TAAL: Tampering Attack on Any Key-based Logic Locked Circuits

AYUSH JAIN, ZIQI ZHOU, and UJJWAL GUIN, Auburn University, USA

Due to the globalization of semiconductor manufacturing and test processes, the system-on-a-chip (SoC) designers no longer design the complete SoC and manufacture chips on their own. This outsourcing of the design and manufacturing of Integrated Circuits (ICs) has resulted in several threats, such as overproduction of ICs, sale of out-of-specification/rejected ICs, and piracy of Intellectual Properties (IPs). Logic locking has emerged as a promising defense strategy against these threats. However, various attacks about the extraction of secret keys have undermined the security of logic locking techniques. Over the years, researchers have proposed different techniques to prevent existing attacks. In this paper, we propose a novel attack that can break any logic locking techniques that rely on the stored secret key. This proposed *TAAL* attack is based on implanting a hardware Trojan in the netlist, which leaks the secret key to an adversary once activated. As an untrusted foundry can extract the netlist of a design from the layout/mask information, it is feasible to implement such a hardware Trojan. All three proposed types of *TAAL* attacks can be used for extracting secret keys. We have introduced the models for both the combinational and sequential hardware Trojans that evade manufacturing tests. An adversary only needs to choose one hardware Trojan out of a large set of all possible Trojans to launch the *TAAL* attack.

Additional Key Words and Phrases: Logic locking, IP Piracy, IC Overproduction, Hardware Trojans, Tampering

ACM Reference Format:
Ayush Jain, Ziqi Zhou, and Ujjwal Guin. 2020. TAAL: Tampering Attack on Any Key-based Logic Locked Circuits. *ACM Trans. Des. Autom. Electron. Syst.* 1, 1 (December 2020), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The continuous addition of new functionality in a system-on-a-chip (SoC) has enforced designers to adopt newer and lower technology nodes to manufacture chips primarily to reduce the overall area and the resultant cost of a chip. Building and maintaining such a fabrication plant (foundry) requires a multi-billion dollar investment [2]. As a result, the semiconductor industry has moved towards horizontal integration, where an SoC designer acquires intellectual properties (IPs) from many different vendors and sends the design to a foundry for manufacturing, which is generally located offshore.

The hardware layers that were assumed to be trusted are no longer true with the outsourcing of IC fabrication in a globalized and distributed design flow, including multiple entities. Third-party IPs, fabrication, and test facilities of chips represent security threats to the current horizontal integration of the production. The security threats posed by these entities include – (*i*) overproduction of ICs [4, 5, 9, 15, 24, 31, 63], where an untrusted foundry fabricates more chips without the consent

Authors' address: Ayush Jain, ayush.jain@auburn.edu; Ziqi Zhou, ziqi.zhou@auburn.edu; Ujjwal Guin, ujjwal.guin@auburn.edu, Auburn University, Department of Electrical and Computer Engineering, Auburn, AL, USA.

1084-4309/2020/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of the SoC designer in order to generate revenue by selling them in the market, (*ii*) sale of out-of-specification/rejected ICs [21, 24, 25, 56, 104], and (*iii*) IP piracy [11, 14, 78, 79, 104], where an entity in the supply chain can use, modify and/or sell functional IPs illegally. Over the years, researchers have proposed different techniques to prevent the aforementioned attacks and they are IC metering [4, 5, 40, 63], logic locking [24, 58, 63], hardware watermarking [19, 36, 53], and split manufacturing [33, 82].

Logic locking has emerged as a promising technique and gained significant attention from the researchers to address the different threats emerging from untrusted manufacturing. In logic locking, the netlist of a circuit is locked so that it produces incorrect results unless it is programmed with a secret key. The locks are generally inserted in the netlist using XOR gates. Traditional logic locking methods involved selection of key gate location as random selection [25, 26, 63], strong interference-based selection [58], and fault-analysis based selection [60]. Over the years, researchers have also proposed different attacks to extract the secret key and undermine the locking mechanisms. Boolean Satisfiability (SAT)-based attacks have demonstrated effective ways of extracting the secret keys. Countermeasures are also proposed so that SAT-based attacks become infeasible.

