2019 IEEE 37th VLSI Test Symposium (VTS)

Special Session: Countering IP Security threats in Supply chain

Hassan Salmani
EECS Departmente
Howard University
hassan.salmani@howard.edu

Tamzidul Hoque, Swarup Bhunia
ECE Department
Florida University
{thoque,swarup}@ufl.edu

Muhammad Yasin, Jeyavijayan (JV) Rajendran
ECE Department
Texas A&M University
{myasin, jeyavijayan}@tamu.edu

Naghmeh Karimi
CSEE Department
University of Maryland Baltimore County
naghmeh.karimi@umbc.edu

***Abstract*—The continuing decrease in feature size of integrated circuits, and the increase of the complexity and cost of design and fabrication has led to outsourcing the design and fabrication of integrated circuits to third parties across the globe, and in turn has introduced several security vulnerabilities. The adversaries in the supply chain can pirate integrated circuits, overproduce these circuits, perform reverse engineering, and/or insert hardware Trojans in these circuits. Developing countermeasures against such security threats is highly crucial. Accordingly, this paper first develops a learning-based trust verification framework to detect hardware Trojans. To tackle Trojan insertion, IP piracy and overproduction, logic locking schemes and in particular stripped functionality logic locking is discussed and its resiliency against the state-of-the-art attacks is investigated.**

I. Introduction

Aggressive scaling of VLSI technology results in more complex systems in a single chip. High complexity and cost of design and fabrication of such circuits has invoked the outsourcing of design and fabrication to different parties across the globe. Such globalization of Integrated circuit (IC) design flow has jeopardized the security and trustworthiness of ICs and introduced new security vulnerabilities among which the followings have received significant attention: (1) tampering the circuit to insert malicious circuitry in the form of hardware Trojans aiming at denial of service or leaking sensitive data; (2) reverse engineering aiming at gaining information about the design, and consequently stealing and claiming the ownership of the Intellectual Property (IP); (3) cloning and unallowed overproduction by the foundry.

A hardware Trojan can be inserted during the design or fabrication phase. It resides in hardware and is activated during the hardware operation. In practice, a hardware Trojan may result in denial of service, decreasing the device performance, or leaking sensitive information [1]–[3]. Hardware Trojans usually comprise of a small fraction of the circuit area as otherwise they may be detected due to the change of chip dimension. Moreover, hardware Trojans are mainly stealthy to avoid being detected easily [4], [5].

Utilizing destructive reverse engineering schemes to check the genuineness of manufactured chips is highly costly and cannot ensure that those untested to be Trojan free [6]. On the other hand, deploying VLSI testing schemes in detecting Trojans is not highly effective, as the trigger condition of a Trojan rarely appears [7]. The problem is exacerbated for sequential Trojans as they need a sequence of vectors to be triggered. Side-Channel based Trojan detection schemes that monitor the device's side-channel parameters such as power signatures, path delays, and electromagnetic emanation are more effective methods to detect hardware Trojans as they do not need to trigger a Trojan to detect it [8]. However, the success of these schemes in detecting Trojans is affected by the Trojan size and the process variations occur during the manufacturing process [9], [10]. Machine learning based solutions are promising in detecting hardware Trojans due to their ability to integrate a large number of Trojan properties which could lead to robustness against feature bypassing efforts and significant coverage for existing and future Trojan classes. Accordingly, in this paper, to detect Trojans, a supervised learning-based trust verification framework is proposed and its challenges are discussed extensively.

As discussed earlier, IP piracy, overproduction, and reverse engineering are among other threats that jeopardize the security of integrated circuits. To tackle such threats, logic locking has been proposed in recent years [11], [12]. In practice, logic locking is applied at the design stage and hides the functionality of a design by inserting key-controlled logic gates (key gates) into the original circuit. A locked IC preserves the correct functionality only when the correct key is provided to the circuit. These key gates are fed by the key inputs stored in a tamper-proof memory. As with logic locking, the correct functionality of the design will be hidden from the adversary, not only reverse engineering and piracy is prevented but also hardware-Trojan insertion becomes highly difficult, if not impossible. In practice, an adversary should insert a Trojan in such a way that it is rarely triggered. Otherwise, it would be detected during manufacturing test process. As

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

with the logic locking the correct functionality of the chip is not revealed, Trojan insertion would be very difficult since the inserted Trojan may be triggered more often without the knowledge about the proper functionality of the design. In this paper we focus on a recently proposed logic locking scheme so-called stripped functionality logic locking (SFLL) [13] and discuss its resiliency against recent logic-locking attacks.

The rest of this paper is organized as follows. Section II deals with different type of IPs and explains how hardware Trojans affect each of these IP types. It then discusses the hardware Trojan detection and prevention schemes. Then, Section III proposes a learning-based trust verification framework that provides a completely automated trust-verification flow. The challenges and the possible future improvement of the proposed flow is also discussed in this section. Section IV focusses on logic-locking schemes and how an IP can be protected against piracy, reverse engineering, and Trojan insertion via logic locking. In particular, this section focuses on stripped-functionality logic locking and how this locking scheme will thwart different class of attacks. Finally, Section V concludes the paper.

