Keynote: A Disquisition on Logic Locking

Abhishek Chakraborty, *Student Member, IEEE*, Nithyashankari Gummidipoondi Jayasankaran, *Student Member, IEEE*, Yuntao Liu, *Student Member, IEEE*, Jeyavijayan Rajendran, *Member, IEEE*, Ozgur Sinanoglu, *Member, IEEE*, Ankur Srivastava, *Member, IEEE*, Yang Xie, *Student Member, IEEE*, Muhammad Yasin, *Member, IEEE*, and Michael Zuzak, *Student Member, IEEE*

***Abstract*—The fabless business model has given rise to many security threats, including piracy of intellectual property (IP), overproduction, counterfeiting, reverse engineering (RE), and hardware Trojans (HT). Such threats severely undermine the benefits of the fabless model. Among the countermeasures developed to thwart piracy and RE attacks, *logic locking* has emerged as a promising and versatile solution that is being adopted by both academia and industry. The idea behind logic locking is to lock the design using a "keying" mechanism; only the rightful owner has control over the locked design. Therefore, the design remains non-functional without the knowledge of the key.**

In this paper, we survey the evolution of logic locking over the last decade. We introduce various "cat and mouse" games involved in logic locking along with its novel applications—including, processor pipelines, graphics-processing units (GPUs), and analog circuits. We aim this paper to be a primer for researchers interested in developing new logic locking techniques and employing logic locking in different application domains.

***Index Terms*—Design for security, Intellectual property protection, Hardware supply-chain security, Logic locking, Boolean satisfiability**

I. Introduction

A. Design-for-trust (DfTr) Techniques

The increasing cost of integrated circuit (IC) manufacturing has forced many companies, such as Qualcomm and Broadcom, to operate fabless: i.e., outsource the expensive fabrication process to offshore foundries such as TSMC and Samsung. Today, there are approximately 1350 fabless design companies [11]. While this outsourcing has provided many benefits such as reduction in cost and time-to-market, the increased accessibility of valuable assets to potentially untrusted entities has led to many security vulnerabilities [12]. Hardware security threats include IP piracy by rogue agents in integration houses, unauthorized overproduction of ICs at offshore foundries, counterfeiting of ICs at recycling facilities [13], RE by untrusted end-users [14], and insertion of hardware Trojans at multiple avenues in the IC supply chain [12], [14]–[16].

The work was supported by in part by the Office of Naval Research (ONR Award #N00014-18-1-2058), the Air Force Office of Scientific Research (AFOSR MURI FA9550-14-1-0351), the National Science Foundation (NSF CNS-1822848), the Defense Advanced Research Projects Agency (DARPA HR001116S0001-FP39), and the NYUAD Center for Cyber Security (CCS). The author names are listed in alphabetical order.

N. G. Jayasankaran, JV Rajendran, and M. Yasin are with the Electrical and Computer Engineering Department at the Texas A&M University, TX 77843 USA (Correspondence email: {myasin, jv.rajendran}@tamu.edu).

A. Chakraborty, Y. Liu, A. Srivastava, Y. Xie, and M. Zuzak are with the Electrical and Computer Engineering Department at the University of Maryland College Park, MD 20742 USA.

O. Sinanoglu is with the Division of Engineering at the New York University Abu Dhabi, 129188 UAE.

TABLE I
PROTECTION OFFERED BY DFTR TECHNIQUES AGAINST UNTRUSTED ENTITIES IN THE IC SUPPLY CHAIN. ✓ DENOTES THAT A TECHNIQUE CAN PROTECT PIRACY CONDUCTED BY AN UNTRUSTED ENTITY. × DENOTES THAT A TECHNIQUE CANNOT PROTECT PIRACY CONDUCTED BY AN UNTRUSTED ENTITY.

DfTr Technique	3PIP vendor	SoC integrator	Foundry	Test facility	End-user
Watermarking [17], [18]	×	×	×	✓	✓
Camouflaging [21]–[23]	×	×	×	×	✓
Split manufacturing [25], [26]	×	×	✓	×	×
Metering (passive) [19], [20]	×	×	×	✓	✓
Logic locking [1]–[10]	✓	✓	✓	✓	✓

Over the last decade, researchers have developed a plethora of defense solutions to enforce security and trust in the IC supply chain. Collectively referred to as DfTr techniques, these defenses include: 1) watermarking that embeds a designer's signature into the design [17], [18], 2) metering that enables tracking of individual ICs throughout their lifetime [19], [20], 3) camouflaging that introduces look-alike structures at the layout-level [21]–[24], 4) split manufacturing that involves partial fabrication at two separate foundries [25], [26], and 5) logic locking that locks a design with key-controlled protection logic [1]–[10], [27].

Logic locking can protect against potential attackers located anywhere in the IC supply chain. This can include a rogue system-on-chip (SoC) integrator, an untrusted foundry, an untrusted test facility, or a malicious end-user. As illustrated in Table I, most of the other DfTr techniques, such as camouflaging or split manufacturing, can only protect against a limited set of malicious entities. Consequently, logic locking has garnered significant interest, not only from the research community but also from industry and government agencies. Mentor Graphics, a major computer-aided design (CAD) tool provider, has launched TrustChain, a framework that supports logic locking and camouflaging [28]. Defense Advanced Research Projects Agency (DARPA) has recently started the *Obfuscated Manufacturing of GPS* program to develop locking techniques against an untrusted foundry [29].

B. Logic locking

Logic locking modifies the target design-to-be-protected by locking the original design with a secret key; the resulting design is then called the *locked* design. As illustrated in Fig. 2, in addition to the functional inputs, the locked design also has *key inputs*. These key inputs carry the secret key to the the locked design. Only upon applying the correct values to the key inputs, which collectively form the key, the locked

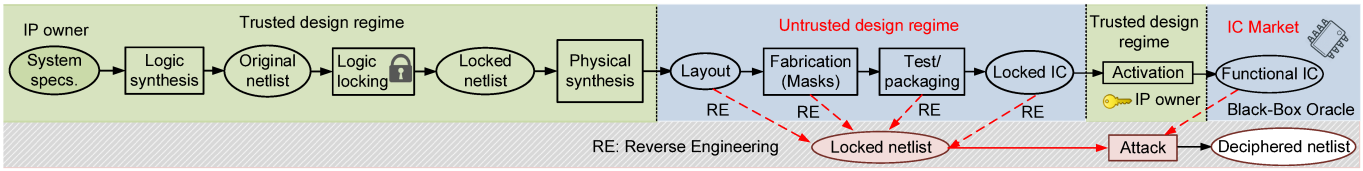


Fig. 1. The IC design flow including logic locking [1]–[10].

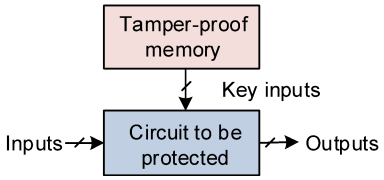


Fig. 2. An illustration of a locked design. Only on applying the correct key, the design is functional [30].

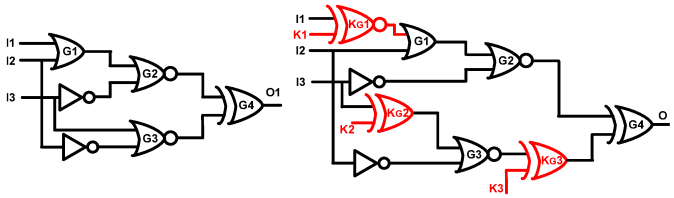


Fig. 3. An illustration of logic locking with the original and locked circuit [1], [31]. The correct key value is 100.

design becomes functionally equivalent to the original design. The key is stored in a secure, tamper-proof memory, whose contents cannot be tampered with or accessed outside the chip.

As shown in Fig. 1, the locking procedure is carried out in the trusted design house. The locked netlist passes through the untrusted design and manufacturing phases. Without the secret key, the attacker cannot obtain the original design. Thus, locking prevents the attackers from reverse engineering (RE) or overproducing the design. Moreover, it makes it harder to identify safe places to insert hardware Trojans [30], [32]. Upon fabrication, a chip is *activated* by loading the secret key onto the chip either 1) in a trusted facility or 2) remotely via a secure key exchange protocol (see Section II-D), thereby making the chip functional [1], [33].

Example. Fig. 3 shows an original netlist and its locked version, which is obtained by inserting three XOR/XNOR key-gates into the original netlist. The correct key for the netlist is 100, which effectively turns all key-gates into buffers, leading to the correct output. For an incorrect key, however, errors are introduced in the circuit output.

C. Evolution of logic locking

Over the years, many locking techniques have been developed. In the earliest ones, locking is facilitated by inserting XOR/XNOR gates [1]–[3] or look-up tables (LUTs) [34]. Later, techniques based on VLSI testing principles have been proposed to increase the effectiveness of locking techniques, i.e., an incorrect key should result in incorrect output [35], [36]. This has also led to many key-recovery attacks that exploit the vulnerabilities of logic locking techniques [2], [33], [36]–[38]. A powerful attack that has broken all the aforementioned techniques is the *Boolean satisfiability (SAT)-based key-pruning attack*, referred to as the *SAT attack*. The research in the post-SAT era focuses on defending against the SAT attack through the use of primitives such as point-functions [4], [5], [39]. The emphasis on a specific class of security primitives has spawned newer classes of attacks such as removal attacks [10], [40], [41] and approximate attacks [8], [9]. Traditional, as well as newer side-channel attacks such as differential power analysis (DPA) attack [36], template-based power analysis attack [42], and test-data mining (TDM) attack [33] have also been mounted on logic locking. This "cat and mouse" game continues until today with the SAT attack being the most notable attack and the point-functions being the most widely adopted security primitive [4], [5], [7], [39]. Besides combinational logic locking, sequential locking techniques that lock/obfuscate finite state machine (FSM) have also been developed [19], [43], [44].

Beyond the traditional use in locking the functionality of a circuit, recent research also reveals unconventional applications of logic locking. Logic locking is now being deployed to lock the parametric behavior, e.g., timing and performance, of not only general processing units (GPUs) and processors [45] but also analog and mixed-signal (AMS) circuits [46]. Application-specific logic locking in the context of digital signal processing (DSP) circuits and microfluidic biochips is also being explored [47], [48]. These diverse developments span more than a decade of research. The increasing interest in logic locking and its applications have given rise to the need for a systematization-of-knowledge in this emerging field.

D. Contributions

The primary goal of this paper is to survey the research in logic locking, systematizing the knowledge, trends, and future research directions. We aim that this paper can be used as a primer for logic locking by electronic design automation (EDA) researchers, VLSI engineers, security experts, circuit designers, and others interested in the general realm of hardware security. The contributions of the paper are:

- Surveying the major logic locking techniques and attacks published in the literature.
- Classifying the existing attacks and defenses to elucidate which defense techniques are resilient against what attacks.
- Elaborating the new applications of logic locking, which are beyond just protecting the functionality.
- Summarizing the trends in the field of logic locking and highlighting future research directions.

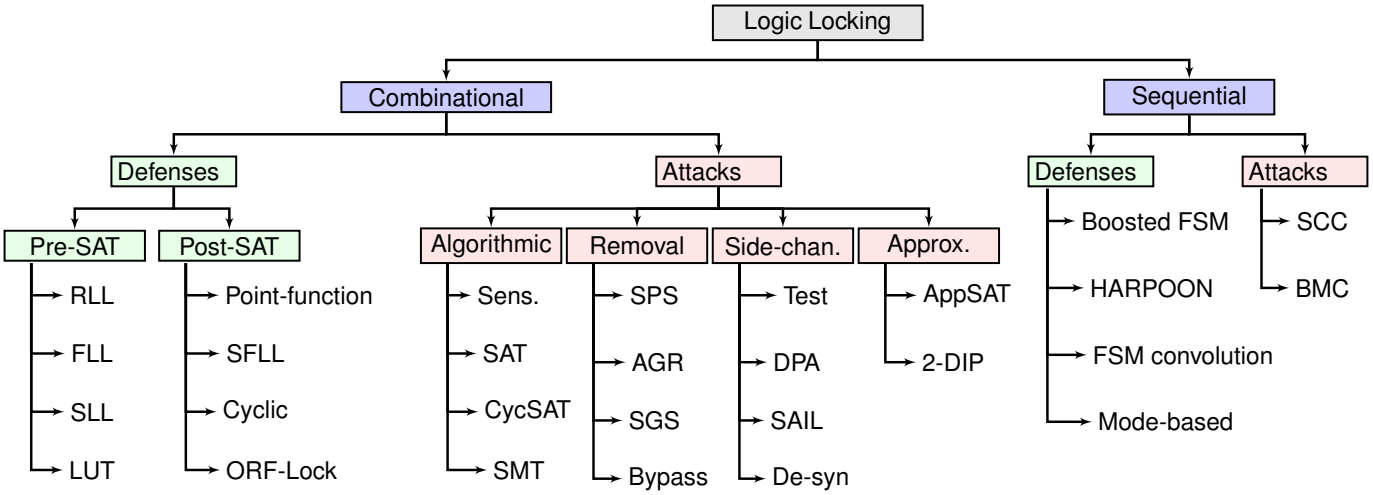


Fig. 4. Classification of logic locking attacks and defenses. Random Logic Locking (RLL) [1], [31], Fault-analysis-based Logic Locking (FLL) [3], Strong Logic Locking (SLL) [2], [49], Stripped Functionality Logic Locking (SFLL) [30], One Way Function-based lock (OWF-lock) [49], Sensitization attack (Sens.) [2], Boolean SATisfiability (SAT) [38], Cyclic SAT (CycSAT) [50], Satisfiability Modulo Theories (SMT) [51], Signal Probability Skew (SPS) [41], Approximate SAT (AppSAT) [8], AppSAT-Guided-Removal (AGR) [40], Sensitization-Guided-SAT (SGS) [40], Differntial Power Analysis (DPA) [36], Structural Analysis using machIne Learning (SAIL) [52], De-synthesis (de-synth) [53], Double Distinguishing Input Pattern (2-DIP) [9], HARdware Protection through Obfuscation Of Netlist (HARPOON) [43], Strongly Connected Component (SCC) [54]–[56], and Bounded Model Checking (BMC) [57], [58].

The paper is organized as follows. Section II presents the threat model utilized by logic locking, the protocol used for activating locked ICs, and classification of logic locking attacks and defenses. Sections III–VI focus on combinational logic locking. Specifically, Section III discusses the pre-SAT logic locking techniques, while Section IV focuses on the SAT and other oracle-guided attacks. Section V highlights various post-SAT defenses. However, it should be noted that many of these defenses are vulnerable to the removal and approximate attacks, as detailed in Section VI. In Section VII, we elaborate on the sequential logic locking defenses and attacks. Section VIII highlights how traditional logic locking has found new applications in innovative ways. Section IX summarizes the challenges and future research directions and, finally, Section X concludes the paper.

II. Logic Locking in IC Supply chain

In this section, we describe how logic locking is used in an IC supply chain to protect a design from the untrusted entities. To this end, the threat model is explained, detailing the capabilities of the defender and the attacker, and more importantly, their constraints. This is followed by a classification of logic locking defenses and attacks.