This paper shows that any locked circuit, where the secret key is stored in an on-chip memory no matter whether it is tamper-proof or not, can be broken by inserting a hardware Trojan. We present this attack as **TAAL**: **T**ampering **A**ttack on **A**ny **L**ocked circuit that uses a stored key for locking the netlist. This attack can defeat the security measures provided from any existing logic locking methods. *We believe that we are the first to show that any key-based logic locked circuit can be exploited by inserting a hardware Trojan in the netlist.* The contributions of this paper are described as follows:

- We propose a novel attack based on malicious modifications of the netlist to target any key-based locked circuit. The attacking approach is to tamper the locked netlist in order to extract the secret key information. Once the valid key information is extracted from an activated IC, an untrusted foundry can unlock any number of chips and sell overproduced and defective chips. As this attack applies to any key-based locked circuits, an adversary can undermine any secure solutions proposed so far to prevent overproduction, sourcing defective/out-of-spec chips, and IP piracy. Note that different researchers have proposed to use logic locking to prevent hardware Trojans [44, 65, 98, 101–103] as an adversary cannot precisely specify the trigger conditions when a design is locked. However, we exploit a Trojan to obtain the secret key, which is the primary contribution of this paper.
- We present three types of TAAL attacks that extract the secret key differently using hardware Trojans placed at different locations in the netlist. *T1 type TAAL* attack directly leaks the secret key to the primary output of an IC once the Trojan is activated (see Figure 2.(a)). On the other hand, *T2 type TAAL* and *T3 type TAAL* attacks rely on the activation and propagation of the secret key to the primary output (see Figure 2.(b) and Figure 3 respectively). Note that an adversary has the freedom of choosing one of these three types of TAAL attacks implemented using combinational, sequential, or analog hardware Trojans.
- We present an existing well-known model for designing a combinational hardware Trojan [107], which can be used to launch the TAAL attacks. An adversary has the freedom to choose the type of hardware Trojan, which can be designed so that it evades manufacturing or production tests and remains undetected. These combinational Trojans can be described as *Type-p* Trojans, as they have *p* trigger inputs. These triggers can come from the primary inputs and/or internal nodes of a locked circuit, which are not affected by the keys. We present that a very large number of Trojans can be created, and an adversary can use only

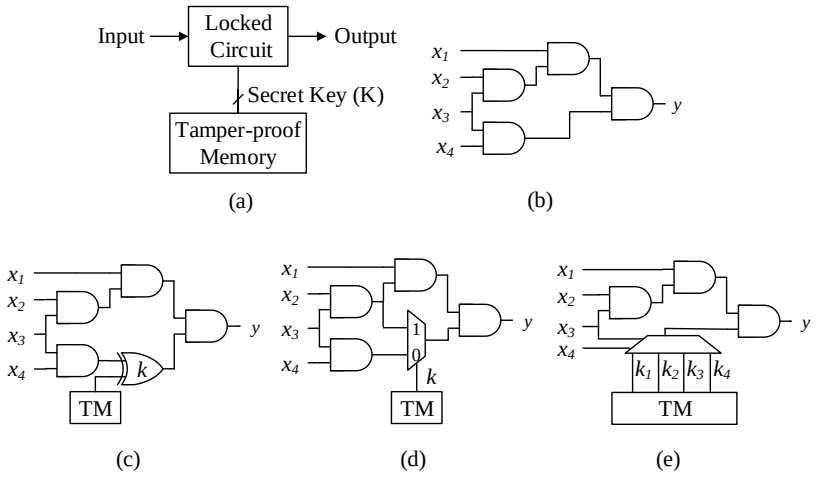

Fig. 1. Different logic locking techniques. (a) A locked circuit, where the secret key (K) is programmed in a tamper-proof memory (TM). (b) Original circuit. (c) XOR-based locking. (d) MUX-based locking. (e) LUT-based locking.

one such a Trojan. It is practically impossible for an SoC designer to detect all these feasible Trojans using logic tests.

- We also present a model for sequential Trojan, which is constructed using a combinational one. A state element (a counter) is added to a combinational Trojan to deliver the payload once it is triggered R times consecutively. Note that the trigger inputs for both the Trojans are the same. The combinational Trojan delivers the payload once triggered; on the other hand, a sequential Trojan needs to be triggered R times consecutively. Note that an adversary can choose any existing design of a hardware Trojan proposed so far.