II. IP Security Threats and Hardware Trojans

IP providers are distributed all across the world, and IPs are being used in designing complex circuits. The application of these complex circuits may range from main frames and personal desktop computers to smart wristwatches, to autonomous vehicles, and to smart homes. Extent and diversity of IP usages mandate their protection against their infringement and modifications. Hardware Trojans (HTs) are a type of circuits that rarely interfere with circuit functional specifications or cause negligible impacts on circuit parametric characteristics. A HT circuit is composed of two main parts: a HT trigger and a HT payload. The HT trigger seeks rare (sequence of) event(s) to modify circuit functionality or its parametric characteristics through the HT payload. Considering the stealthy nature of HTs, they can be discussed from two different perspectives in related to IP protections: IP protection against HT attacks and the use of HTs for IP protection.

A. *IP protection against HT attacks*

The HT trigger maliciously propagates erroneous values under rare conditions through the HT payload to subvert circuit functional specifications or its parametric characteristics. For example, a HT inserted in an IP may seek a specific sequence of instructions to become activated. Upon activation, this HT routes bits of a secret key or signal(s) affected by the secret key to some memory elements or primary outputs. Preventing HT insertion and detecting HTs mandate rigorous vulnerability analyses of IPs toward HT attacks. Such analyses reveal possible types of HTs aiming at infringement and modifications of IPs. Different techniques can be then devised to prevent and detect HTs to protect IPs.

1) HT vulnerability: There are three main categories of IPs [14] soft, firm, and hard. Soft IP blocks are specified at the register transfer level (RTL) or higher-level descriptions. As a hardware description language (HDL) is process independent, they are more suitable for digital cores. They are highly flexible, portable, and reusable, but not necessarily optimized in terms of timing and power. Presented at the layout level, hard IP blocks are highly optimized for a given application in a specific process. Their characteristics are already determined; however, this comes with high cost and lack of flexibility. Firm IP blocks are parameterized circuit descriptions, so they can be optimized according to specific design needs. Firm IPs are between soft and hard IPs, being more flexible and portable than hard IPs, yet more predictable than soft IPs.

Soft IPs presented at RTL using HDLs provide significant opportunities for HTs insertion that can compromise IP security [15] [16]. These IPs can be easily readable by an adversary, and security-sensitive modules can be identified. While a soft IP may meet all specified specifications, it may also provide a lot of spaces for HT insertion. For example, undefined signals under certain conditions can be hijacked to transport sensitive information. Firm IPs are synthesized circuits in reference to some specific design libraries. Compared to a soft IP, a firm IP is composed of signals and logic cells; therefore, an adversary may need to put more effort to identify control and data paths and which parts of circuit perform security-sensitive tasks. However, there may exist still enough rooms to manipulate some signals and logic cells to realize fine-tuned HTs with small footprints [17]. For example, internal memory elements that hold intermediate security-sensitive data can be targeted to covertly leak stored data through some signals to some other memory elements or primary outputs. Hard IPs are placed and routed circuits which are highly optimized at specific technology node toward design goals such as performance and reliability.

Hard IPs provide unprecedented opportunities for implementing types of HTs that are not feasible in soft and firm IPs [18]. For example, some passive elements such as capacitors can tap into some security-sensitive signals, such as a reset signal, and the passive elements can be controlled by software code. As a result, a software-controlled HT can be realized. Vulnerabilities of hard IPs made of analog or mixed-signal circuits to HTs have recently obtained considerable attentions, as well [19]. For example, it has shown that it is possible to leak secret key through manipulating wireless transceivers.

2) HT Prevention: Preventing HT insertion at IPs means their instrumentation such that no space practically remains for an adversary to realize any HTs. For soft IPs, IP encryption at RTL is a technique used in industry to facilitate sharing soft IPs between vendors without revealing their detailed implementation. As a result, a user's capabilities are limited to IP verification and synthesis, and it makes circuit manipulations challenging for an attacker. Some other techniques have proposed equipping a soft IP with information flow control mechanisms to ensure no security-sensitive information (intentionally) leaks from an soft IP [20]. Some work has suggested split manufacturing at the RTL [21]. A soft IP is divided into RTL units, and a submitted circuit to an untrusted foundry is literally a sea of unconnected RTL units. Connections between

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

RTL units are unknown to the untrusted foundry; hence, it makes it very challenging to realize a HT.

For firm IPs, IP obfuscation is one of major techniques that is being widely studied. In general, some selected signals are mixed with some secret keys such that an untrusted party will not be able to easily determine whether the signals are originally inverted or not. As a result, an adversary cannot easily determine signals' functionality. Some other work proposes encryption of IPs' communication with other IPs in a complex circuit. Communication encryption makes it considerably challenging for an attacker to induce internal computation of IPs. As a result, it also becomes very difficult for an attacker to purposefully manipulate communications between IPs [22].

For hard IPs, the split manufacturing technique is another technique that splits a layout in two parts to prevent HT implementation. The layout's substrate and transistors with low level metal layers are manufactured by an (untrusted) foundry, and the upper metal layers are manufactured by another (trusted) foundry. As connections between logic cells seem incomplete for an untrusted foundry, it cannot develop a functional model of circuit. As a result, it becomes very challenging to design HTs effectively attacking security-sensitive assets. Camouflaging is another technique that has been considerably investigated to hide the functionality of logic cells itself. Logic cells are implemented using lookup tables or some programmable multiplexers so that a foundry does not know the functionality of logic cells during manufacturing [23]. Some other techniques in this line have proposed logic cells whose layout are similar in their images, but they deliver different functionalities [24].

