Mikroelektroonika inseneriteadus, magistrikraad
2009–2010	Rio Grande do Suli Föderaalülikool (UFRGS)
	IC digitaalse projekteerimise koolitus
2003–2008	Pernambuco Ülikool (UPE)
	Arvutitehnika osakond
	Arvutitehnika, bakalaureusekraad

4. Keeleoskus

Portugali keel	emakeel
Inglise keel	sorav
Itaalia keel	edasijõudnud

5. Töökogemus

2023–2024	Elmos Semiconductor, tagatöötluse insener
2014–2019	Torino Polütehnikum, teadusassistent
2019–2020	CPqD, tagatöötluse tiimijuht
2014–2019	NSCAD Mikroelektroonika, füüsilise projekteerimise insener ja instruktor
2019–2020	PortoChip, digitaalse projekteerimise insener

6. Arvutioskused

- Operatsioonisüsteemid: GNU/Linux ja Windows
- Dokumenditöötlus: Emacs ja LaTeX
- Programmeerimiskeeled: C/C++, C#, Java ja Python
- Riistvara kirjeldamise keeled: VHDL, Verilog ja System Verilog

7. Tunnustused ja auhinnad

- 2022, 1. koht, Rünnakud loogikalukustuse, peitmise, peenkraanriistvara redigeerimise ja marsruutimistabeli konfiguratsiooni vastu, HeLLO CTF 2022
- 2022, 3. koht, Füüsilise paigutuse kujunduskonkursi turvalisuse lõpetamine, Rahvusvaheline füüsilise disaini sümpoosion (ISPD)

8. Kaitstud lõputööd

- 2012, Paigutuspiirangute hindamine ja kolmikmodulaarse liigvoolu sisestamise tehnikad, prof dr Fernanda Kastensmidt Rio Grande do Suli Föderaalülikool, mikroelektroonika magistriõppe programm
- 2008, Lähend soovimatute sõnumite kontrollimiseks võrkudes kasutades Mail-pot'i, prof dr Wellington dos Santos. Pernambuco Ülikool (UPE), arvutitehnika osakond

9. Uurimisvaldkond

- Riistvara turvalisus
- Riistvaralised troojalased
- Integraallülituste projekteerimine

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-323-3 (PDF)