The rest of the paper is organized as follows: an introduction to logic locking along with an overview of different locking techniques and attacks is provided in Section 2. The proposed attacks based on hardware Trojan to implement the malicious design modification for the extraction of the secret key from any locked circuit are described in Section 3. We provide an algorithm for designing an *Type-p* combinational Trojan considering the set of manufacturing test patterns in Section 4. The number of valid Trojans, along with other factors such as area overhead and leakage power for several benchmark circuits, are presented in Section 5. The future directions are provided in Section 6. Finally, we conclude our paper in Section 7.

2 BACKGROUND

The challenges for protecting a circuit against hardware security threats have been the driving force for developing different techniques to limit the amount of circuit information that can be recovered by an adversary. Logic locking has emerged as a field of significant interest from the researchers, as it can provide complete protection against IC overproduction and IP piracy. Different researchers have proposed to use logic locking to prevent hardware Trojans as well [44, 65, 98, 101–103].

The objective of logic locking is to obfuscate the inner details of the circuit and make it infeasible for an adversary to reconstruct the original netlist. Logic Locking hides the circuit's functionality by inserting additional logic gates into the original design, which we termed as *key gates*. In addition to the original inputs, the locked circuit needs secret key inputs to key gates from on-chip tamper-proof memory (see Figure 1.(a) for details). The correct functionality of the design is obtained

when the key inputs receive the proper secret key value. Applying an invalid key to the key gates would result in incorrect functionality of the locked design. Note that for a securely locked circuit, the design details cannot be recovered using reverse engineering.

Different logic locking methods were devised over the years and can be categorized into three different categories. First, XOR-based logic locking, shown in Figure 1.(c), has received much attention due to its simplicity. In this technique, a set of XOR or XNOR gates are inserted as key gates [24–26, 58, 63, 92, 97, 99, 100]. The secret key is stored in tamper-proof memory (TM), and connections are made from TM to the key gates. Second, in the MUX-based logic locking technique [41, 52], multiplexers (MUX) are inserted so that one of its input is correct, which is the actual net of the circuit. The other input of the MUX is incorrect, which is a dummy net randomly selected from the netlist. This technique is shown in Figure 1.(d). The select signal of the MUX is associated with the key bit from the tamper-proof memory. The correct signal goes through the MUX upon applying valid key value; otherwise, the incorrect signal propagates in the netlist. Third, in LUT-based logic locking, [9, 38, 44], shown in Figure 1.(e), a look-up table with several key inputs is used to lock the netlist. The LUTs replace a combinational logic in the design, making it difficult to predict the output as it depends on several different key values.

The research community has proposed several attacks to exploit the security vulnerability on a logic locked circuit. Subramanyan et al. [74] first showed that a locked circuit could be broken using Boolean Satisfiability (SAT) analysis. The SAT attack algorithm, attributed as an oracle-guided attack, requires a locked netlist, which can be recovered using reverse engineering and functional chip with a valid key stored/programmed in its tamper-proof memory. In this attack, an adversary can query an activated chip and observe the response. Note that the SAT attack requires access to the internal nodes of the circuit through the scan chains, which is common in today's netlist for implementing Design-for-Testability (DFT) [13]. The SAT attack works iteratively to eliminate incorrect key values from the key space using distinguishing input patters (DIPs). A DIP is defined as an input pattern for which two sets of hypothesis keys produce complementary results. By comparing these with the output of an unlocked chip, one set of hypothesis keys is discarded. The SAT attack works efficiently as it discards multiple hypothesis keys in one iteration.