3) HT Detection: There has been extensive study on HT detection techniques before circuit manufacturing and after circuit manufacturing. Majority of techniques targeting HTs before circuit signoff perform various static and dynamic analyses to capture HTs through suspicious logical activities. On the other hand, techniques targeting HTs after circuit manufacturing mainly look for HT footprints on side-channel signals or perform online monitoring to flag any activities different from expected behaviors.

Before circuit manufacturing, a circuit under authentication undergoes various verification techniques by a trusted entity to isolate inserted HTs. As a circuit is available in its original form, i.e. neither encrypted nor obfuscated, different techniques have been proposed based on an IP's abstraction level.

For soft IPs, there exist techniques that perform control-flow subgraph matching or look for known patterns of HTs at RTL to detect them. These techniques assume the existence of a library containing HT triggers and payloads. As it is not expected that HT lines become executed frequently, some techniques suggest identifying suspicious signals by use of formal verification, coverage analysis, removing redundant circuit, sequential automatic test pattern generation and equivalence theorems. Some other work also propose using formal methods to ensure predefined security properties have not been violated when a soft IP arrives to a customer [20] [25].

For firm IPs, some techniques study switching activities of signals to detect HTs. These techniques basically apply (random) test patterns to a circuit and observe switching activities across the circuit. Some techniques then continue with guided test pattern generation targeting suspicious modules to determine whether the suspicious ones are HTs with a higher confidence [26]. Some others perform correlation analyses between switching activities of signals and use machine learning techniques to isolate HTs [27]. Opposite to circuits dynamic analyses, i.e. test pattern application, some techniques perform static analyses of gates and signals to detect HTs. A technique has hypothesized that HT signals need to have either considerably high controllability or high observability values to present a stealthy behavior [28]. As a result, this technique detects HTs based on their testability values in gate-level netlist.

For hard IPs, some techniques may still extract corresponding gate-level netlist of a hard IP to detect HTs. While such techniques mainly look for functional HTs, hard IPs also enable designing new types of HTs. Hard IPs open door to analog and mix-signals IPs so that sophisticated HTs can be designed at the layout level. Detection of majority of HTs for hard IPs is performed after circuit manufacturing.

After circuit manufacturing, detecting HTs takes two main forms: performing circuit testing before using the circuit in a system and conducting runtime monitoring of circuit. To have an effective HT detection after circuit manufacturing, several design-for-trust techniques have been proposed [20]. These techniques magnify HT contributing into a circuit side channel signals during circuit testing or facilitate security property checking during online monitoring. To detect HTs through circuit testing, some techniques strive to (partially) activate HTs and capture their footprints in circuit power consumption [29]. Some other techniques suggest various delay-based testing to capture extra delay introduced by a HT due to HT cells placement and routing [30]. Real-time online learning techniques are being also proposed to learn a complex circuit's behavior based some application-oriented information [31]. Any behavior residing far from expected behaviors in the field is flagged as HTs. Using wrappers around IP modules can also being used to ensure security policies have been followed by taking over control over some input/output signals [32].

B. The use of HTs for IP protection

While majority of study has been on detecting HTs in IPs, there has been some study on using HTs for IP protection. It has shown that a sequential HT can be embedded in a firm IP to realize IP expiration date. These HTs can be obfuscated to avoid their removal, and they become activated when a sequence of rare events occurs. A checking mechanism based on the sequence of states in the host design's finite state machine has been introduced to determine if a soft IP is used in a legal host design or not [33].

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

III. Learn & Churn: A Knowledge guided Approach to Hardware IP Trust Assurance

A. Learning-based Trust Verification

System-on-Chip (SoC) based design paradigm has become commonplace for design houses where SoC developers extensively rely on hardware designs obtained from external vendors. Use of third-party hardware intellectual property (IP) cores provides a significant economic benefit by eliminating the cost and associated delay of developing individual hardware IP cores from scratch. While the soft-IPs obtained in the form of register-transfer-level code or gate-level netlist are verified for functional correctness during and after integration, it is extremely hard to assure that no hidden or malicious functionality exists within the untrusted IP core. The challenge of detecting hidden malicious circuits or hardware Trojans has received significant attention from the government and academia during the last decade where various detection, prevention, and monitoring strategies have been developed that are applicable to specific Trojan insertion scenario. While the detection of foundry inserted Trojans received the highest attention, most countermeasures applicable to this context require a golden design. As this requirement cannot be met for detecting Trojans in untrusted third-party IP (3PIP) cores, a significantly different detection approach is required for IP trust assurance.

Researchers have leveraged the white-box observability of the untrusted 3PIP core to propose various Trojan detection solution that uses static analysis of the IP. Static analysis involves examining the structures within the design without simulating it. Dynamic analysis based detection techniques observe the design response during simulation and often requires the Trojans to be activated for detection. The major challenge in the context of dynamic analysis is the generation of intelligent test vectors that could fully activate the Trojan and make the malicious impact observable. Along with the scalability issue for large IPs, dynamic analysis is limited to the Trojan classes that creates observable functional change within the design once triggered.

Static analysis based detection methods have been proven effective in detecting hard-to-activate Trojans by observing individual functional features of nets within the design such as switching activity and controllability [34]. While these solutions are effective for specific Trojan classes (i.e., ones that rarely trigger), structurally redesigning the Trojans to eliminate the very few functional features have been proven effective in bypassing such technique [35]. Besides, activity and controllability related features could be insufficient in detecting some of the existing (e.g., always-on) and new Trojan types that do not exhibit any difference compared to the normal segment of the design when examined based on these features. Machine learning based solutions are promising in this regard due to their ability to integrate a large number of Trojan properties which could lead to robustness against feature bypassing efforts and significant coverage for existing and future Trojan classes. While similar learning-based trust-verification approach has been studied by others [36]–[38], they suffer from the following major disadvantages: 1) use of either structural or functional features [28], 2) use of features only applicable to Trust-HUB Trojans, 3) common training and testing process for Trojan classes with opposite behaviour, 4) reliance on a limited set of static Trojan database for training, 5) imperfect validation of the approach where the training and testing database contain common instances.