A. Threat model

The de-facto threat model, followed by the latest logic locking research, assumes that only the IP owner has access to the proprietary design details and the secret key within the *trusted design regime* [1], [2], [8], [9], [36]–[38], [42], [50], [57], [59]. The attacker could be an untrusted: 1) SoC integrator, 2) foundry, 3) test/assembly facility, or 4) end-user. The attacker's objective is to pirate the valuable IP either through theft or RE, overbuild or counterfeit the IC design. Since the core of IP piracy is complete/partial RE, we describe it in more detail in the following section.

Reverse engineering (RE). A malicious end-user can obtain the locked gate-level netlist by utilizing state-of-the-art RE techniques [60]. RE of an IC is a process of identifying its structure, design, and functionality. RE is a multi-step process involving de-packaging an IC, delayering it, imaging individual layers, and analyzing the collected images to extract the netlist [14]. Companies such as Chipworks [61] and Silicon Investigation [62] provide RE services. Degate [63] offers tools to aid RE. There are many tutorials available for RE [64]. Technology node scaling has not hampered RE [65].

The standard logic locking threat model assumes that the attacker has access to the following:

1) *Locked netlist.* An end-user can obtain the locked netlist by RE an IC. An attacker in the untrusted foundry can extract the same information from the GDSII files containing the layout delivered to an untrusted foundry for fabrication.
2) *Activated chip.* The attacker can buy an activated chip (a.k.a. functional IC) from the market. A functional chip serves as an oracle to an attacker who can apply inputs of his/her choice and can observe the correct outputs. The presence of oracle enables oracle-guided attacks.

Logic locking assumes that the attacker cannot insert probes into a logic-locked IC to read the secret key. While this straightforward attack can quash logic locking, it also leads to a wider security problem of leaking secrets from the chip [66]. However, protection mechanisms, such as analog shields, exist to prevent such attacks [67]. Logic locking (as well as other DfTr countermeasures) typically assume that such protection mechanisms are in place to prevent probing attacks.

B. Practical concerns: Unique keys for each IC

Logic locking, by itself, locks all fabricated circuits with a common key. If this common key is directly given to a malicious user, she/he can leak it to the untrusted foundry. The foundry can then use this key to unlock the overproduced

chips. To defeat this type of attack, the designer must have unique keys for every chip. In this way, a user cannot distribute the keys she/he obtained for one chip to unlock other chips. To facilitate this mechanism, one needs a device-specific ID. Circuits such as physical unclonable functions (PUFs) or true random number generators (TRNGs) can be utilized to generate unique keys for each IC.

PUFs leverage process variations to generate unique challenge-response pairs for each fabricated device [68]. Upon power-up, a PUF is supplied with a challenge; the unique PUF response can serve as the secret key for the chip, which is loaded to the on-chip tamper-proof memory. Upon recording the PUF response, read access to PUF output is disabled. As explained in the next subsection, with the availability of on-chip crypto infrastructure, the designer may deliver the common key remotely to the chip [1], [19]. PUF circuits can themselves be error-prone, leading to erroneous responses and thus generating erroneous common keys. To alleviate this problem, researchers have developed coding schemes to correct the errors generated in PUFs response [68].

C. Tamper-evident memories

Semiconductor companies, such as Maxim Integrated [69] and Altera [70], have been designing tamper-proof memory designs. In fact, such memories are used not only for logic locking, but also to store cryptographic keys in access-control systems, network-storage servers, set-top boxes, etc. [71]. DS3660 chip from Maxim Integrated is an example of tamper-proof memory. A directive from the U.S. Department of Defense enforces that systems should have anti-tamper property [72].

Anti-tamper memories that are equipped with sensors, interlocks, and anti-tamper meshes offer protection against physical-tampering attacks and can have a storage capability of at least a few kilobytes. On detecting an attack, they have provisions to erase internal data; in case of logic locking, the secret keys are erased, rendering probing attacks ineffective [73].

Logic locking techniques can leverage an on-chip instance of such memory designs. The size requirement (few hundred bits) will be substantially less than the size of memories (few Kbytes at least) offered by companies. Such memories can be either on-chip or off-chip but in the context of logic locking, on-chip memory is preferred, as an attacker can snoop on the wires carrying the keys. While this can be prevented by using a secure communication link, this approach may prove cost prohibitive.

D. Remote activation protocol

A locked chip can be activated in a secure facility by the IP owner, which requires shipping of fabricated chips to the trusted facility. An alternative is to activate the chips at the foundry itself by making use of on-chip public key cryptography. As previously mentioned, unique keys for each fabricated chip may be generated by making use of PUFs or TRNGs. The EPIC — ending piracy of integrated circuits — protocol for remote activation of a locked chip is as follows [31]:

Step 1: As illustrated in Fig. 6, the designer locks the design with a common key (CK). She/He also embeds her/his public key MK-Pub in the design. The target chip contains a public-key cryptographic algorithm (e.g., RSA) and a PUF/TRNG. The locked design is sent to the untrusted foundry, where the chip is manufactured and tested for defects. Note that during the testing process, the key need not be loaded onto the chip [33].

Step 2: Upon first power-up, the chip generates a random number using the PUF/TRNG, which is consequently used to generate a public-private key pair, RCK-Pub and RCK-Pri, respectively. The foundry reads RCK-Pub and sends it to the designer.

Step 3: The designer encrypts CK with RCK-Pub and signs it with MK-Pri to obtain the user key (UK).

Step 4: The designer provides UK to the user of the chip, who then applies it and activates the chip. The RSA module within the chip decrypts UK with MK-Pub and RCK-Pri to obtain CK, enabling activation of the chip.

E. Classification of logic locking

We classify logic locking broadly into combinational and sequential categories. As shown in Fig. 4, in each category, there are multiple classes of attacks and defenses. The combinational locking techniques can be divided into pre-SAT and post-SAT. The pre-SAT defenses, which lock a circuit using XOR/XNOR or MUX key-gates, include random (RLL) [1], [31], fault-analysis-based (FLL) [3], strong (SLL) [2], [49], and LUT-based [34] logic locking. The post-SAT defenses branch off into four sub-categories: point-function-based locking [4], [5], [39], [74], stripped-functionality logic locking (SFLL) [30], cyclic locking [50], [75]–[80] and one-way function-based locking (OWF-Lock) [49] (see Section V).

The combinational locking attacks fall into four classes: 1) algorithmic, 2) approximate, 3) removal, and 4) side-channel attacks. The algorithmic attacks exploit the algorithmic weaknesses of logic locking techniques to retrieve the *exact* correct key. These attacks include the sensitization [2], SAT [38], cyclic SAT (CycSAT) [50], and satisfiability modulo theory (SMT) attack [51]. The approximate attacks, e.g., approximate SAT (AppSAT) [8] and double-distinguishing input pattern (double-DIP) [9], are variants of the SAT attack that recover only an *approximate* key. The removal attacks [40], [41] (see Section VI-A) utilize the structural properties of a netlist to distinguish the protection logic from the original circuit. The side-channel attacks include power side-channel attacks (DPA [36], [81] template-based [42]), test data attacks [33], [37], the de-synthesis attack [53] and the structural analysis using machine learning (SAIL) attack [52].

The sequential logic locking research focuses on locking (holding) the FSM in a set of *obfuscated states* until the correct input sequence (key) is applied. In boosted FSM (BFSM) [19] and HARPOON — hardware protection through the obfuscation of the netlist [43], there is a distinction between obfuscated and normal states. This distinction is minimized in subsequent techniques [44], [82], [83].

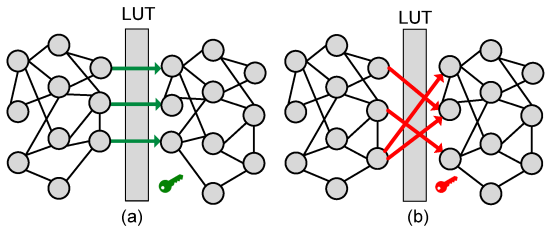


Fig. 5. A depiction of LUT-based locking. (a) The information flow is intact for the correct key. (b) For an incorrect key, the LUT modifies the information flow [34].

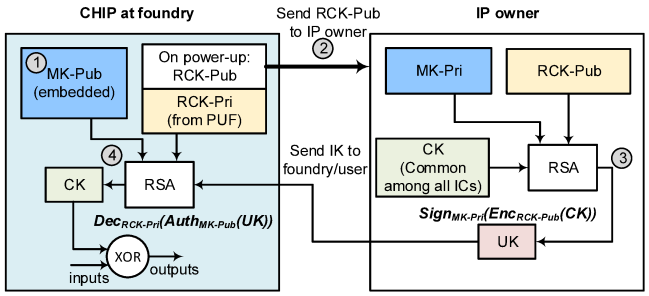


Fig. 6. EPIC protocol for remote activation of the locked chip [1], [31].

There are only a handful of attacks on sequential locking. The first class of attacks aims at RE a netlist, identifying the state registers, and reconstructing the FSM [44], [54]–[56]. These attacks are based on Tarjan's strongly connected component (SCC) identification algorithm. The second class of sequential locking attacks is based on bounded-model-checking (BMC) [57], [58].

III. Pre-SAT Logic Locking

This section focuses on the logic locking techniques introduced prior to the advent of the SAT attack. These techniques are the earliest attempts at thwarting IP piracy conducted by either an untrusted foundry or an untrusted end-user. None of these techniques are secure against SAT attack and are provided to explain the evolution of logic locking techniques.

A. Random logic locking (RLL)

EPIC introduced the first protocol and methodology for logic locking [1], [31]. EPIC locks a design by inserting XOR/XNOR gates at k random locations in a netlist while simultaneously meeting the timing constraints. For example, $k = 3$ key-gates are inserted randomly in the netlist in Fig. 3. To eradicate simple one-to-one between mapping the key-gate type (XOR/XNOR) and the key values (0/1), EPIC introduces additional inverters at the output of selected key-gates and bubble-pushes the inverters around the netlist.

B. Look-up table (LUT)-based locking

Instead of using XOR/XNOR gates as key-gates, LUTs can also be deployed [34]. The contents of the LUTs act as the secret key and thus as the logical barriers. Correct LUT programming ensures that the information flows within the design as desired. Modifications to the information flow lead to incorrect output, as illustrated in Fig. 5. The LUTs may be strategically placed using observability don't cares, such that incorrect keys will result in an optimal output corruptibility of 50% [34]. The output corruptibility represents the Hamming distance (HD) between the correct outputs and the incorrect outputs, obtained on applying random incorrect. An HD value of 50% results in the maximal ambiguity for the attacker.

Subsequent research suggests that lightweight and secure implementation of LUTs can be generated by 1) entangling smaller LUTs together [84], or 2) utilizing emerging technologies such as spin transfer torque to construct LUTs [85]–[87].

C. Fault-analysis-based logic locking (FLL)

RLL does not guarantee that an incorrect key leads to an incorrect output for all/most the input patterns. This causes the output corruptibility to remain relatively low ($< 50\%$) [3]. FLL utilizes the principles of VLSI testing to insert key-gates at locations that lead to the optimal output corruptibility of 50%. FLL characterizes the influence of a gate location G on the output using the notion of *fault impact* (FI) that encapsulates:

1) How many input patterns can activate a stuck-at-x, $x \in \{0, 1\}$ fault at the output of the gate G (NoP_x)?
2) Given a stuck-at fault at location G, how many primary outputs are affected (NoO_x)?

$$FI_G = (NoP_0 \times NoO_0) + (NoP_1 \times NoO_1) \quad (1)$$

By inserting XOR/XNOR key-gates at the output of the gates with the highest FI_G, an invalid key likely has the most impact on the outputs. This enables FLL to achieve an HD value of 50% [3]. XOR/XNOR key-gates can also be combined with MUX key-gates to achieve the same objective [88], [89].

Variants of FLL. The FLL algorithm is iterative; fault simulation is re-conducted upon insertion of each key-gate, leading to a large execution time. *Centrality-indicator*-based locking reduces the computational effort of FLL by traversing the circuit graph only once to compute metrics such as closeness-centrality and between-centrality [90]. Candidate locations are sorted based on the metric of interest, and the key-gates are inserted at the best k locations. *Weighted logic locking* (WLL) achieves a higher corruptibility with a fewer key-gates [91]. A traditional XOR gate, with the key input connected directly to a key-bit, introduces an error with 50% *actuation* probability. In WLL, an XOR key-gate is fed by multiple key-bits through additional AND/OR gates, which increases the actuation probability and leads to a higher output corruptibility. As simulating faults is computationally expensive, FLL can be significantly accelerated through field programmable gate array (FPGA) emulation [92]. This technique inserts key-gates at all candidate locations and loads the design to an FPGA, enabling faster fault simulation through the dynamic assignment of key values.

D. Strong logic locking (SLL)

RLL and FLL can be broken using the sensitization attack (see Section IV-A), which can leak the secret key, one bit at

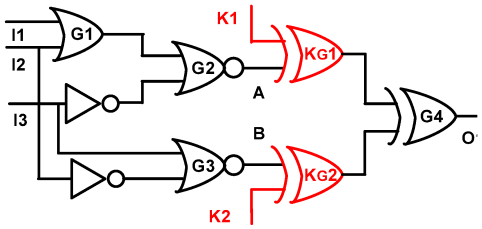


Fig. 7. A circuit locked using SLL [2]. Neither K1 nor K2 can be propagated to the output without controlling the other key-bit [2], [49].

a time, through the primary outputs. SLL hampers the attack by ensuring interdependence among key-gates and making it infeasible to sensitize the key-bits one at a time [2], [49]. As shown in Fig. 7, the attacker can control the value of both nets, A and B, by applying inputs of her/his choice. However, to determine the final output O1, she/he needs to know both K1 and K2 simultaneously. K1 and K2 are *pairwise secure* since neither of them can be sensitized to O1, without knowing or controlling the other key-bit.

Another similar countermeasure to the sensitization attack is key-inter-dependency-based locking (KLL) [93]. This technique iteratively inserts a key-gate G using FLL and then inserts key-gates in the logic cone of G to hamper sensitization. It also introduces a key-dependency block, comprising multiple layers of XOR/XNOR gates. Thus, the key-bits are fully mixed before being fed to the subsequent key-gates [93].

IV. Attacks on Pre-SAT Logic Locking

In this section, we present attacks launched against pre-SAT logic locking. The most notable among these attacks is the SAT attack, which has altered the field of logic locking.

A. Sensitization attack

The sensitization attack is the first oracle-guided attack. It determines individual key-bits by generating and applying patterns that sensitize them to the primary outputs of a functional IC [2]. For the locked netlist in Fig. 3, the key-bit K3 can be sensitized to the output O if the upper input to the gate G4 is set to 0 (its non-controlling value). This can be achieved by setting $I3 = 0$ and $I2 = 0$, regardless of the value of K1 and K2. Thus, when the input is set to 000, the output O will be equivalent to K3. To propagate K1 to the output, K3 is set with the value determined from the previous step. Likewise, K2 is found by setting K1 and K3 with the predetermined values. This attack circumvents RLL and FLL; however, it cannot break SLL [49].