In [39], we propose a supervised learning based framework that eliminates all the aforementioned weaknesses and provides a completely automated trust-verification flow containing four primary steps as shown in Fig. 1. First, a set of Trojan-free sample IPs are collected on which a large number of valid Trojans are inserted for each known class of Trojans that we want to detect. An automated Trojan insertion tool is used for this purpose. For a particular class, the Trojan instances are varied with respect to a number of structural and functional parameters including the Trojan fan-in, switching activity, insertion location etc. Tool-based insertion of Trojan helps to eliminate the reliance on a limited static database during training. A total of 14 different generic features are used that effectively capture both the functional and structural behaviour of each net. Along with the activity related features, we examined a number of functional features that are related to the Boolean function of the net's driving logic. With the help of commercial synthesis tools or in-house netlist parser, the desired functional and structural properties for each net is extracted. The labeled training data for each Trojan class is then used for generating separate trained model capable of detecting corresponding classes of Trojans. Class-wise separation of training and testing process enables better understanding of the verification outcome and provides increased detection performance compared to other supervised learning based approaches [39]. Finally, the test database is generated from the suspect IP and provided to each trained model dedicated to detecting the diverse Trojan classes. Each model identifies a set of suspect nets that could potentially belong to Trojans.

B. Challenges in Learning-based Trust Verification

In this section, we discuss some of the major challenges and possible future improvements that are significant for the proposed approach.

1) Selection of Sample Benchmarks: Generation of training set requires a number of sample benchmarks to be inserted with Trojans. The data points obtained from the normal and Trojan inserted segments of these benchmarks help the classifier to learn about the non-Trojan and Trojan net behaviour respectively. In [39], four of the largest ISCAS89 benchmarks were selected as sample benchmarks and the designs were inserted with approximately 800 combinational and sequential Trojans. Since the available sample IPs could have diverse functionality, it could be hard to select the most appropriate benchmarks for a given suspect design. One naive approach would be to choose training benchmarks that have similar functionality to the suspect IP. For instance, if the suspect IP is an AES core, several other open source block ciphers

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

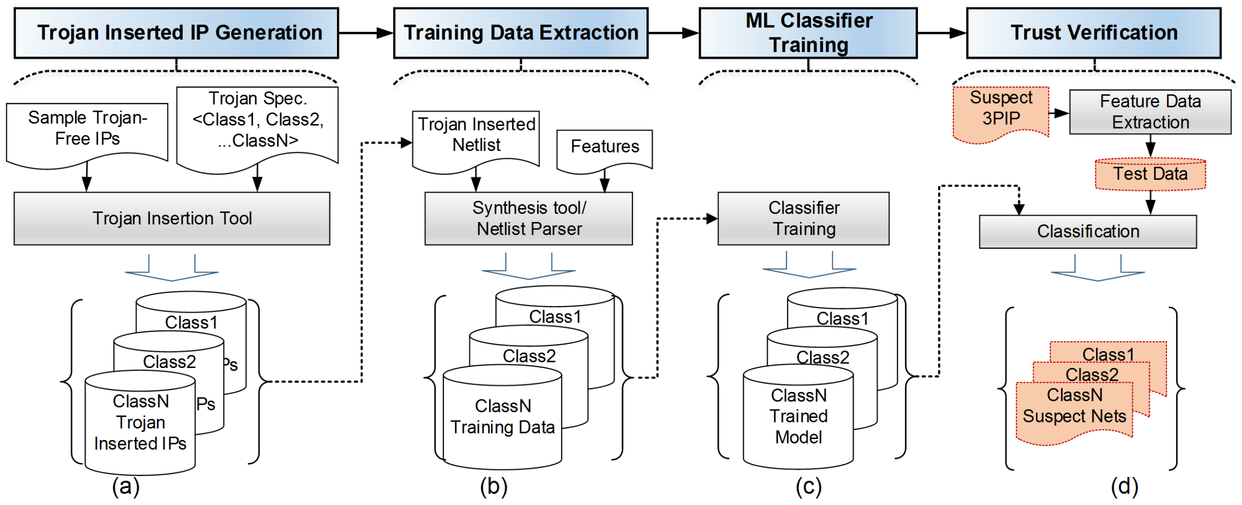

Fig. 1. The overall machine learning based IP trust verification framework proposed in [39].

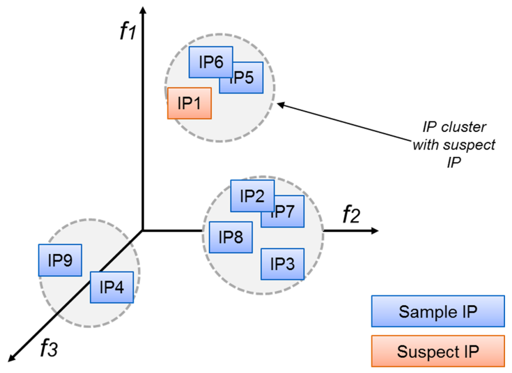

Fig. 2. IP-level clustering to identify sample benchmarks for training that are similar to the suspect IP.

could be selected. While this approach is intuitive, a more formal approach would be to apply an unsupervised clustering at the IP-level based on some features as shown in Fig. 2. However, the features to be used for net-level classification are not directly applicable for IP-level clustering. Hence, some function of the net-level features should be used. For instance, if we intend to use controllability of nets as net-level features for Trojan detection, the IP-level clustering could be done using the average and standard deviation of controllability of all nets in the IP. It is expected that the sample IPs that form a cluster with the suspect IP have similarity among the majority of nets with respect to the features to be used for Trojan detection. Hence, as shown in Fig. 2, to detect Trojans in IP1, IP5 and IP6 could be used as training benchmarks.

2) Trojan Insertion Parameters: The Trojan insertion tool has a number of configurable parameters which dictates the functional and structural behavior of the inserted Trojans. Since for a sample design with moderate size (10K nets) thousands of valid Trojan instances could be inserted, under a given trust verification time bound only a limited number of Trojans could be inserted to generate the training data. Hence, the optimum number of Trojans to be inserted and the range of Trojan parameters that must be covered among these limited Trojan set is an important research question that could dictate the quality of the training data and required time to verify an IP.

3) Feature Selection: As the number of features to be extracted from the design increases, training and test time is affected. Besides, not all features would be useful for each class of Trojans. Even though the advantage of certain features for specific classes of Trojan could be intuitively understood with domain knowledge, there could be features that combinedly facilitate the classification but appears ineffective when observed separately. For the reduction of verification time and a better understanding of the test results, it is important to select the best possible set of feature for individual Trojan class. Hence, a wrapper-based feature selection process could be used. In wrapper algorithm, a mining algorithm (e.g., Naive Bayes) is predefined first and utilized for feature subset evaluation. For each generated feature subset, the mining algorithm is applied to data and the goodness of the subset is determined by assessing the quality of the mined outputs. Hence, wrapper based feature selection could be applied on the training data to estimate which features could provide the best detection accuracy for a specific Trojan class.

4) Selection of Classifiers : Once the training data is generated, one or more classification algorithm has to be selected for training. Depending on the nature of the training and test data, the performance of the classifier is impacted. For instance, in [39], three different classifiers (i.e., Random Forest, Naive Bayes, and AdaBoost-DecisionStump) were tested along with their voting ensemble that combines the result from all three. As shown in Fig. 3, for two different classes of Trojans (i.e., combinational and sequential), the performance of the classifiers vary significantly among different classification algorithm. While no single classifier outperformed the rest in detecting all Trojan classes, the best result was obtained through the voting of multiple classifiers. The performance of

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

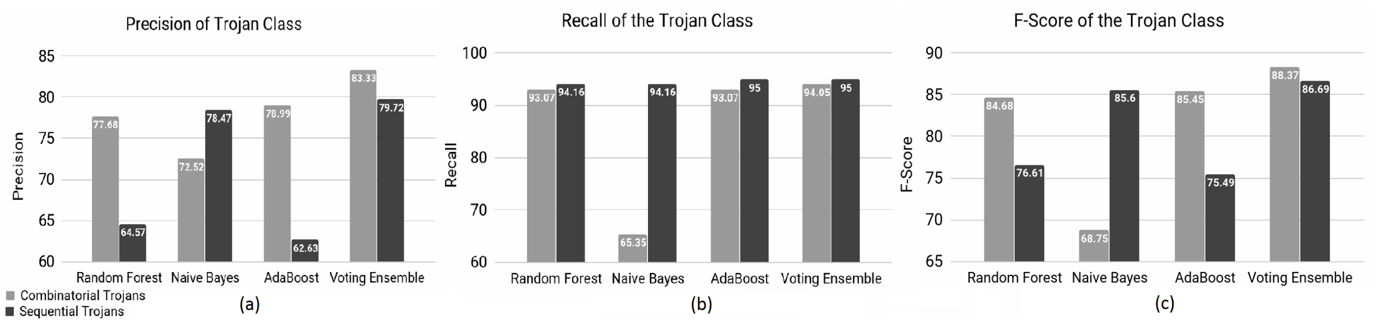

Fig. 3. The average (a) precision, (b) recalll, and (c) F-Score of the two Trojan classes for different machine learning algorithms in detecting a number of combinational and sequential Trojans obtained from Trust-HUB.

such a voting scheme is yet to be observed for other classes of Trojans (e.g., always-on and DeTrust Trojans [35]). While a voting-based scheme sounds attractive, it increases the overall verification time as multiple classifiers have to be used. Hence, selection of the best classifier for each class of Trojan should be done by experimenting with a large number of classifiers and diverse Trojan classes.

5) Adversarial Learning : Similar to the application of machine learning in malware and network intrusion detection, the proposed supervised learning for IP trust verification is vulnerable to adversarial learning. An attacker with the Trojan insertion tool and knowledge of the overall scheme could try to design Trojans to bypass the proposed verification approach. The assumption is that the attacker can test the Trojan inserted IP with the proposed scheme before sending it to the design house where the suspect IP is tested by the verification engineer under a similar implementation of the proposed scheme. One method of attack would be to understand the feature that is responsible for the successful detection of a particular Trojan. The Trojan could be redesigned ensuring that the identifying signature is reduced to a point where the Trojan is no longer detectable. However, the attacker must ensure that the modification does not allow the Trojan to get easily detected or observed using new features or complementary detection techniques (e.g., logic testing). Thwarting adversarial attack would require verification personnel to execute experiments similar to the attacker to improve the Trojan designs to be inserted by the tool for robust training.