B. Logic cone analysis (LCA) attack

A circuit can be divided into sub-circuits (a.k.a. logic cones), each consisting of the gates in the transitive fan-in of a primary output [88]. For RLL and FLL, the number of key inputs in individual logic cones may be relatively small, enabling brute-force attacks. A similar divide-and-conquer approach is adopted by the DPA attack [36], [81].

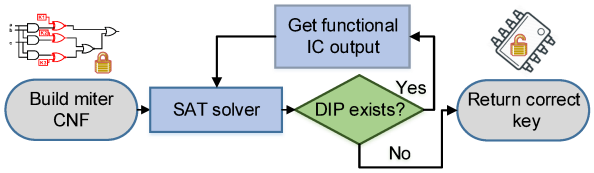


Fig. 8. Flowchart of the SAT attack [38].

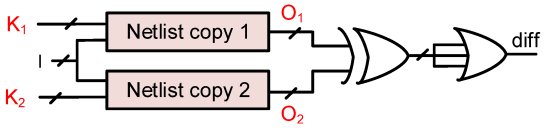


Fig. 9. Miter circuit to compute DIPs [38].

C. Boolean Satisfiability (SAT) attack

The SAT attack is the most effective of all logic locking attacks [38]: it can easily circumvent all the techniques discussed in Section III. The attack iteratively rules out incorrect keys using DIPs [38]. A DIP X_d is an input value for which (at least) two different keys, $K1$ and $K2$, produce differing outputs, $O1$ and $O2$, respectively. Since $O1$ and $O2$ are different, at least one of the key values is incorrect. In practice, a single DIP can eliminate multiple incorrect keys.

The DIPs are computed by constructing a miter circuit shown in Fig. 9. The primary inputs are common to the two copies of the locked circuit, while the key inputs are left independent. The corresponding outputs of the two circuits are XORed and then ORed to generate the *diff* signal. The conjunctive normal form (CNF) of the miter circuit is passed to a SAT solver, which computes a DIP X_d such that *diff=1*. X_d is applied to a functional IC (which serves as an oracle) and the correct output O_d is obtained. The input-output (I/O) pair (X_d, O_d) eliminates a subset of incorrect keys from the search space. To rule out the remaining incorrect keys, a new I/O pair is added to the SAT formula in each iteration, as illustrated in Fig. 8. The attack is successful when no further DIP is found, which implies that all incorrect key values have been pruned.

Example. Let us apply the SAT attack to the locked netlist shown in Fig. 3. Table II presents the output of the original circuit in column O and the output of the locked circuit for different key values (k0, k1,..., k7) in the following eight columns. In iteration 1, the DIP 000 is applied, for which four key values: k1, k3, k5, and k7 produce incorrect output, leading to the elimination of these key values from the search space. In iteration 2, two key values k0 and k2 are pruned using the DIP 001. In iteration 3, k6 is pruned using the DIP 010, identifying k4 as the correct key. This analysis also shows that the larger the number of incorrect key values ruled out per DIP, the fewer the DIPs needed for the attack, resulting in a smaller execution time. The SAT attack can also be used to de-obfuscate camouflaged circuits using incremental SAT solving [94], [95]. The camouflaged cells are modeled as programming vector, which are equivalent to the key inputs to the miter circuit. Based on the value of the key input, the camouflaged cell will represent a particular gate.

TABLE II
ANALYSIS OF THE SAT ATTACK AGAINST LOGIC LOCKING [38]. COLUMNS K0-K7 SHOW THE LOCKED CIRCUIT'S OUTPUT FOR DIFFERENT KEY VALUES. RED ENTRIES IN EACH ROW DENOTE AN INCORRECT OUTPUT. THE CORRECT KEY IS K4. NOTE THAT THE ATTACK DOES NOT EXPLICITLY PRUNE INDIVIDUAL KEYS.

Inputs			Original	Output O for different key values								
I1	I2	I3	Output O	K0	K1	K2	K3	K4	K5	K6	K7	Pruned Keys
0	0	0	0	0	1	0	1	0	1	0	1	iteration 1: K1, K3, K5, K7
0	0	1	1	0	1	0	1	1	0	1	0	iteration 2: K0, K2
0	1	0	1	1	0	0	1	1	0	0	1	iteration 3: K6
0	1	1	0	1	0	0	1	0	1	1	0	
1	0	0	0	0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	0	1	0	1	
1	1	0	1	1	0	0	1	1	0	0	1	
1	1	1	0	1	0	0	1	0	1	1	0	

D. Oracle-less attacks

Unlike the afore-mentioned attacks, the following attacks do not need an oracle. Thus, a malicious foundry can use these attacks to find the key without buying the golden chip from the market.

1) Test-data attacks: Test data is provided to the foundry/test facility for use in the manufacturing test. For logic-locked ICs, information about the secret key may be embedded in the test data if care is not taken during automatic test pattern generation (ATPG) [33]. These oracle-less attacks, i.e., the hill-climbing attack [37] and the test-data-mining (TDM) attack [33], use test data instead of the functional IC.

The hill-climbing attack simulates the locked circuit with 1) an initial key guess and 2) a key derived by flipping a bit in the initial key. The bit flip is retained, i.e., the derived key becomes the subsequent initial key, if it reduces the HD between the circuit output and the test response. The TDM attack uses test stimuli and responses as constraints and conducts ATPG to extract a key. In addition to satisfying the test data constraints, the key also maximizes the fault coverage. By considering the fault coverage, the TDM attack can outperform the hill-climbing attack.

2) De-synthesis attack: The de-synthesis attack is an oracle-less attack that recovers the secret key using only a locked netlist [53]. The attack uses a hill-climbing approach to re-synthesize the locked netlist for different key guesses. The key guess that yields the maximum similarity between the locked netlist and its re-synthesized versions is considered as the correct key. When launched on RLL, the attack can recover around 65% of key-bits correctly.

Another impact of incorrect keys is to introduce logic redundancy in a netlist, rendering certain faults untestable [96]. The redundancy elimination attack identifies the most likely value for a key-bit by comparing the changes to the number of untestable faults for a key-bit value of zero and a key-bit value of one [96].

3) Machine learning-based attacks: The SAIL attack is an oracle-less attack that uses machine learning to guess the secret by characterizing the impact of key-gate insertion on structure of a locked netlist [52]. A key observation is that insertion of a key-gate and the subsequent re-synthesis involves only a handful of local structural transformations encompassing three or fewer levels of logic. SAIL trains a neural network on examples of pre- and post-synthesis netlists. The trained model can revert a target netlist to its pre-synthesis stage, extracting key values from the key-gate types. When launched against RLL, the attack can retrieve up to 94% key-bits for one-level changes and only about 49% key for three-level changes [52].

BOCAnet, another machine learning-based "black-box" attack, trains a neural network only on input/output pairs [97]. The attack essentially tries to learn a Boolean function from a set of I/O pairs. It can be configured to find the secret key from a set of I/O pairs, the output for a given input, or even the input for a given output.

V. POST-SAT LOGIC LOCKING

The effectiveness of the SAT and other attacks against pre-SAT locking techniques mandates defenses that can resist these attacks. Section V-A presents the first class of such defenses that use point-functions to render the number of SAT attack iterations exponential in the key size. These defenses, however, exhibit weakness against approximate and removal attacks. SFLL overcomes these limitations. Section V-B elaborates on the three variants of SFLL. Section V-C and V-D introduce cyclic logic locking and one-way function-based locking (OWF-Lock), respectively, that are alternative attempts to defeat the SAT attack.

A. Point-function-based logic locking

The SAT attack is successful against the existing logic locking techniques as the number of DIPs needed to break these techniques is quite small: 90% of the circuits studied by [38] can be broken with 250 or fewer DIPs. SARLock, Anti-SAT, and ATD, all make use of point-functions such as AND-trees to control the distinguishing ability of DIPs. Point-functions are Boolean functions that produce a one for only one input pattern (i.e., minterm); the output is zero otherwise. Point-function-based locking techniques push the SAT attack towards its worst-case scenario. This happens when the attack can rule out at most one incorrect key value per DIP [4], [5], [39]. Table III illustrates such a scenario for a circuit with three primary inputs and three key inputs. In each row, there is at most one key value that generates an incorrect output. The SAT attack would require $2^3 - 1 = 7$ DIPs to succeed in this case; in general, the number of DIPs $\approx 2^k$.

1) SAT attack resilient logic locking (SARLock): SARLock comprises a comparator (constituted using XNOR gates and an AND-tree) whose output is XORed with the original circuit, as

TABLE III
RESISTING THE SAT ATTACK BY CONTROLLING THE DISTINGUISHING ABILITY OF DIPS [4]. AT MOST ONE INCORRECT KEY CORRUPTS THE OUTPUT FOR ANY DIP. K4 IS THE CORRECT KEY.

Inputs			Original	Output O for different key values							
I1	I2	I3	Output O	K0	K1	K2	K3	K4	K5	K6	K7
0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1
0	1	1	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0
1	1	0	1	1	1	1	1	1	1	0	1
1	1	1	0	0	0	0	0	0	0	0	1

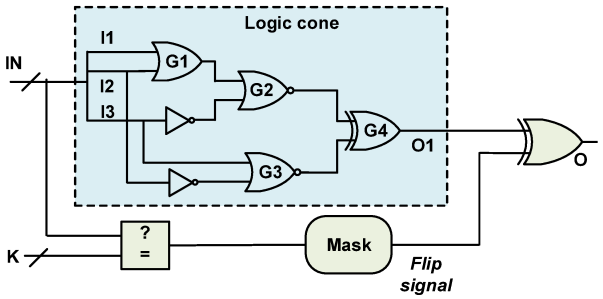


Fig. 10. A SARLock circuit [4]. The *flip* signal is asserted upon a match between the primary inputs and the incorrect key. The blue area shows the logic cone under protection and blocks in green represents the protection logic.

shown in Fig. 10. The comparator generates a *flip* signal that is asserted when the input pattern matches the key value [4]. To prevent the *flip* signal from being asserted for the correct key value, which is 100 in Table III, a mask logic is inserted. This logic masks the flip signal whenever the correct key is applied. The number of DIPs for the locked circuit is $2^k - 1$. The protection logic consists of $k+1$ two-input XOR/XNOR gates and $2k+1$ two-input AND gates, ensuring minimal implementation overhead. On increasing the key size, k, the area overhead grows linearly, while there is an exponential increase in the number of DIPs.

While *SARLock* thwarts SAT attack, a removal attack (refer to Section VI-A) can identify the comparator from its unique structure and remove the SARLock circuitry, revealing the original circuit from its silicon implementation. Another fundamental shortcoming of SARLock and other point-function-based techniques is the low output corruptibility.

2) Anti-SAT: The Anti-SAT block integrates two complementary Boolean functions (e.g., an AND and a NAND) to deliver maximal security against the SAT attack in a configurable fashion [5]. Fig. 11 illustrates one configuration of the Anti-SAT with two logic blocks g and $\overline{g}$, which share the same set of inputs X. Two sets of key-gates, denoted as K_{l1} and K_{l2}, are inserted at the inputs of the two blocks. The output of g and $\overline{g}$ are fed into an AND gate to form the final output Y. Consequently, $Y = g(X \oplus K_{l1}) \wedge \overline{g(X \oplus K_{l2})}$. The correct key (which can be achieved by setting $K_{l1} = K_{l2}$) makes $Y = 0$ for all input patterns. For an incorrect key ($K_{l1} \neq K_{l2}$), $Y = 1$ for specific input patterns, resulting in an incorrect output. Then,

Let us assume a Boolean function g with n inputs and p

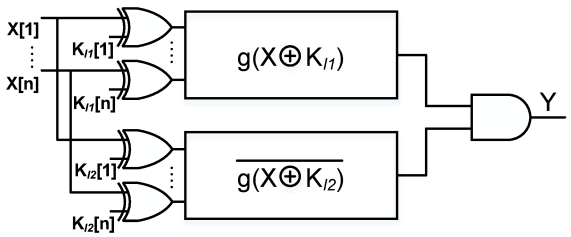


Fig. 11. Anti-SAT consists of two complementary blocks [5].

denoting the cardinality of its on-set. Note that on-set is the set of input vectors that make g equal to one. Then,

$$\#DIPs \geqslant \frac{2^{2n} - 2^n}{p(2^n - p)} \tag{2}$$

When p is sufficiently close to 1 or sufficiently close to $2^n - 1$, the lower bound is exponential in n, resulting in 2^n DIPs. Compared to SARLock, Anti-SAT allows a designer to configure the security level by varying p, albeit at a slightly higher overhead.

Strong Anti-SAT inherits the complementary structure of Anti-SAT and increases the number of incorrect keys that cause the error to certain critical input minterms [98]. This ensures high application-level error for inherently error-resilient applications (e.g., machine learning) while maintaining the robustness against SAT attack. Anti-SAT and SAS are vulnerable to removal attacks (see Section VI-A).

3) AND-tree Detection (ATD): ATD finds and locks AND/OR trees that already exist in the original circuit with the objective of reducing the implementation cost [39]. However, large trees rarely exist in typical circuits, mandating the use of large dummy trees that are susceptible to removal. Similar to SARLock and Anti-SAT, ATD has low output corruptibility and is also vulnerable to removal attacks.

4) Diversified tree logic locking (DTL): DTL is an extension of point-function locking. It resists the approximate attacks by replacing a point-function with a Boolean function of higher on-set, similar to that in Anti-SAT [74]. DTL generates the desired Boolean function by systematically replacing selected gates inside an AND-tree with NAND/NOR/OR gates. DTL is not secure against removal attacks unless combined with other techniques such as SFLL-Flex or SFLL-fault.

B. Stripped-functionality logic locking (SFLL)

To overcome the vulnerabilities of point-function-based locking, i.e., vulnerability to removal attack and low output corruptibility, SFLL adopts the philosophy of stripped-functionality. It modifies the original circuit and later restores the functionality by applying the correct key. The locked circuit comprises a *functionality stripped circuit* (FSC) and a restore unit.

1) SFLL-HD: Functionality-stripping can be implemented in various flavors. In one flavor, the original circuit is modified by introducing built-in errors only for the input patterns that are of HD h from the secret key, i.e., $O\_locked \neq O, if(HD(IN, K_{secret}) \neq h)$ where k_{secret} is known only to the designer. Such input patterns are referred to as the *protected input patterns* (PIPs). Naturally, the restore operation is also based on the HD between the input pattern and the secret key; the FSC output is restored (flipped back) only when $HD(K, IN) = h$.