IV. Logic Locking

As mentioned earlier, the globalization of the integrated circuit (IC) supply chain has given rise to security threats such as IP piracy, overbuilding, reverse engineering, and hardware Trojans [40]. Logic locking has emerged as a promising defense against these threats [11], [13], [41]–[44]. Logic locking locks a circuit by adding introducing additional key-controlled logic into a circuit. In addition to the original inputs, a locked circuit has *key inputs* that are driven by an on-chip tamper-proof memory. The added protection logic may consist of XOR *key* gates [11], [41], look-up-tables (LUTs) [45] or complex Boolean functions [13], [43]. The *locked* netlist passes through the untrusted design phases, i.e., untrusted manufacturing, test, and assembly. Without the knowledge of the secret key, neither a design can be pirated nor an IC can be made functional.

A. A brief history of logic locking

The earliest logic locking techniques focused on discovering the most suitable locations for inserting XOR/XNOR key-gates [11], [12], [45]. However, many key-recovery attacks have been launched on these techniques [12], [42]. The most notable among these attacks is the SAT that attack that leverages *Boolean satisfiability* to eliminate incorrect key. Subsequent research on logic locking defends against the SAT attack by combining an original circuit with a point-function [43], [46]. Although point-function techniques, e.g., SARLock [46] and Anti-SAT [43], thwart the SAT attack, they are vulnerable to the removal attacks. These attacks can identify and remove the protection circuity [47]. An approximate key for these techniques can also be recovered by different variants of the SAT attack [48]. Stripped Functionality Logic Locking (SFLL) is the first technique to defend against all the aforementioned attacks in a provable way [13]. Another recent technique, cyclic logic locking, resist the SAT attack by introducing cycles in a netlist [49].

Apart from the combinational logic techniques mentioned above, sequential locking techniques that lock finite state machines have also been developed [50], [51]. However, all these techniques are susceptible to state-machine-reconstruction [52] and bounded-model-checking-based attacks [53], [54].

B. Stripped-Functionality Logic Locking (SFLL)

As mentioned, logic locking is a promising defense against piracy and reverse engineering. This section focuses on a specific class of logic locking techniques, referred to as stripped-functionality logic locking (SFLL). We will elaborate on the resilience of SFLL against the latest attacks. In particular, in this section, we summarize the operation of SFLL and its variants. We present "post-SFLL"attacks that have been developed after the inception of SFLL, and to answer the frequently asked questions (FAQs) on the security of SFLL, we elaborate on the resilience of SFLL against these post-SFLL attacks.

SFLL thwarts all classes of attacks on logic locking, i.e., SAT [42], removal [47], [56], and approximate [48], [57].

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

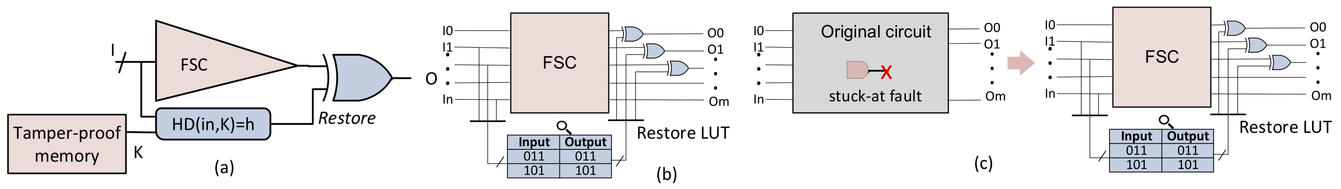

Fig. 4. The three variants of SFLL: a) SFLL-HD [13], b) SFLL-Flex [13], and c) SFLL-Fault [55].

SFLL is based on the notion of "strip and restore". Selected functionality is taken-away from the original circuit and is stored as secret key(s). Upon fabrication, the chip is activated by loading the secret key(s) to an on-chip memory.

SFLL has three variants: SFLL-HD, SFLL-flex, and SFLL-fault, which are discussed next.

1) SFLL-HD: SFLL-HD creates a functionality-stripped circuit (FSC) by inverting the output of the original circuit for $\binom{k}{h}$ input patterns that are of Hamming distance (HD) h from the k-bit secret key. These patterns are referred to as protected input patterns. The number of the protected input patterns represents the resilience to removal attacks. As depicted in Fig. 4(a), a single HD-unit forms the restore unit that recovers the original functionality upon applying the correct key to it. For the incorrect keys, however, an error is introduced into the circuit systematically, i.e., in a way that maximizes the resilience to the SAT attack [13].

2) SFLL-Flex: SFLL-HD suits applications where it is useful to protect a large number of input patterns; the patterns may be selected on an arbitrary basis. However, in many applications, a specific set of input patterns or a range of input patterns has to be protected. SFLL-Flex locks a circuit cost-effectively by compacting the patterns-to-be-protected using a small set of input cubes (input patterns with don't care bits) [13]. The cubes are stored on an on-chip look-up table as illustrated in Fig. 4(b).

3) SFLL-Fault: SFLL-HD and SFLL-Flex realize functionality-stripping through existing logic synthesis algorithms. For example, the FSC in SFLL-HD is obtained by synthesizing the original circuit along with an HD-unit containing a hard-coded secret key. The HD-unit is susceptible to removal when it is not fully merged with the original circuit. Instead of adding logic to a circuit, SFLL-Fault subtracts the logic through fault injection [55]. SFLL-Fault, thus, does not leave any structural traces in a netlist and hampers removal attacks [55].

C. Post-SFLL attacks and resilience of SFLL

1) CycSAT [58] and BeSAT [59]: CycSAT is a variant of SAT attack that breaks cyclic logic locking [58]. CycSAT adds additional clauses that help extract a key that either renders the circuit acyclic or ensures that none of the cycles are sensitizable. CycSAT, however, may run into scalability issues when the cyclic circuit maintains a state, i.e., the output for a given key/input combination differs based upon the state. BeSAT improves upon CycSAT to resolve statefulness [59].

Resilience of SFLL. Both CycSAT and BeSAT build upon the SAT attack by adding constraints to remove cycles from a design. When considering the resilience of SFLL, these attacks are no different than the SAT attack. The additional clauses play no role in defeating SFLL. Thus, SFLL thwarts CycSAT and BeSAT in the same say way as it thwarts the SAT attack.

2) Structural analysis using machine learning (SAIL) [60]: The SAIL attack uses machine learning to determine the secret key. The attack exploits the correlation between the secret key value and the netlist structure [60]. Note that upon insertion of XOR key-gates, the netlist structure typically changes only locally, i.e., within a few levels of logic.

Resilience of SFLL. The SAIL attack can succeed only when the impact of XOR/XNOR key-gate insertion is localized within a few levels of logic. The SAIL attack is successful against RLL, where the impact of individual key-gates is localized and detached from that of the other key-gates [60]. This allows the machine learning algorithm to learn templates that help match the local netlist changes to the key-bit value. SFLL, however, does not introduce any localized changes to a netlist that may be linked to the values of individual key-bits. In SFLL, all the key-bits are entangled, e.g., through an AND-tree or an HD-unit, and cannot be targeted on an individual basis. The SAIL attack thus fails to target SFLL, where a divide-and-conquer strategy is not applicable.

3) Satisfiability modulo theory (SMT) attack [61]: The SMT attack is a superset of the SAT attack since it relies on SMTs that are more general than SAT [61]. The SMT attack incorporates the SAT attack and its multiple variants, including the accelerated SAT attack and the approximate attacks.

Resilience of SFLL. The SMT attack mainly shows that all variants of the SAT attack can also be implemented using SMT. The attack targets logic locking techniques that have been vulnerable to other variants of the SAT attack. It does not explicitly target any weakness of SFLL. Since SFLL is secure against the different variants of SAT attack that the SMT attack implements, it also remains secure against the SMT attack. The SMT attack against SFLL would require the same number of iterations as the SAT attack.

4) Functional analysis attack on logic locking (FALL) [62]: The FALL attack utilizes structural and functional properties of the HD-unit to identify and remove it [62]. The attack is successful when the HD-unit with the hard-coded secret key is left as is during logic synthesis.

Resilience of SFLL. As already mentioned, SFLL-HD and SFLL-Flex "add" an HD-unit or an AND-tree, respectively,

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

with a hard-coded secret key before re-synthesizing the circuit. The HD-unit and the AND-tree may remain intact post-synthesis. The FALL attack can thus circumvent SFLL-HD and SFLL-Flex, since it can use the properties of the HD-unit and AND-trees to locate them inside a netlist. This also gives rise to the need for developing secure logic synthesis algorithms that can defend against the attack.

However, the FALL attack cannot break SFLL-Fault that strips functionality by "subtracting/removing" logic from the original circuit. The subtraction of the logic does not leave any traces for an attacker to exploit. An attacker cannot guess exactly what logic removed from a circuit to construct its current form.

5) Redundancy identification [63]: The redundancy identification attack relies on the observation that incorrect keys may create logic redundancy in a netlist, rendering certain faults untestable [63]. By comparing the changes to the structure of a netlist for a key-bit value of 0 and a key-bit value of 1, the attack can identify the more likely value for a given key-bit.

Resilience of SFLL. Similar to the SAIL attack, the redundancy identification attack succeeds using a divide-and-conquer approach that targets individual key-bits. The attack cannot break SFLL, where all key-bits feed a restore unit and must be considered in tandem.

In sum, most of CycSAT, SAIL, SMT, FALL, and redundancy identification attacks target the weaknesses of specific combinational and sequential logic locking techniques but fail to break SFLL. The other attacks target only specific variants of SFLL. Thereby, no existing attack can break all variants of SFLL. Note that we need secure logic synthesis algorithms to further secure SFLL and other logic synthesis techniques against structural attacks.

V. Conclusion

This paper discussed different IP security threats that can occur in the supply chain and the schemes that can be deployed to tackle such threats. In particular, we focused on Trojan detection schemes in different type of IPs and proposed a learning-based framework that when deployed significantly improves Trojan detection rate. In addition, we discussed how logic locking schemes counter IP piracy, cloning, reverse engineering and Trojan insertion threats and investigated how the recently proposed stripped functionality logic locking scheme can tackle the state-of-the-art logic locking attacks.

References

[1] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug 2014.

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

[3] J. Dubeuf, D. Hly, and R. Karri, "Run-time detection of hardware Trojans: The processor protection unit," in *ETS*, 2013, pp. 1–6.

[4] M. Banga and M. S. Hsiao, "A region based approach for the identification of hardware Trojans," in *HOST*, 2008, pp. 40–47.