For the special case of $h = 0$, illustrated in Fig. 12 and Table IV, there is only one PIP 100, which is the same as the secret key [7]. The output of the FSC is inverted for the pattern 100. Only upon the application of the correct key K4= 100 to the restore unit. For a given k and h, the removal attack resilience is $\binom{k}{h}$, which denotes the number of PIPs. The

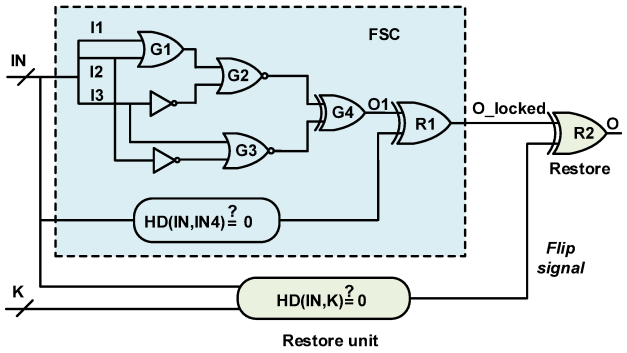


Fig. 12. An SFLL-HD^0 circuit with $n = k = 3$ and $IN = 100$ being the protected input pattern. The FSC hardcodes the secret key K4. The blue area shows the FSC and the blocks in green represent the protection logic [30].

TABLE IV
THE FSC IS MINIMALLY DIFFERENT FROM THE ORIGINAL FUNCTION: THEY PRODUCE A DIFFERENT RESPONSE FOR ONE PROTECTED INPUT PATTERN (IN = 100). THE RESTORE UNIT CANCELS THIS ERROR FOR THE CORRECT KEY K4 [30].

Inputs			Original		Output O for different key values							
I1	I2	I3	O1	O_locked	K0	K1	K2	K3	K4	K5	K6	K7
0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	1	1	1	1
0	1	0	1	1	1	1	0	1	1	1	1	1
0	1	1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	1	1	1	1	0	1	1	1
1	0	1	0	0	0	0	0	0	0	1	0	0
1	1	0	1	1	1	1	1	1	1	1	0	1
1	1	1	0	0	0	0	0	0	0	0	0	1

security-level attained against SAT attack is $k - \lceil \log_2 \binom{k}{h} \rceil$. The resilience against approximate attack in terms of the output error rate is $\frac{\binom{k}{h}}{2^k}$.

2) SFLL-Flex: While SFLL-HD is suited for protecting a large number of arbitrary PIPs, SFLL-flex is more suited for applications where a designer wants to protect a few PIPs of his choice [30]. SFLL-flex takes the user-defined initial cubes (PIPs with don't care bits) as input and compresses them to into the final cubes. Functionality-stripping is then affected on the final cubes using simulated annealing. The restore unit consists of the final cubes and the *flip vectors* stored in a LUT.

3) SFLL-Fault: SFLL-HD and SFLL-Flex utilize existing logic synthesis tools to perform functionality-stripping. The FSC is obtained by hard-coding the secret key/final cubes in the circuit, which is susceptible to removal attacks [99]. SFLL-fault uses fault injection to affect the functionality-strip operation, essentially subtracting logic from the original circuit without leaving any traces in the netlist [99]. The other SFLL variants operate by adding logic to the original circuit, which the current synthesis tools fail to merge with the original circuit.

C. Cyclic logic locking

A new approach to SAT-resiliency based on graph-theory, known as cyclic logic locking, is being heavily investigated recently [50], [75]–[80]. Cyclic locking thwarts SAT-based adversary by exploiting the directed acyclic graph (DAG) representation of the combinational circuits [38]. The first

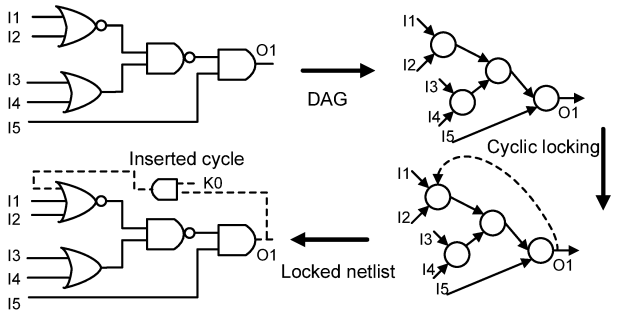


Fig. 13. By inserting backward edges in the DAG, CycLock introduces cycles in a circuit, rendering the SAT attack infeasible [75].

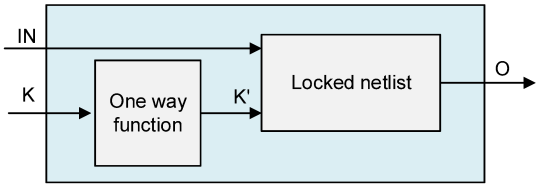


Fig. 14. An illustration of OWF-lock [49]. The probability of computing K from K' is negligible.

cyclic locking technique, which we refer to as CycLock [75], inserts key-driven cycles into a circuit to render its graph cyclic. This is illustrated in Fig. 13. The basic assumption is that cycles are hard to resolve for SAT solvers. However, CycLock can be easily circumvented by the CycSAT attack and its improved version BeSAT [78] (see Section VI-C). CycSAT adds additional clauses to the SAT formula to return a key that renders the circuit acyclic [50]. Another similar attack breaks CycLock by pruning out keys that form cycles [100].

A wave of CycSAT-resistant techniques such as Cross-Lock [80], SRCLock [79], and others [76], [77] have been proposed. Cross-Lock integrates cycles with a cross-bar of programmable vias [80]. SRCLock introduces feedback paths in a way that renders the number of cycles exponential in the number of feedback paths [79]. It forces CycSAT to add an exponential number of clauses to break all the cycles. Similarly [76] and [77] also increase the number of cyclic configurations exponentially in the key size, by utilizing different reconfigurable structures.

D. One-way function-based locking (OWF-Lock)

Resilience against the SAT attack can also be achieved by utilizing structures that are hard to resolve for an SAT solver [49], [101]. As shown in Fig. 14, OWF-lock synthesizes a one-way function along with RLL/FLL-locked circuit [49]. The one-wayness of the OWF renders it computationally infeasible to determine K from O or even K'. The diffusion property leads to a high output corruptibility. OWF-Lock, however, relies on the synthesis of the OWF and the locked netlist to be secure against removal attacks.

VI. Attacks on Post-SAT Logic Locking

Attacks have also been launched on the SAT-attack-resilient techniques discussed in the previous section. The structural and functional properties of point-functions make them vulnerable to removal attacks, which are presented in Section VI-A, as well as approximate attacks, which are presented in Section VI-B. Cyclic locking techniques are vulnerable to CycSAT and its derivative attacks, as elaborated in Section VI-C. In Section VI-D, we introduce the SMT attack, which can incorporate the SAT attack and its different variants. In the context of side-channels, only power-side channel has been exploited for attacks; we discuss these attacks in Section VI-F.

A. Removal attacks

A removal attack identifies the protection logic and removes it from the locked netlist, recovering the original design [40], [41]. As discussed earlier, removal attacks are effective on defenses that retain the original design and the protection logic separable. The existing removal attacks focus on removing point-function implementations.

The signal probability skew (SPS) attack analyzes the bias in the signal probabilities to locate point-functions whose output is highly skewed towards 0 [41]. The AppSAT-guided-removal (AGR) attack uses AppSAT to differentiate the Anti-SAT key-bits from the obfuscation key-bits, followed by structural post-processing to remove obfuscated Anti-SAT block [40]. The sensitization-guided SAT (SGS) attack circumvents the protection in ATD by identifying the dummy trees [40]. The Bypass attack builds a functionally-correct netlist by identifying input patterns for which a randomly selected key generates incorrect output. A bypass circuit is added to restore the circuit output [10].

A recent attack, referred to as the FALL attack [102], combines structural and functional analysis to locate and remove the comparator and HD units inside SFLL-HD circuits, recovering the original circuit. The attack is applicable when the cube-stripper (i.e., the HD-unit with a hard-coded secret key) is left as is in the circuit during logic synthesis. Apart from the structural analysis, the FALL attack relies on the unateness and HD properties of the HD-unit to locate it.

B. Approximate attacks

While the SAT attack terminates only upon retrieving the correct key, the AppSAT [8] and Double-DIP [9] attacks terminate earlier returning an approximate key, which results in an approximate netlist. The termination criteria for AppSAT is dictated by an error rate set by the attacker. Double-DIP terminates when it can no longer find *2-DIPs* that eliminate at least two incorrect keys. Both attacks mainly target compound logic locking techniques, wherein a high-corruptibility locking technique (e.g., FLL) is integrated with a low-corruptibility technique (e.g., SARLock). The approximate attacks reduce a compound logic locking technique (e.g. SARLock+FLL) to its low-corruptibility constituent (i.e., SARLock).

C. CycSAT

The CycSAT attack targets cyclic logic locking [50]. It modifies the SAT attack to operate on a cyclic graph. The attack has a pre-processing step that computes additional clauses to be added to the SAT formula. These clauses help extract a key that renders the retrieved circuit acyclic. Alternatively, the retrieved circuit may be cyclic, but none of the cycles are sensitized under the extracted key. The behavioral SAT attack extends CycSAT to consider the functional behavior of the locked circuit. This modification helps tackle the issue of statefulness, where the locked circuit may produce different output for a given key and input combination [78].

D. Satisfiability modulo theory (SMT) attack

Unlike SAT solvers, SMT solvers can handle non-Boolean variables, which makes the SMT attack a superset of the SAT attack [51]. SMTs targets a more general class of constraint satisfaction problems, such as decision problems with constraints specified using first-order theories that include real numbers, arithmetic, and bit vectors. The SMT attack can realize the SAT attack and its different variants, such as the approximate attacks. It also assimilates an independently developed version of the TimingSAT attack [103] that breaks delay locking (see Section VIII-D).

E. Attacks against SFLL

Certain instances of SFLL, such as SFLL-HD and SFLL-flex, perform the functionality-stripping operation by adding logic to the original function. The added logic is expected to be blended into the rest of the design, hiding any traces for a RE attack; yet, the use of traditional security-agnostic synthesis tools still leave traces behind. Consequently, attacks can analyze the netlist structure to trace signals of certain characteristics, identify the added logic and remove it to recover the original functionality/design, circumventing these defenses [102], [104], [105]. To withstand such attacks, security-aware synthesis tool should be developed and used to affect the functionality-stripping operation without leaving any traces in the locked netlist.

Alternatively, the functionality-stripping operation can be performed by truly *removing* logic from the original function in line with the notion of stripping; a recent version of SFLL, SFLL-fault [99], [106], injects faults to the original function for this purpose. This way, not only no traces are left but also the attacker is forced to reconstruct the removed logic that could be implementing one of a very large number (2^{2^k}) of functions. Here, k is the size of the key.

F. Power side-channel attacks

The differential power analysis (DPA) attack is a side-channel attack that utilizes the correlation between power consumption and the key value to extract the secret key [107]. The DPA attack on logic locking can be launched in a divide-and-conquer fashion by targeting individual logic cones [36], [81]. The attack applies several input patterns to a functional IC and records the power traces and as well as the circuit

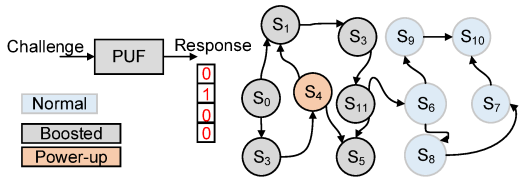


Fig. 15. An illustration of BFSM [82]. Upon power-up, the PUF response to the given challenge is 0100, making S_4 as the power-up state. To traverse from S_4 to the initial functional state S_6, the FSM may traverse the following path: $S_4 \rightarrow S_1 \rightarrow S_3 \rightarrow S_{11} \rightarrow S_6$.

output. It classifies power traces into a zero-bin and one-bin, based on the locked circuit output for a different key. The difference of the power values in two bins exhibits a spike for the correct key.

The template-based power analysis attack also deciphers the key through power measurements. Based on the location of key-gates at different logic depths, it enables the attacker to unlock the circuit functionality level-by-level [42].

VII. Sequential Logic Locking

Having completed the discussion on the combinational logic locking techniques, we turn our attention to the sequential logic locking attacks and defenses. With only a handful of attacks and defenses, a meaningful classification of the sequential locking techniques could not be developed. Most of these techniques add *obfuscated* states to an FSM (see Section VII-A– VII-C). These techniques are vulnerable to FSM reconstruction and BMC-based attacks, which are discussed in Section VII-D and Section VII-E, respectively. Mode-based obfuscation, presented in Section VII-G, protects a circuit by locking it in one of the non-functional modes.

A. Boosted FSM (BFSM)

In BFSM, an exponentially large (in the key size) number of obfuscated states are introduced [19]. The state upon power-up is determined by applying a challenge to an on-chip PUF and loading the PUF responses into state FFs. The probability that the FSM is initialized into an obfuscated state (and not a functional state) is very high since the number of obfuscated states is much larger than the number of functional states. To unlock the FSM, i.e., traverse to the initial functional state S_{in}, an input sequence must be provided that initiates the required state transitions. With the full knowledge of the BFSM, a designer can construct the input sequence that takes the FSM to S_{in} from any of the power-up states. An attacker without the knowledge of the BFSM cannot compute the desired sequence.

Example. In the BFSM shown in Fig. 15, the PUF response is 0100, indicating that the power-up state is S_4. To traverse to the first functional state S_6, the FSM may follow the path $S_4 \rightarrow S_1 \rightarrow S_3 \rightarrow S_{11} \rightarrow S_6$.

The brute-force attack effort required to break BFSM is exponentially high in the number of state elements. BFSM can also circumvent RE asthe extraction of an FSM is computationally intractable. The security of BFSM against brute-force attacks can further be enhanced by incorporating the *blackhole* states from where there return to the functional states [19].

B. HARPOON

Similar to BFSM, HARPOON also introduces obfuscated states; however, the number of added states is much smaller [43]. The locked design can be considered to have two FSMs: the obfuscated FSM and the functional FSM. Upon powering up, the FSM enters a pre-defined reset state S_r. The circuit is then in the obfuscated mode and produces incorrect outputs. Only upon supplying the correct input sequence K_{sec}, the FSM transitions to the functional state S_{in}. All subsequent transitions are between the functional states. An attacker without access to K_{sec} cannot make the circuit functional.

Example. Consider the FSM in Fig. 16(a). The functional FSM comprises the states $\{S_5, \cdots, S_{10}\}$; $S_{in} = S_5$ is the initial functional state. The obfuscated FSM comprises the states $\{S_R, S_1, \cdots, S_4\}$. To reach the state S_5, the obfuscated FSM may follow either the path: $S_R \rightarrow S_1 \rightarrow S_2 \rightarrow S_5$ or the path: $S_R \rightarrow S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_5$.

C. FSM convolution

In HARPOON, there exists a distinct boundary between the functional and the obfuscated states, which an attacker can exploit. Multiple locking techniques increase the resilience by "convoluting" the two FSMs. *State interlocking* blurs this FSM demarcation by increasing the number of transitions among the two sets of states [82]. Even upon applying an incorrect key, the FSM may transition from S_R to a functional state. To keep track of the state transitions, the transition logic embeds a virtual "code-word" that dictates both subsequent state transitions and circuit output.

Example. Let us consider S_6 as the current state in Fig. 16(b). From this state, the FSM can transition to either S_7 or S_9, depending on the path it took to enter S_6, which is encoded in the code-word. For a k-bit code-word, N state elements in the obfuscated FSM, and q states in the original (unobfuscated) FSM, the number of brute-force attempt for state interlocking is estimated to be $\sum_{k=1}^{N+k} \binom{N+k}{k} \times q$. A similar interlocking approach is proposed in [55].

State entanglement thwarts the FSM separation attacks (see Section VII-D) by further obliterating the boundary between the two sets of states [44]. As shown in Fig. 16(c), there is no clear distinction between obfuscated and functional states.

Dynamic state deflection (DSD) introduces multiple black-hole FSMs to hide the boundary between the obfuscated and the functional states. It ensures that the circuit deflects *dynamically* to different blackhole FSMs upon application of different incorrect keys [83]. In Fig. 16(d), there are three black-hole FSMs with no return to a functional state.

While the aforementioned sequential locking techniques modify the topology of the FSM, hidden transitions can be introduced into an FSM based on the physical parameters such as the operating frequency, coupling capacitance, or temporal profile of the clock [108]. For example, a glitch can be leveraged to enact a state transition at a particular frequency; such transitions cannot be recovered from the FSM structure.

D. FSM reconstruction and separation attacks

An essential step in circumventing sequential locking is to extract the high-level FSM of from the gate-level description of

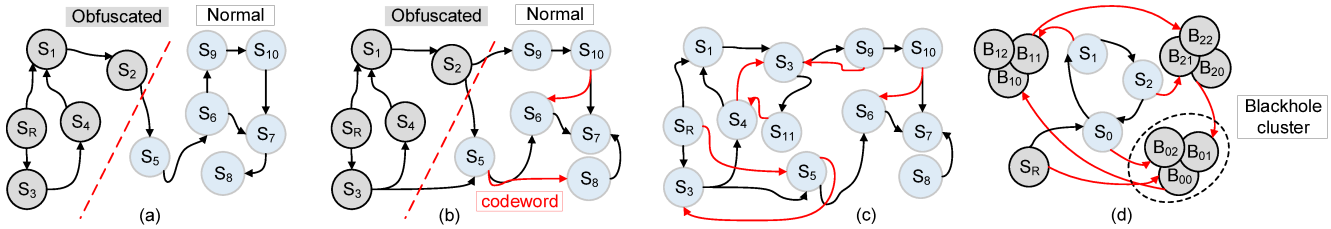


Fig. 16. HARPOON and its derivative FSM logic locking techniques. a) HARPOON introduces an obfuscated FSM [43]. b) State interlocking adds extra transitions between the normal and obfuscated FSM [82]. c) State entanglement further obliterates the boundary between normal and obfuscated FSMs [44]. d) DSD introduces multiple blackholes FSMs [83].

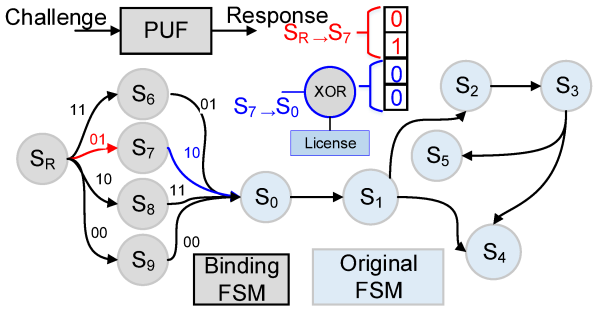


Fig. 17. PUF-FSM binding [109]. The PUF responses lock the FSM in a "binding" state. Only the supplying the license, the FSM transitions into the normal mode. Each chip/FPGA can have a unique license.

a netlist. Multiple research efforts [54]–[56] utilize heuristics to identify the state registers, build Boolean expressions for the state transitions, and extract the FSM. These FSM reconstruction steps (initially postulated in REFSM [54]) are the backbone of three independently-developed FSM separation attacks [54]–[56] that use Tarjan's SCC algorithm to bifurcate obfuscated and functional states.

The three attacks differ in terms of the heuristics utilized. While [54] and [55] focus on retrieving the correct input sequence, [56] notes that the SCC algorithm may not scale well to FSMs with a large number of states. Thus, an *initial state patching* procedure is developed that can bypass the obfuscated FSM by resetting the state FFs to S_{in}. The value of S_{in} is determined either by SCC or by analyzing the output function of a circuit.

Hardware nanomites defend against the SCC-based attacks by leveraging partial dynamic reconfiguration of FPGAs [56]. The FSM is partitioned into sub-FSMs, which are withheld and loaded dynamically onto the FPGA only at runtime. Eliminating the need to load the entire FSM onto the FPGA at once helps thwart attacks that analyze the FSM structure.

E. Bounded model checking (BMC)-based attacks

In BMC-based attacks, the sequential circuit is unrolled p times and model-checking queries are applied [57], [58]. An attack terminates when a unique key is recovered or when the locked circuit becomes combinational equivalent of the original circuit. The runtime of these attacks can be improved by combining iterative SAT solving with the dynamic simplification of complex conditions (SAT clauses) that pile-up over successive attack iterations [58]

F. PUF-FSM binding

The PUF-FSM binding scheme, which targets FPGAs, is similar to BFSM. The scheme uses a PUF to lock each FPGA instance with a unique key [109]. It adds several layers of *binding* states. Upon power-up, the FSM initializes to the state S_R, from where it traverses to one of the binding states, based upon the PUF responses. In Fig. 17, the PUF response 01 induces the transition $S_R \rightarrow S_7$. For transitioning to the functional state S_0, the correct input sequence a.k.a. *license* for the chip must be provided. The license is computed by XORing the PUF response $R = 00$ with the FSM key $K = 10$ that is required for the transition $S_7 \rightarrow S_0$. In our example, the chip license is $10 \oplus 00 = 10$. This scheme has an added advantage over the traditional binding methods: It does not require storing the key on an FPGA. By using only a PUF, the scheme can enable pay-per-use licensing mechanism.

G. Mode-based functional obfuscation

Several classes of circuits can be reconfigured to operate in different modes. For example, certain implementations of DSP circuits, such as digital filters, allow the same circuit to behave as a third-order filter, a sixth-order, or a ninth-order filter in a time-multiplexed manner [48]. In certain applications, not all the modes may be desired; the undesired modes thus imply an incorrect behavior. Mode-based obfuscation can be realized by implementing the "switches" used in the digital filters in a programmable fashion, e.g., by replacing each switch with a key-controlled MUX that provides the correct configuration data at runtime.

Fig. 18 elaborates on mode-based obfuscation [48]. The key has two parts: the initialization key and the configuration key. The obfuscating FSM is the same as in HARPOON [43]; only upon entering the correct initialization key, the FSM enters a functional state. The reconfigurator and the ring counter control the function (mode) of the DSP circuit by generating the configuration data for each configurable switch in the design based on the configuration key. An incorrect configuration key leads to non-meaningful modes.

Mode-based obfuscation can be static or dynamic in nature. In static obfuscation, the circuit output depends solely upon the key value, which is fixed and loaded at the time of activation of the chip. In dynamic obfuscation, however, the circuit output is dictated by the key, the mode, and the trigger signal, as depicted in Fig. 19. When the trigger circuit is activated, the chip produces an incorrect output [47]. Upon applying

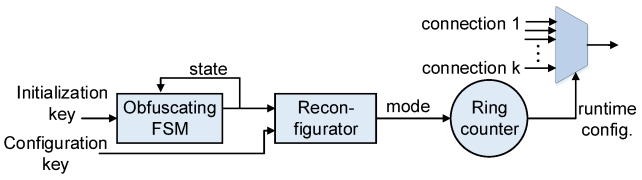


Fig. 18. Mode-based static obfuscation for digital signal processing (DSP) circuits [48]. The obfuscating FSM is controlled by the initialization key. Upon entering the normal mode, the reconfigurator generates control signals for a ring counter, which produces the correct configuration data for each switch.

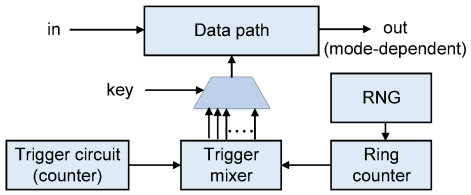


Fig. 19. Mode-based dynamic obfuscation [47]. The initial value to the ring counter is provided by an on-chip random number generator (RNG). The trigger signal is asserted rare conditions in a periodic manner.

random keys to a system, a time-varying/dynamic response is observed. The occasionally incorrect circuit output, obtained upon activation of the trigger, helps to thwart SAT and other oracle-guided attacks. The estimated lower bound on the number of brute-force attempts is given as $L \times K \times 2^K$. Here K and L are the key size and the activation period, respectively. The activation period denotes the number of cycles required to complete an operation, e.g., computation of the FFT. The advantage of dynamic obfuscation over static is increased security with even smaller key sizes. For example, the brute-force attack on dynamic obfuscation requires 3.3×10^{24} cycles for $K = 32$, compared to the 2.1×10^9 cycles required for static obfuscation [47].

VIII. Beyond Functionality Locking

While logic locking techniques have traditionally focused on protecting the circuit functionality, researchers have found novel applications for it in the last few years. This section discusses such applications of logic locking in newer contexts, such as analog circuits and microfluidic biochips. There also exist non-digital ways of locking a circuit, which not only protect the functionality but also the parametric behavior, such as performance and timing. We also present innovative use-cases, where a locked digital circuit protects a larger system.

A. Logic locking in actual silicon

SFLL-HD is the first logic locking technique to be demonstrated in actual silicon. The chip contains an ARM Cortex M0-based microcontroller, whose program counter (PC) is locked with FLL+SFLL-HD^0. Hence, upon applying incorrect key values, a program is halted [30]. Another chip implements multiple locking techniques on a biomedical processor, which predicts the heart attack events from the electrocardiogram signal [6]. This chip also generates its own unique key from the electrocardiogram signal.

B. Logic locking to defeat hardware Trojans (HT)

Insertion of HT by manipulation of the netlist without affecting the main functionality of the design has been of major concern to fabless semiconductor companies as well as government agencies. Logic locking can help defend against such threats by making it harder for the attacker to identify the covert locations for Trojan insertion. For example, [110] proposes the introduction of AND/OR gates to judiciously alter signal probabilities in a netlist, resulting in the reduction in the number of signals with low *controllability*. However, if inserted successfully, the Trojan circuits are triggered in very rare conditions making the HT detection process within a limited time an extremely challenging problem. In [111], an approach which leverages logic locking to facilitate path-delay based HT detection has been outlined. The proposed technique masks the circuit logic (by introducing fake short paths) such that every net in the modified design belongs to a short-enough path so that path-delay analysis can be successfully performed.

C. Performance locking

In certain Intel processors such as the *Clarkdale* and *Sandy Bridge* series, additional cache or hyper-threading features can be activated with extra payment [112], [113]. Recently, logic locking techniques have been applied to offer similar performance-locking features, albeit with provable security guarantees [45]. The chip is designed to deliver both high and low performances, as dictated by the key. Fig. 20 depicts the business and threat model of performance locking.

To degrade performance, additional LOCK states are inserted into the FSM. There is no useful computation carried out in these states, leading to a reduction in performance [45]. Superior performance can be unlocked with the correct key, which bypasses the LOCK states. Performance-locking can be implemented using the functionality-strip and restore operations. For example, the stall signal in a pipelined processor can be perturbed for a performance-degrading input pattern (PDIP). For an incorrect key (Key $\neq$ PDIP), the stall signal may be asserted, locking the circuit in LOCK states for N cycles and degrading performance, as shown in Fig. 21. The correct key restores the perturbation and unlocks superior performance.

D. Delay-based logic locking (DLL)

DLL renders a circuit's functionality as well as the timing dependent on the key [114]. The correct key recovers the original functionality as well as the timing profile that satisfies the pre-defined timing constraints. The delay of a circuit is manipulated using a tunable delay key-gate (TDK), which comprises of conventional key-gate (e.g., XOR/XNOR) and a tunable delay buffer [115]. Each TDK has two key inputs. The functional-key k_1 dictates the behavior of XOR key-gate, whereas, the delay-key k_2 determines whether the TDK delay is d_0 or d_1.

A new attack called TimingSAT [103] has been proposed to unlock DLL circuits. The attack models the timing characteristics of various gates present in the design as Boolean functions to build timing profile embedded SAT formulations. The proposed attack operates in two stages. In the first stage, the adversary finds the functional-key using conventional SAT attack. In the second stage, he/she utilizes timing information

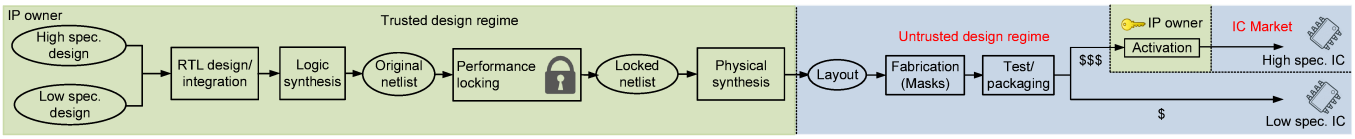
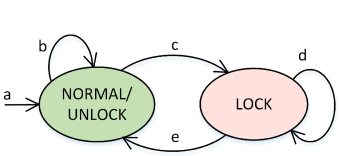


Fig. 20. Performance locking allows premium users to unlock superior performance, while regular ICs offer inferior performance [45].



Transition	Input	Execution
a	Reset = 1	Normal
b	Key = PDIP	Normal
c	Key ≠ PDIP	Lock if (IN = Key/ IN = PDIP)
d	Count ≤ N	Lock
e	Count = 0	Normal

Fig. 21. The state transition graph of the inserted FSM. N indicates the number of stall cycles. Normal execution indicates the design is in *NORMAL/UNLOCK* state. Lock indicates the design is in *LOCK* state [45].

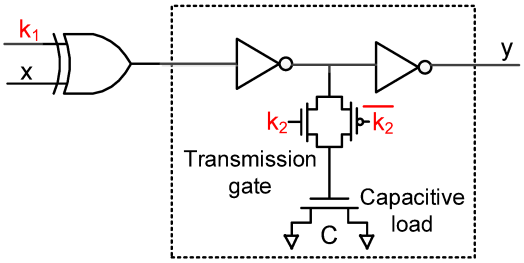


Fig. 22. Tunable delay key-gate (TDK) [114].

embedded in the locked netlist to retrieve the correct delay-key. Similar SAT-based attack has also been demonstrated against camouflaged netlists protected using timing-based parametric camouflaging [116]. The SMT attack discussed in Section VI-D can also circumvent DLL.

E. Graphics-processing unit (GPU) obfuscation

Apart from being applied to the traditional benchmarks and processors, logic locking has also been studied specifically in the context of GPUs. The objective is to lock the performance of a multi-cycle GPU, by locking its memory controller that implements the cache block replacement policy [117]. A locked controller design severely impacts the hit rate of cache memory. In [117], it was shown that naïve implementation of a locked GPU remains susceptible to an oracle-less approximate attack. This attack translates the multi-cycle GPU core netlist to a functionally-equivalent single-cycle netlist and utilizes the GPU instruction set architecture to compute DIPs. This attack can be thwarted by modifying the cache block replacement policy such that the locked cache judiciously generates cache hits for the incorrect keys. This degrades the performance of the applications running on the locked GPU and thwarts approximate attacks [117].