[5] N. Karimi, J.-L. Danger, and S. Guilley, "On the effect of aging in detecting hardware Trojan horses with template analysis," in *International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 281–286.

[6] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *HOST*, 2008, pp. 51–57.

[7] R. Rad et al., "Power supply signal calibration techniques for improving detection resolution to hardware Trojans," in *ICCAD*, 2008, pp. 632–639.

[8] D. Agrawal et al., "Trojan detection using IC fingerprinting," in *IEEE Symp. on Security and Privacy*, 2007, pp. 296–310.

[9] T. Hoque et al., "Golden-free hardware Trojan detection with high sensitivity under process noise," *Journal of Electronic Testing*, vol. 33, no. 1, pp. 107–124, 2017.

[10] J. Zhang, H. Yu, and Q. Xu, "Htoutlier: Hardware Trojan detection with side-channel signature outlier identification," in *HOST*, 2012, pp. 55–58.

[11] J. Roy, F. Koushanfar, and I. Markov, "Ending Piracy of Integrated Circuits," *IEEE Computer*, vol. 43, pp. 30–38, 2010.

[12] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," *IEEE/ACM Design Automation Conference*, pp. 83–89, 2012.

[13] 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.

[14] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. P. Pande, C. Grecu, and A. Ivanov, "System-on-chip: Reuse and integration," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1050–1069, 2006.

[15] H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2013, pp. 190–195.

[16] H. Salmani and S. G. Miremadi, "Contribution of controller area networks controllers to masquerade failures," in *11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, 2005.

[17] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, 2013, pp. 471–474.

[18] H. Salmani and M. M. Tehranipoor, "Vulnerability analysis of a circuit layout to hardware Trojan insertion," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1214–1225, 2016.

[19] A. Antonopoulos, C. Kapatsori, and Y. Makris, "Security and trust in the analog/mixed-signal/RF domain: A survey and a perspective," in *2017 22nd IEEE European Test Symposium (ETS)*, 2017, pp. 1–10.

[20] H. Salmani, *Trusted Digital Circuits - Hardware Trojan Vulnerabilities, Prevention and Detection*. New York: Springer., 2018.

[21] X. Cui, J. J. Zhang, K. Wu, S. Garg, and R. Karri, "Split manufacturing-based register transfer-level obfuscation," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 1, pp. 11:1–11:22, 2019.

[22] S. Bhunia, S. Ray, and S. Sur-Kolay, *Fundamentals of IP and SoC Security*. New York: Springer., 2017.

[23] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid STT-CMOS designs for reverse-engineering prevention," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, pp. 88:1–88:6.

[24] M. Tehranipoor, H. Salmani, and X. Zhang, *Integrated Circuit Authentication - Hardware Trojans and Counterfeit Detection*. New York: Springer., 2013.

[25] P. Mishra, S. Swarup, and M. Tehranipoor, *Hardware IP Security and Trust*. New York: Springer., 2017.

[26] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-Trojans at gate-level netlists," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 465–470.

[27] B. Çakir and S. Malik, "Hardware Trojan detection for gate-level ICs using signal correlation based clustering," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 471–476.

[28] H. Salmani, "COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2017.

[29] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, "Hardware Trojan detection and isolation using current integration and localized current analysis," in *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, 2008, pp. 87–95.

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.

[30] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware Trojans impacting circuits delay," *IEEE Design & Test of Computers*, vol. 30, no. 2, pp. 26–34, 2013.

[31] A. Kulkarni, Y. Pino, and T. Mohsenin, "SVM-based real-time hardware Trojan detection for many-core platform," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016, pp. 362–367.

[32] J. Portillo and E. John, "Using static hardware wrappers to thwart hardware Trojans and code bugs at runtime," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2018, pp. 1034–1037.

[33] S. Narasimhan, R. S. Chakraborty, and S. Chakraborty, "Hardware IP protection during evaluation using embedded sequential Trojan," *IEEE Design Test of Computers*, vol. 29, no. 3, pp. 70–79, 2012.

[34] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 697–708.

[35] J. Zhang, F. Yuan, and Q. Xu, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 153–166.

[36] K. Hasegawa, M. Yanagisawa, and N. Togawa, "A hardware-trojan classification method using machine learning at gate-level netlists based on trojan features," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 7, pp. 1427–1438, 2017.

[37] ——, "Hardware trojans classification for gate-level netlists using multi-layer neural networks," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2017, pp. 227–232.

[38] ——, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[39] T. Hoque, J. Cruz, P. Chakraborty, and S. Bhunia, "Hardware ip trust validation: Learn (the untrustworthy), and verify," in *2018 IEEE International Test Conference (ITC)*. IEEE, 2018, pp. 1–10.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 127–146, 2016.

[47] 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.

[48] 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.

[49] ——, "Cyclic Obfuscation for Creating SAT-Unresolvable Circuits," *ACM Great Lakes Symposium on VLSI*, pp. 173–178, 2017.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

[56] 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.

[57] Y. Shen and H. Zhou, "Double DIP: Re-Evaluating Security of Logic Encryption Algorithms," *ACM Great Lakes Symposium on VLSI*, pp. 179–184, 2017.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," *Design, Automation & Test in Europe Conference Exhibition*, 2019, to appear.

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

Authorized licensed use limited to: Somaiya University. Downloaded on October 09,2023 at 07:15:05 UTC from IEEE Xplore. Restrictions apply.