F. Analog and mixed-signal (AMS) locking

The relatively low transistor count and distinct layout patterns in analog circuits make it easier for malicious entities to reverse engineer the layout and pirate the circuit. Recent

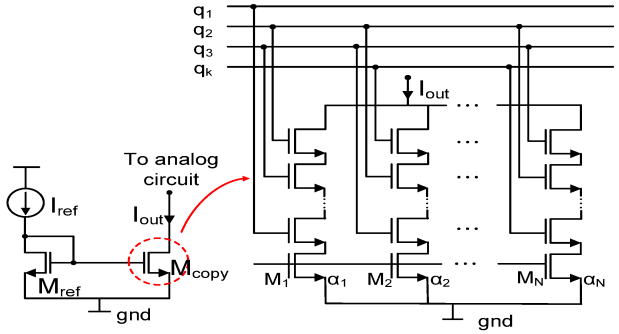


Fig. 23. Configurable current mirrors to thwart IP piracy of analog circuits [119].

research deploys logic locking to prevent overproduction of AMS ICs [46]. The idea is that only a correct key makes the AMS circuit to operate as per the specifications.

1) Analog circuit locking: The specifications of an analog circuit such as the center frequency, bandwidth, oscillation frequency, etc. depend on the precise value of the bias current as well as the width-to-length ratio (W/L) of the transistors. In [118], a single transistor is replaced with a series of key-controlled transistors to lock analog circuits such as differential amplifiers and bandpass filters. An SMT-based combinational locking technique is proposed in [119]. The current mirror, providing the required bias current I_{out} is made configurable. As illustrated in Fig. 23, the copying transistor M_{copy} in the current mirror is replaced by a transistor matrix of size $(R \times N)$ with each of the transistor gates connected to one of the k key lines. I_{out} is as the sum of currents in individual branches. The SMT formulation is such that for a specific R, N, and k, there is only one key $Q = (q_1 q_2 \cdots q_k)$ that produces the desired I_{out}. For all other key combinations, I_{out} will either be above the specified upper-bound or below the lower-bound set by the designer, making the circuit non-functional.

2) AMS circuit locking: As shown in Fig. 24, the digital section of an AMS circuit can be locked using SFLL [30]. Only on applying the correct key, the optimizer tunes the passive components in the bandpass filter (e.g., resistors (R) and capacitors (C)) to compensate for the effect of process variations. Hence, the filter operates within the desired specification. For all incorrect keys, the specifications are violated. For cost-effective locking, a sensitivity analysis is conducted to determine the most influential tuning knobs (R and C's) to be locked. This delivers maximal performance degradation with minimal implementation overhead.

MixLock is another lightweight and non-intrusive approach for locking AMS circuits [120]. Here, the digital part of the circuit is locked using existing locking techniques such that the attained security level is maximized from both analog and digital perspectives. The security level in the digital domain

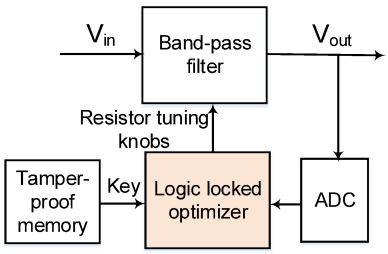


Fig. 24. Logic locking of the BPF circuit. Only on applying the right key, the optimizer sets the resistor value by considering the effect of process variation [46].

is measured as the resiliency against SAT attack. The security level in the analog domain is quantified as the error rate, i.e., the percentage of incorrect keys that result in the violation of the performance specifications, e.g., the signal-to-noise ratio.

G. Locking digital microfluidic biochips (DMFBs)

Another interesting area where logic locking has found application is the protection of biochemical assay protocols in DMFBs. DMFBs are also vulnerable to supply chain attacks, such as the assay manipulation attacks, which can jeopardize the clinical diagnostics [121]. By locking a DMFB assay with microfluidic multiplexers, the DMFB access can be restricted to the authorized users in possession of the correct key [122].

H. Locking at different abstraction levels

Thus far, we have focused on gate-level and register-transfer level (RTL) locking. Locking can also be extended to other abstraction levels such as high-level synthesis and transistor-level.

1) High-level synthesis (HLS)-based locking: Recently, logic locking has been demonstrated using high-level synthesis tools [123], albeit under a restrictive (oracle-less) threat model. The only untrusted entity is a foundry that does not has access to the functional IC. HLS can lock the arithmetic operations, constant values, and the control flow of a design to protect against oracle-less attack. Such a threat model is useful in scenarios where the fabricated chips are used by a few clients and are not available in the IC market.

2) Locking at the transistor level: Locking can also be effected at the lowest level of abstraction, i.e., the transistor level.In TRAP [124], individual transistors are locked by introducing transistor-specific programmable fabric, which is programmed post-fabrication. Such a low-level locking scheme enables multi-faceted locking of individual gates, on-chip interconnect, and even FSMs. TRAP also allows a netlist to be updated on a dynamic basis, thereby hampering the extraction of a "fixed" locked netlist, which is required by all existing attacks [124].

I. Scan locking

Existing oracle-guided attacks assume scan chain access. This access allows a sequential circuit to be treated as a combinational one by converting the scan-cell outputs (inputs) to pseudo-primary inputs (outputs). Scan locking aims at locking the scan chain responses, effectively revoking oracle access. One approach is to modify the scan cell architecture so that it can hold previous values on-demand. The test protocol is such that key values are not captured into the scan-cells during the test mode, debilitating oracle-guided attacks [125]. A similar approach combines functional and scan locking, along with dynamically generated keys (using a linear feedback shift register) to remotely test locked chips before activation [126].

Another scan locking approach referred to as Encrypt-Flip-Flip (EFF) inserts key controlled MUXes at the outputs of selected scan-cells [127]. Each MUX selects either Q or $\overline{Q}$ output of the scan-cell as dictated by the key value. During scan-in and scan-out, the values loaded into the scan-cells and the responses observed are corrupted by the key. This operation prevents an attacker from loading patterns of his/her choice or inferring the data captured into scan-cells. However, EFF can be circumvented using the ScanSAT attack [128], which unrolls the locked scan-chain, modeling it as a combinational circuit. The SAT attack can then be applied on this unrolled circuit even with obfuscated responses. The BMC-based attacks [57] also challenge the effectiveness of scan locking by extracting secret keys without scan access.

IX. Discussion

In this section, we discuss several important aspects of logic locking. Section IX-A sums up the logic locking research presented in the paper, highlighting the relationship between different classes of attacks and defenses. Section IX-B summarizes the major challenges faced by logic locking research. Section IX-C spotlights future research directions. Section IX-D points to the related survey papers and other relevant resources in the field of logic locking.

A. An overview of attacks and defenses

The handful of sequential locking techniques focus on introducing obfuscated states, which exhibit susceptibility to FSM separation and BMC attacks [57], [58]. We, therefore, solely elaborate on the relationship among different classes of combinational locking attacks and defenses in Table V. The table also depicts the threat model of each attack and defense. An attacker may have access to a locked netlist (○), a functional IC (●), test data (⋆), or a combination of these items. The most common threat model (represented as ○●) assumes that an attacker has both a locked netlist and a functional IC. The oracle-less attacks, e.g., SGS [40] and de-synthesis [53], do not require a functional IC (○). HLS [123] is the only defense with this threat model. The threat model for test-data attacks require access to a locked netlist and test data (○⋆).

All pre-SAT locking techniques are vulnerable to the SAT attack as well as various side-channel attacks. Each point-function-based locking technique can be broken with one or more removal attacks. For the same techniques, an approximate key (represented as ≈) may be recovered using the approximate attacks. SFLL-HD exhibits strong resilience against all traditional attacks but is rendered insecure due to current logic synthesis techniques. The recent FALL attack [102] exploits this vulnerability to break SFLL-HD. SFLL-fault

TABLE V

A SUMMARY OF COMBINATIONAL LOGIC LOCKING ATTACKS AND DEFENSES WITH THE CORRESPONDING THREAT MODELS. TM STANDS FOR THREAT MODEL. ○ DENOTES A LOCKED NETLIST. ● DENOTES A FUNCTIONAL IC/ORACLE. ⋆ DENOTES TEST-DATA. ✓ DENOTES THAT A DEFENSE IS SECURE AGAINST THE ATTACK. ≈ DENOTES APPROXIMATE KEY RECOVERY. NA DENOTES THAT THE ATTACK IS NOT APPLICABLE UNDER THE DEFENSE TM.

			Algorithmic					**Removal**					**Side-channel**					**Approx.**		
	Attack / **Defense**		Sens. [2]	LCA [88]	SAT [38]	CycSAT [50]	SMT [51]	SPS [41]	AGR [40]	SGS [40]	Bypass [10]	FALL [102]	TDM [33]	Hill [37]	Power [36], [42]	SAIL [52]	De-syn [53]	AppSAT [8]	2-DIP [9]	TimingSAT [103]
		TM	○●	○●	○●	○●	○●	○●	○	○●	○●	○●	○⋆	○⋆	○●	○●	○	○●	○●	○●
Pre-SAT	RLL [1], [31]	○●	×	×	×	×	×	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
	LUT-based [34]	○●	×	×	×	×	×	✓	✓	✓	✓	✓	×	×	×	✓	✓	×	×	×
	FLL [35]	○●	×	×	×	×	×	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
	WLL [91]	○●	×	×	×	×	×	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
	Centrality [90]	○●	×	×	×	×	×	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
	SLL [2], [49]	○●	✓	✓	×	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×	×
	KLL [93]	○●	✓	✓	×	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×	×
Point	SARLock [4]	○●	✓	✓	✓	✓	≈	×	×	×	×	✓	✓	✓	✓	✓	✓	≈	≈	✓
	Anti-SAT [41]	○●	✓	✓	✓	✓	≈	×	×	×	×	✓	✓	✓	✓	✓	✓	≈	✓	✓
	ATD [39]	○●	✓	✓	✓	✓	≈	✓	✓	×	×	✓	✓	✓	✓	✓	✓	≈	≈	✓
	DTL [74]	○●	✓	✓	✓	✓	✓	×	×	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Compound [4], [5]	○●	✓	✓	✓	✓	≈	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	≈	≈	✓
SFLL	SFLL-HD [7]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
	SFLL-Flex [30]	○●	✓	✓	✓	✓	≈	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	≈	✓	✓
	SFLL-Fault [99]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cyclic	CycLock [75]	○●	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Cross-Lock [80]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	SRCLock [79]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Others	OWF-Lock [49]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	DLL [114]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
	TRAP [124]	○●	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	HLS [123]	○	NA	NA	NA	NA	NA	NA	✓	NA	NA	NA	NA	NA	NA	NA	✓	NA	NA	NA

is resilient to FALL, but as it protects only a handful of input patterns, it is prone to the approximate attacks. As CycLock was quickly broken using CycSAT [77], the other variants of cyclic locking that are relatively recent must be subjected to further scrutiny. OWF-Lock is not vulnerable to any known attack, however, its implementation overhead can be prohibitive ($10\times$ $-20\times$) [49]. The oracle-less threat model assumed by HLS renders most attacks inapplicable to it [123]. However, most practical scenarios violate this assumption.

B. Challenges and roadblocks

The discussion in the previous section indicates that both combinational and sequential logic locking techniques exhibit specific weaknesses that impede the widespread adoption of logic locking. Logic locking is still in its infancy compared to other fields, such as PUFs and side-channel analysis. Accordingly, the precise notions of security in logic locking and the relevant metrics are still a work in progress [30], [120], [129]. Most defenses, e.g., point-function-based and cyclic locking, address only specific attacks and are consequently broken by the next generation of more sophisticated attacks [40], [50]. Often there is a trade-off between resilience against different attacks. Similar to the case of many cryptographic algorithms, provably-secure locking algorithms (e.g., SFLL-HD or Anti-SAT) may exhibit security vulnerabilities when it comes to the implementation [41], [102]. Cost-effectiveness and security have always been at crossroads, which also applies to logic locking. Many existing techniques protect only selected parts of a circuit. Nevertheless, this emerging field has seen significant growth since its inception, and there is potential for further research to find novel solutions to the problems mentioned above.

C. Future research directions

In the light of the discussion on challenges faced by logic locking, we envision the following future research directions:

1) Integrated combinational+sequential locking: The sequential and combinational logic locking may be integrated for increased security and cost-effective implementation. The BMC attacks are the first attempt in this regards [57], [58]. This line of research can also help develop a common threat model that unifies the assumptions about scan access.

2) Leveraging program obfuscation: A much-needed paradigm shift in the area of logic locking is to curtail the current practice of attack-specific defenses. Logic locking has many similarities with the program obfuscation problem [130]. Accordingly, locking can benefit from the developments in the field of program obfuscation. The well-defined notions of security, such as indistinguishable obfuscation, as well as results on the (im)possibility of obfuscating different classes of Boolean functions [131], [132] can be used in the context of locking.

3) Leveraging EDA: One of the major impediments to the adoption of logic locking techniques is the overhead incurred by these techniques. The power of EDA tools can be harnessed to build lightweight implementations, making logic locking more desirable for IC designers. Another challenge is to ensure the security properties offered by a logic locking

technique remains intact post-synthesis. For this purpose, existing verification tools need to be reinforced with security-relevant properties. Newer logic synthesis techniques should be developed to offer provable security against attacks that rely on the structure and functionality of a netlist. Finally, logic locking techniques are now getting morphed to secure system-wide applications [106], [117], [133]. Since EDA tools can now operate at different abstraction levels with a global perspective of the system and design goals, they can be re-purposed to ensure the security of system-wide applications with logic locking as security primitive.

4) Business-oriented security: In Section VIII, we discussed novel applications of logic locking beyond functionality locking. For adoption of logic locking by the broader industry, the business model must be as lucrative as the security advantages obtained under a given threat model. The additional effort invested in locking, activating ICs, and the expensive silicon footprint dedicated to locking circuitry must be compensated well in financial terms. Digital rights management using sequential locking [20] and performance locking [45] are the earliest attempts in addressing the economic benefits. There is a strong need for discovering further applications of logic locking that make sense from a business standpoint.

D. Related survey and resources

The increasing interest in logic locking and circuit obfuscation research have led to a number of survey papers and books in this area. Some are listed below for interested readers.

1) A recent (2018) survey on the use of formal methods in IP protection is provided in [134].
2) A shorter survey on combinational logic locking is performed in [135].
3) A survey on supply chain risk and mitigation approaches is given in [136].
4) A larger set of IP protection methods are discussed in [137].
5) For a primer on hardware security problems in general, readers may refer to [12]
6) A detailed introduction to DfTr techniques including logic locking, camouflaging, and split manufacturing, up to 2014, is given in [138].
7) There are several books highlighting different obfuscation techniques [139], [140], [141]. [141] sheds lights on the security primitives utilized in the context of IP protection, and [142] emphasize IP trust and validation methods.

X. Conclusion

Logic locking started as a solution to thwart piracy and RE in a globalized supply chain. The last decade has seen tremendous progress in this field, ranging from attempts towards developing provably-secure locking techniques to novel applications in microfluidic biochips. The earliest locking techniques locked a circuit to maximize the error rate for incorrect keys. These techniques did not build upon well-established security principles and remain vulnerable to SAT and other attacks. In the post-SAT era, researchers developed mathematically-secure locking techniques that rely on point-functions to derive security properties. While mathematically-secure, these techniques exhibit implementation vulnerabilities and can be broken using newer classes of removal and approximate attacks. A similar trend has been observed more recently for the cyclic locking techniques.

Among various locking techniques, SFLL turns out to be a promising solution that offers provable security against the most classes of attacks; however, SFLL also suffers from implementation vulnerabilities. A variant of SFLL mitigates these vulnerabilities through fault-insertion-based logic synthesis albeit on a small scale, i.e., protecting only a handful of input patterns. Thus, there is a need to develop security-aware logic synthesis to address these implementation issues.

Nevertheless, this emerging field has tremendous potential for further progress. Logic locking can be strengthened to offer increased security at the system-level by leveraging the developments in program obfuscation and logic synthesis. Applications in newer domains such as performance locking, AMS circuits, and microfluidic biochips strengthen the business case for logic locking, leading to its widespread adoption.

References

[1] J. Roy, F. Koushanfar, and I. Markov, "Ending Piracy of Integrated Circuits," *IEEE Computer*, vol. 43, pp. 30–38, 2010.
[2] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," *IEEE/ACM Design Automation Conference*, pp. 83–89, 2012.
[3] J. Rajendran, H. Zhang, C. Zhang, G. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault Analysis-Based Logic Encryption," *IEEE Transactions on Computer*, vol. 64, no. 2, pp. 410–424, 2015.
[4] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 236–241, 2016.
[5] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 127–146, 2016.
[6] M. Yasin, T. Tekeste, H. Saleh, B. Mohammad, O. Sinanoglu, and M. Ismail, "Ultra-Low Power, Secure IoT Platform for Predicting Cardiovascular Diseases," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2624–2637, 2017.
[7] M. Yasin, A. Sengupta, B. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, "What to Lock?: Functional and Parametric Locking," *ACM Great Lakes Symposium on VLSI*, pp. 351–356, 2017.
[8] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 95–100, 2017.
[9] Y. Shen and H. Zhou, "Double DIP: Re-Evaluating Security of Logic Encryption Algorithms," *ACM Great Lakes Symposium on VLSI*, pp. 179–184, 2017.
[10] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 189–210, 2017.
[11] Evertiq, "Top 10 fabless IC design companies Qualcomm in the lead," https://evertiq.com/news/40662, 2016, last accessed on 02/09/19.
[12] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.
[13] 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 IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.
[14] R. Torrance and D. James, "The State-of-the-Art in Semiconductor Reverse Engineering," *IEEE/ACM Design Automation Conference*, pp. 333–338, 2011.
[15] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[16] SEMI, "Innovation is at Risk Losses of up to $4 Billion Annually due to IP Infringement," www.semi.org/en/Issues/IntellectualProperty/ssLINK/P043785, 2008, last accessed on 08/01/18.

[17] A. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," *IEEE/ACM Design Automation Conference*, pp. 776–781, 1998.

[18] D. Kirovski and M. Potkonjak, "Local Watermarks: Methodology and Application to Behavioral Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1277–1283, 2003.

[19] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security," *USENIX Security Symposium*, pp. 291–306, 2007.

[20] 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.

[21] SypherMedia, "Syphermedia library circuit camouflage technology," https://www.insidesecure.com/Products/Silicon-IP/Circuit-Camouflage-Technology, 2017, last accessed on 08/01/18.

[22] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," *ACM SIGSAC Conference on Computer & Communications Security*, pp. 709–720, 2013.

[23] J. Baukus, L. Chow, R. Cocchi, and B. Wang, "Method and Apparatus for Camouflaging a Standard Cell based Integrated Circuit with Micro Circuits and Post Processing," 2012, US Patent no. 20120139582.

[24] J. Baukus, L. Chow, R. Cocchi, P.O., and B. Wang, "Building Block for a Secure CMOS Logic Cell Library," 2012, US Patent no. 8111089.

[25] R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semiconductor Circuits," 2007, US Patent no. 7,195,931.

[26] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," *ACM USENIX Security Symposium*, pp. 495–510, 2013.

[27] M. Chen, E. Moghaddam, N. Mukherjee, J. Rajski, J. Tyszer, and J. Zawada, "Hardware Protection via Logic Locking Test Points," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3020–3030, 2018.

[28] Mentor, "TrustChain Security Platform," https://www.mentor.com/products/sm/trustchain-security-platform, 2017, last accessed on 01/02/19.

[29] DARPA, "Obfuscated Manufacturing of GPS," http://www.cvent.com/events/obfuscated-manufacturing-for-gps-omg-kickoff-meeting/event-summary-72f8574132fc4b38ac6f8ce1761e94af.aspx, 2018, last accessed on 08/12/18.

[30] 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 & Communications Security*, pp. 1601–1618, 2017.

[31] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," *IEEE/ACM Design, Automation & Test in Europe*, pp. 1069–1074, 2008.

[32] S. Dupuis, P. Ba, G. Di Natale, M. Flottes, and B. Rouzeyre, "A Novel Hardware Logic Encryption Technique for Thwarting Illegal Overproduction and Hardware Trojans," *International On-Line Testing Symposium*, pp. 49–54, 2014.

[33] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of Logic Encrypted Chips: Pre-test or Post-Test?" *IEEE/ACM Design, Automation & Test in Europe*, pp. 139–144, 2016.

[34] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[35] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic Encryption: A Fault Analysis Perspective," *IEEE/ACM Design, Automation & Test in Europe*, pp. 953–958, 2012.

[36] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security Analysis of Logic Encryption Against the Most Effective Side-Channel Attack: DPA," *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pp. 97–102, 2015.

[37] S. M. Plaza and I. L. Markov, "Solving the Third-Shift Problem in IC Piracy with Test-aware Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[38] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 137–143, 2015.

[39] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Pan, "Provably Secure Camouflaging Strategy for IC Protection," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 28:1–28:8, 2016.

[40] M. Yasin and B. Mazumdar and O. Sinanoglu and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.

[41] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security Analysis of Anti-SAT," *IEEE Asia and South Pacific Design Automation Conference*, pp. 342–347, 2016.

[42] A. Chakraborty, Y. Xie, and A. Srivastava, "Template Attack Based Deobfuscation of Integrated Circuits," *IEEE International Conference on Computer Design*, pp. 41–44, 2017.

[43] R. 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*, vol. 28, no. 10, pp. 1493–1502, 2009.

[44] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit Sequential Logic Obfuscation: Attacks and Defenses," *IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017.

[45] M. Zaman, A. Sengupta, D. Liu, O. Sinanoglu, Y. Makris, and J. J. Rajendran, "Towards Provably-Secure Performance Locking," *IEEE Design, Automation & Test in Europe Conference & Exhibition*, pp. 1592–1597, 2018.

[46] N. G. Jayasankaran, A. S. Borbon, E. Sanchez-Sinencio, J. Hu, and J. Rajendran, "Towards Provably-Secure Analog and Mixed-Signal Locking Against Overproduction," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, 2018.

[47] S. Koteshwara, C. H. Kim, and K. K. Parhi, "Key-Based Dynamic Functional Obfuscation of Integrated Circuits Using Sequentially Triggered Mode-Based Design," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 79–93, 2018.

[48] Y. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 23, no. 5, pp. 819–830, 2015.

[49] M. Yasin, 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*, vol. 35, pp. 1411–1424, 2016.

[50] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-Based Attack on Cyclic Logic Encryptions," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 49–56, 2017.

[51] K. Azar, H. 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 Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 97–122, 2018.

[52] Kocher, Paul and Jaffe, Joshua and Jun, Benjamin, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," *ACM International Cryptology Conference on Advances in Cryptology*, pp. 388–397, 2018.

[53] M. E. Massad, J. Zhang, S. Garg, and M. V. Tripunitara, "Logic Locking for Secure Outsourced Chip Fabrication: A New Attack and Provably Secure Defense Mechanism," *Computing Research Repository*, vol. abs/1703.10187, 2017.

[54] T. Meade, S. Zhang, and Y. Jin, "Netlist Reverse Engineering for High-Level Functionality Reconstruction," *IEEE Asia and South Pacific Design Automation Conference*, pp. 655–660, 2016.

[55] A. Patooghy, E. Aerabi, H. Rezaei, M. Mark, M. Fazeli, and M. A. Kinsy, "Mystic: Mystifying IP Cores Using an Always-ON FSM Obfuscation Method," *IEEE Computer Society Annual Symposium on VLSI*, pp. 626–631, 2018.

[56] M. Fyrbiak, S. Wallat, J. Déchelotte, N. Albartus, S. Böcker, R. Tessier, and C. Paar, "On the difficulty of fsm-based hardware obfuscation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 293–330, 2018.

[57] M. El Massad, S. Garg, and M. Tripunitara, "Reverse Engineering Camouflaged Sequential Circuits Without Scan Access," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 33–40, 2017.

[58] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-Condition Crunching for Fast Sequential Circuit Deobfuscation," *Design, Automation & Test in Europe Conference Exhibition*, 2019, to appear.

[59] Y. Xie, C. Bao, and A. Srivastava, "Security-Aware 2.5 D Integrated Circuit Design Flow Against Hardware IP Piracy," *IEEE Computer*, vol. 50, no. 5, pp. 62–71, 2017.

[60] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 363–381, 2009.

[61] Chipworks, "Chipworks Launches World's First Reverse Engineering eStore," http://www.marketwired.com/press-release/chipworks-launches-worlds-first-reverse-engineering-estore-1342008.htm, 2010, last accessed on 08/18/18.

[62] Silicon Investigations, "Silicon Investigations Reference Material for IC Reverse Engineering and Patent Protection," http://www.siliconinvestigations.com/ref/ref.htm, last accessed on 07/22/19.

[63] Degate, "Degate Documentation," http://www.degate.org/documentation/, 2012, last accessed on 08/18/18.

[64] S. Zoo, "Layman's guide to IC Reverse engineering," http://siliconzoo.org/tutorial.html, last accessed on 08/18/18.

[65] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on Chip to System Reverse Engineering," *ACM journal on emerging technologies in computing systems (JETC)*, vol. 13, no. 1, p. 6, 2016.

[66] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and Entering Through the Silicon," *ACM SIGSAC Conference on Computer & Communications Security*, pp. 733–744, 2013.

[67] X. T. Ngo, J.-L. Danger, S. Guilley, T. Graba, Y. Mathieu, Z. Najm, and S. Bhasin, "Cryptographically Secure Shield for Security IPs Protection," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 354–360, 2017.

[68] C. Herder, M. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[69] Maxim Integrated, "DeepCover Cryptographic Controller for Embedded Devices," https://www.maximintegrated.com/en/datasheet/index.mvp/id/9173, 2019, last accessed on 07/22/19.

[70] Altera, "Anti-Tamper Capabilities in FPGA Designs," https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01066-anti-tamper-capabilities-fpga.pdf, 2019, last accessed on 07/22/19.

[71] M. Integrated, "DeepCover Security Manager for Low-Voltage Operation with 1KB Secure Memory and Programmable Tamper Hierarchy."

[72] Department of Defense, "Critical Program Information (CPI) Identification and Protection Within Research, Development, Test, and Evaluation (RDT&E)," https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/520039p.pdf, 2018, last accessed on 07/22/19.

[73] M. T. Rahman and S. Tajik and M. S. Rahman and M. Tehranipoor and N. Asadizanjani, "The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes," https://eprint.iacr.org/2019/719.pdf, 2019, last accessed on 07/22/19.

[74] 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*, vol. 14, no. 2, pp. 347–359, 2019.

[75] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic Obfuscation for Creating SAT-Unresolvable Circuits," *ACM Great Lakes Symposium on VLSI*, pp. 173–178, 2017.

[76] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic Locking and Memristor-Based Obfuscation Against CycSAT and Inside Foundry Attacks," *Design, Automation & Test in Europe Conference & Exhibition*, pp. 85–90, 2018.

[77] A. Rezaei, Y. Li, Y. Shen, S. Kong, and H. Zhou, "CycSAT-unresolvable Cyclic Logic Encryption Using Unreachable States," *IEEE Asia and South Pacific Design Automation Conference*, pp. 358–363, 2019.

[78] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott, and H. Zhou, "BeSAT: Behavioral SAT-based Attack on Cyclic Logic Encryption," *IEEE Asia and South Pacific Design Automation Conference*, pp. 657–662, 2019.

[79] S. Roshanisefat, H. Mardani Kamali, and A. Sasan, "SRCLock: SAT-Resistant Cyclic Logic Locking for Protecting the Hardware," *ACM Great Lakes Symposium on VLSI*, pp. 153–158, 2018.

[80] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "Cross-Lock: Dense Layout-Level Interconnect Locking using Cross-bar Architectures," *ACM Great Lakes Symposium on VLSI*, pp. 147–152, 2018.

[81] A. Sengupta, B. Mazumdar, M. Yasin, and O. Sinanoglu, "Logic Locking with Provable Security Against Power Analysis Attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[82] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall, "Interlocking Obfuscation for Anti-tamper Hardware," *ACM Annual Cyber Security and Information Intelligence Research Workshop*, p. 8, 2013.

[83] J. Dofe, Y. Zhang, and Q. Yu, "Dsd: A dynamic state-deflection method for gate-level netlist obfuscation," *IEEE Computer Society Annual Symposium on VLSI*, pp. 565–570, 2016.

[84] S. Khaleghi, K. D. Zhao, and W. Rao, "IC Piracy Prevention via Design Withholding and Entanglement," *IEEE Asia and South Pacific Design Automation Conference*, pp. 821–826, 2015.

[85] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid STT-CMOS Designs for Reverse-Engineering Prevention," *IEEE/ACM Design Automation Conference*, pp. 1–6, 2016.

[86] 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," *IEEE Computer Society Annual Symposium on VLSI*, pp. 405–410, 2018.

[87] J. Yang, X. Wang, Q. Zhou, Z. Wang, H. Li, Y. Chen, and W. Zhao, "c," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 1, pp. 57–69, 2019.

[88] Y.-W. Lee and N. A. Touba, "Improving Logic Obfuscation via Logic Cone Analysis," *IEEE Latin-American Test Symposium*, pp. 1–6, 2015.

[89] Q. Alasad, Y. Bi, and J.-S. Yuan, "E2LEMI:Energy-Efficient Logic Encryption Using Multiplexer Insertion," *Electronics*, 2017.

[90] B. Colombier, L. Bossuet, and D. Hély, "Centrality Indicators for Efficient and Scalable Logic Masking," *IEEE Computer Society Annual Symposium on VLSI*, pp. 98–103, 2017.

[91] N. Karousos, K. Pexaras, I. G. Karybali, and E. Kalligeros, "Weighted Logic Locking: A New Approach for IC Piracy Protection," *IEEE International Symposium on On-Line Testing and Robust System Design*, pp. 221–226, 2017.

[92] S. Gören, C. C. Gürsoy, and A. Yildiz, "Speeding Up Logic Locking via Fault Emulation and Dynamic Multiple Fault Injection," *Journal of Electronic Testing*, vol. 31, no. 5, pp. 525–536, 2015.

[93] R. Karmakar, N. Prasad, S. Chattopadhyay, R. Kapur, and I. Sengupta, "A New Logic Encryption Strategy Ensuring Key Interdependency," *IEEE International Conference on VLSI Design and IEEE International Conference on Embedded Systems*, pp. 429–434, 2017.

[94] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb, "Incremental SAT-Based Reverse Engineering of Camouflaged Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1647–1659, 2017.

[95] Mohamed El Massad and Siddarth Garg and Mahesh V. Tripunitara, "Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes," *NDSS Symposium*, 2015.

[96] L. X. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," *IEEE/ACM Design, Automation & Test in Europe*, 2019, to appear.

[97] F. Tehranipoor, N. Karimian, M. M. Kermani, and H. Mahmoodi, "Deep rnn-oriented paradigm shift through bocanet: Broken obfuscated circuit attack," 2018.

[98] Y. Liu et al., "Secure and effective logic locking for machine learning applications," *Cryptology ePrint, Report 2019/003*, 2019, https://eprint.iacr.org/2019/003.

[99] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "ATPG-based cost-effective, secure logic locking," *IEEE VLSI Test Symposium*, pp. 1–6, 2018.

[100] Y.-C. Chen, "Enhancements to SAT Attack: Speedup and Breaking Cyclic Logic Encryption," *ACM Transactions on Design Automation of Electronic Systems*, vol. 23, no. 4, p. 52, 2018.

[101] M. Massad, S. Garg, and M. Tripunitara, "Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes," *Network and Distributed System Security Symposium*, 2015.

[102] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," *IEEE/ACM Design, Automation Test in Europe Conference Exhibition*, pp. 936–939, 2019.

[103] A. Chakraborty, Y. Liu, and A. Srivastava, "TimingSAT: timing profile embedded SAT attack," *IEEE/ACM International Conference on Computer-Aided Design*, p. 6, 2018.

[104] H. Zhou, Y. Shen, and A. Rezaei, "Vulnerability and Remedy of Stripped Function Logic Locking," https://eprint.iacr.org/2019/139.pdf, 2019, last accessed on 7/22/19.

[105] 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.

[106] A. Sengupta, M. Ashraf, M. Nabeel, and O. Sinanoglu, "Customized Locking of IP blocks on a Multi-Million-Gate SoC," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–7, 2018.

[107] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *ACM International Cryptology Conference on Advances in Cryptology*, pp. 388–397, 1999.

[108] K. Juretus and I. Savidis, "Time Domain Sequential Locking for Increased Security," *IEEE International Symposium on Circuits and Systems*, pp. 1–5, 2018.

[109] J. Zhang, Y. Lin, Y. Lyu, G. Qu, R. C. Cheung, W. Che, Q. Zhou, and J. Bian, "FPGA IP Protection by Binding Finite State Machine to Physical Unclonable Function," *IEEE International Conference on Field Programmable Logic and Applications*, pp. 1–4, 2013.

[110] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre, "A Novel Hardware Logic Encryption Technique for Thwarting Illegal Overproduction and Hardware Trojans," *IEEE International On-Line Testing Symposium*, pp. 49–54, 2014.

[111] A. Nejat, D. Hely, and V. Beroulle, "ESCALATION: Leveraging Logic Masking to Facilitate Path-Delay-Based Hardware Trojan Detection Methods," *Journal of Hardware and Systems Security*, vol. 2, no. 1, pp. 83–96, 2018.

[112] A. Kingsley, "Facepalm of the Day: Intel charges customers $50 to unlock CPU features," https://goo.gl/ZNY6Z4, 2013, last accessed on 08/01/18.

[113] K. Vtt, "Intel to Offer CPU Upgrades via Software for Selected Models," https://bit.ly/2Mnbn2j, 2011, last accessed on 08/01/18.

[114] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," *ACM/IEEE Design Automation Conference*, pp. 1–6, 2017.

[115] J.-L. Tsai, D. Baik, C. C.-P. Chen, and K. K. Saluja, "A Yield Improvement Methodology using Pre-and Post-Silicon Statistical Clock Scheduling," *IEEE/ACM International conference on Computer-aided design*, pp. 611–618, 2004.

[116] M. Li, K. Shamsi, Y. Jin, and D. Z. Pan, "TimingSAT: Decamouflaging Timing-based Logic Obfuscation," *IEEE International Test Conference*, pp. 1–10, 2018.

[117] A. Chakraborty, Y. Xie, and A. Srivastava, "GPU Obfuscation: Attack and Defense Strategies," *IEEE/ACM Design Automation Conference*, pp. 122:1–122:6, 2018.

[118] V. V. Rao and I. Savidis, "Protecting Analog Circuits with Parameter Biasing Obfuscation," *IEEE Latin American Test Symposium*, pp. 1–6, 2017.

[119] J. Wang, C. Shi, A. Sanabria-Borbon, E. Sánchez-Sinencio, and J. Hu, "Thwarting Analog IC Piracy via Combinational Locking," *IEEE International Test Conference*, pp. 1–10, 2017.

[120] J. Leonhard, M. T. N. Muhammad Yasin abd Shadi Turk, M.-M. Louërat, R. Chotin-Avot, H. Aboushady, O. Sinanoglu, and H.-G. Stratigopoulos, "MixLock: Securing Mixed-Signal Circuits via Logic Locking," *Design, Automation & Test in Europe Conference Exhibition*, 2019, to appear.

[121] S. S. Ali, M. Ibrahim, O. Sinanoglu, K. Chakrabarty, and R. Karri, "Security Assessment of Cyberphysical Digital Microfluidic Biochips," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 3, pp. 445–458, 2016.

[122] S. S. Ali, M. Ibrahim, O. Sinanoglu, K. Chakrabarty, and R. Karri, "Microfluidic Encryption of On-Chip Biochemical Assays," *IEEE Biomedical Circuits and Systems Conference*, pp. 152–155, 2016.

[123] C. Pilato, F. Regazzoni, R. Karri, and S. Garg, "TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis," *Proceedings of the 55th Annual Design Automation Conference*, pp. 155:1–155:6, 2018.

[124] M. Shihab, J. Tian, G. R. Reddy, B. Hu, W. S. Jr., B. C. Schaefer, C. Sechen, and Y. Makris, "Design Obfuscation through Selective Post-Fabrication Transistor-Level Programming," *Design, Automation & Test in Europe Conference Exhibition*, 2019, to appear.

[125] U. Guin, Z. Zhou, and A. Singh, "Robust Design-for-Security Architecture for Enabling Trust in IC Manufacturing and Test," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 26, no. 5, pp. 818–830, 2018.

[126] X. Wang, Y. Guo, T. Ramhan, D. Zhang, and M. Tehranipoor, "DOST: Dynamically Obfuscated Wrapper for Split Test against IC Piracy," *Asian Hardware Oriented Security and Trust Symposium*, pp. 1–6, 2017.

[127] R. Karmakar, S. Chatopadhyay, and R. Kapur, "Encrypt Flip-Flop: A Novel Logic Encryption Technique For Sequential Circuits," *arXiv preprint arXiv:1801.04961*, 2018.

[128] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking Obfuscated Scan Chains," *IEEE Asia and South Pacific Design Automation Conference*, pp. 352–357, 2019.

[129] D. Šišejković, R. Leupers, G. Ascheid, and S. Metzner, "A Unifying Logic Encryption Security Metric," *ACM International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, pp. 179–186, 2018.

[130] B. Anckaert, M. Madou, B. De Sutter, B. De Bus, K. De Bosschere, and B. Preneel, "Program Obfuscation: A Quantitative Approach," in *ACM workshop on Quality of Protection*, 2007, pp. 15–20.

[131] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (Im)possibility of Obfuscating Programs," *Annual International Cryptology Conference*, pp. 1–18, 2001.

[132] H. Wee, "On Obfuscating Point Functions," *ACM Symposium on Theory of Computing*, pp. 523–532, 2005.

[133] C. Pilato, S. Garg, K. Wu, R. Karri, and F. Regazzoni, "Securing Hardware Accelerators: A New Challenge for High-Level Synthesis," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 77–80, 2018.

[134] S. Keshavarz, C. Yu, S. Ghandali, X. Xu, and D. Holcomb, "Survey on Applications of Formal Methods in Reverse Engineering and Intellectual Property Protection," *Journal of Hardware and Systems Security*, vol. 2, no. 3, pp. 214–224, 2018.

[135] M. Yasin and O. Sinanoglu, "Evolution of Logic Locking," *IFIP/IEEE International Conference onVery Large Scale Integration*, pp. 1–6, 2017.

[136] B. Liu and G. Qu, "VLSI Supply Chain Security Risks and Mitigation Techniques: A Survey," *Integration*, vol. 55, pp. 438–448, 2016.

[137] B. Colombier and L. Bossuet, "Survey of Hardware Protection of Design Data for Integrated Circuits and Intellectual Properties," *IET Computers & Digital Techniques*, pp. 274–287, 2014.

[138] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining Trust in VLSI Design: Design-for-Trust Techniques," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, 2014.

[139] S. Bhunia and M. Tehranipoor, *Hardware Security: A Hands-on Learning Approach*. Morgan Kaufmann, 2018.

[140] D. Forte, S. Bhunia, and M. M. Tehranipoor, *Hardware Protection through Obfuscation*. Springer, 2017.

[141] F. Bossuet and L. Torres, *Foundations of Hardware IP Protection*. Springer, 2017.

[142] P. Mishra, S. Bhunia, and M. Tehranipoor, *Hardware IP security and Trust*. Springer, 2017.

Abhishek Chakraborty (S'18) is currently pursuing his PhD degree in the Department of Electrical and Computer Engineering, University of Maryland College Park, MD, USA. He obtained his MS in Computer Science and Engineering from Indian Institute of Technology Kharagpur, India in 2016. His primary research interests include Computer Architecture, Hardware Security, and Electronic Design Automation.

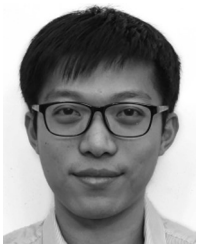
N G Jayasankaran (S'18) is pursuing her Ph.D. in Texas A&M University under the guidance of Prof. Jiang Hu and Prof. JV Rajendran. Her research interest is in the hardware security domain, emphasizing on analog and mixed-signal circuits security. Before joining Texas A&M, she was working on SoC emulation, FPGA design, and board testing for base transceiver station based applications.

Yuntao Liu (S'16) is a Ph.D. candidate with the University of Maryland, College Park, advised by Prof. Ankur Srivastava. His research focus is hardware security, including physical unclonable functions, security in emerging fabrication technologies, logic locking, and the security of machine learning hardware.

Jeyavijayan (JV) Rajendran (S'09, M'15) is an Assistant Professor in the Department of Electrical and Computer Engineering at the Texas A&M University. Previously, he was an Assistant Professor at UT Dallas between 2015 and 2017. He obtained his Ph.D. degree from New York University in August 2015. His research interests include hardware security and computer security. His research has won the NSF CAREER Award in 2017, the ACM SIGDA Outstanding Young Faculty Award in 2019, the ACM SIGDA Outstanding Ph.D. Dissertation Award in 2017, and the Alexander Hessel Award for the Best Ph.D. Dissertation in the Electrical and Computer Engineering Department at NYU in 2016, along with several best student paper awards. He organizes the annual Embedded Security Challenge, a red-team/blue-team hardware security competition and has cofounded Hack@DAC, a student security competition co-located with DAC, and FOSTER. He is a member of IEEE and ACM.

Ozgur Sinanoglu (M'10, SM'15) is a professor of Electrical and Computer Engineering at New York University Abu Dhabi. He obtained his PhD in Computer Science and Engineering from University of California San Diego in 2004. He has industry experience at TI, IBM and Qualcomm, and has been with NYU Abu Dhabi since 2010. During his PhD, he won the IBM PhD fellowship award twice. He is also the recipient of the best paper awards at IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013. Prof. Sinanoglus research interests include design-for-test, design-for-security and design-for-trust for VLSI circuits, where he has around 200 conference and journal papers, and 20 issued and pending US Patents. His recent research in hardware security and trust is being funded by US National Science Foundation, US Department of Defense, Semiconductor Research Corporation, Intel Corp and Mubadala Technology.

Ankur Srivastava (S'00, M'02, SM'15) Dr. Srivastava received his B.Tech in Electrical Engineering from Indian Institute of Technology Delhi in 1998 and PhD in Computer Science from UCLA in 2002. He was awarded the prestigious Outstanding Dissertation Award from the CS department of UCLA in 2002. His primary research interests lie in the field of high performance, low power and secure electronic systems and applications such as computer vision, data and storage centers and sensor networks. He has published numerous papers on these topics at prestigious venues. He has been a part of the technical program & organizing committees of several conferences such as ICCAD, DAC, ISPD, ICCD, GLSVLSI, HOST, and others. He has served as the associate editor for IEEE Transactions on VLSI, IEEE Transactions on CAD and INTEGRATION: VLSI Journal. His research and teaching contributions have also been recognized through various awards.



Yang Xie (S'14) received the B.S. degree in Electrical Engineering from Zhejiang University, China in 2013 and the Ph.D. degree in the Electrical and Computer Engineering Department, University of Maryland College Park, MD, USA in 2018. His research interests include hardware security, trustworthy hardware design, IP piracy prevention, and 3-D IC security.

Muhammad Yasin (S'13, M'18) is a postdoctoral researcher in the Department of Electrical and Computer Engineering at Texas A&M University. He obtained a Ph.D. degree in the Electrical and Computer Engineering Department at New York University in 2018; an MS in Microsystems Engineering from Masdar Institute of Science and Technology, UAE in 2013; and a BS in Electrical Engineering from University of Engineering and Technology (UET) Lahore, Pakistan in 2007. His research interests include Hardware Security, Design for Trust, and Logic Locking. He has co-authored one book, six journal papers, and fourteen conference papers, in addition to filing three US patents. He won the US semi-finals of the TTTC's E.J. McCluskey Best Doctoral Thesis Award in 2018.

Michael Zuzak (S'19) received his M.S. and B.S. in Electrical Engineering from the University of Maryland, College Park, MD, USA in 2016 and 2014, respectively. He is currently pursuing his Ph.D. in Electrical Engineering from the University of Maryland, College Park, MD, USA. His current research interests include hardware security, computer architecture, and electronic design automation.