Thereafter, researchers have focused on improving and developing locking techniques to be resilient against the SAT attack. Subsequent work in this direction involved Anti-SAT [92, 93], SARLock [97], TTLock [100], SFLL [99], design-for-security (DFS) architecture [24–26]. These proposed techniques use one-point functions. SARlock inverts the output of the circuit for one input pattern corresponding to one incorrect key. This input pattern differs for different incorrect keys. Anti-SAT involves two complementary external logic circuits which are supplied with the same key values and same inputs. The output of these two circuits converge into a AND gate whose output is always 0 for the correct key applied; else, it may be 1 that leads to corrupted internal node value in the original netlist to produce incorrect outputs. Anti-SAT, initially, was proven vulnerable to the signal probability skew (SPS) [97] attack and removal/bypass attack [95]. SARLock was broken by Double DIP attack [68], Approximate SAT attack [66] and removal/bypass attack [95].

Due to limitations of SARlock, Yasin et al. developed an improved version of this design and referred to as TTLock [100], where the original design itself is modified to produce corrupted/inverted results upon applying an incorrect key for a single input pattern. Stripped functionality based logic locking (SFLL) [99] was proposed to provide more flexibility in the number of protected input patterns. The design is no longer the same as the original design due to stripped parts of the functionality resulting in erroneous output. A separate restore unit is responsible for removing this error in the output, supplying correct key values to it. However, Subramanyan et al. has shown that SFLL can be defeated through FALL attack [71]. The attack is built on three primary steps, namely, structural analysis, functional analysis, and key confirmation. The structural analysis is performed

to identify the gates that are the output of the cube stripping function in SFLL. After identification of these candidate gates, the functional analysis targets the property of cube stripping functions, which results in a set of potential key values. Finally, the key confirmation algorithm identifies the correct key from the set of potential key values.

As the SAT-attack is based on the availability of accessing the internal states of a circuit through the scan chains, Guin et al. proposed placing multiple flip-flops capturing signals controlled by different key bits at the same level of the parallel scan chains, which were used in the current test compression methodologies [24]. However, a vulnerability existed in this design, when an adversary performs multi-cycle tests, such as delay tests (transition delay faults and path delay faults) [13]. This leads to the necessity for developing a new design-for-security (DFS) architecture to prevent leaking of the key during any manufacturing tests [25, 26]. This design prevents scanning out the internal states after a chip is being activated, and the keys are programmed/stored in the circuit.

Apart from SAT-based attacks, probing attacks [55, 85] have also shown serious threats to the security of logic locking, where an attacker makes contact with the probes at signal wires in order to extract sensitive information, mainly, the secret key. With the help of a focused ion beam (FIB), a powerful circuit editing tool that can mill and deposit material with nanoscale precision, an attacker can circumvent protection mechanisms and reach wires carrying sensitive information. However, the countermeasures reflect the complexity of shield-structure and nanopyramid structures as the defense, making it difficult to perform these attacks [67, 86]. Recently, Zhang et al. proposed an oracle less attack to extract the key from locked circuits [105]. The notion of this attack is to compare the locked and unlocked instances of repeated Boolean functions in the netlist to predict the key. A solution was proposed to countermeasure the attack as well.

3 PROPOSED TAAL ATTACK FOR EXTRACTING SECRET KEYS

The general hardware security strategy adopted for designing and manufacturing a circuit involves a logic locking, where a chip is unlocked by storing a secret key in the tamper-proof memory. As this secret key is the same for all the chips manufactured with the same design, finding this key from one chip undermines the security resulted from logic locking. We show that an adversary can easily extract the key for a chip using our proposed TAAL attacks, built on tampering through malicious modification by inserting a hardware Trojan to a locked circuit. In this section, we will present three types of TAAL attacks.

3.1 Adversarial Model

The adversarial model is given to defining the capabilities and intentions of an attacker clearly. In this model, the attacker (adversary) is assumed to be an untrusted foundry and possesses the following:

- The attacker has access to the locked netlist of a circuit. An untrusted foundry has access to all the layout information, which can be extracted from the GDSII or OASIS file. The netlist can be reconstructed from the layout using reverse engineering with advanced technological tools [80].
- The attacker can determine the location of the tamper-proof memory. It can also find the location of key gates in a netlist, as it can easily trace the route of the other input of the key gate to the tamper-proof memory.
- The attacker can tamper a netlist for its malicious intentions through inserting additional circuitry, commonly known as hardware Trojans, about which the SoC designer is unaware.
- The attacker has access to all the manufacturing test (e.g. stuck-at-fault, delay fault) patterns. Commonly, the production tests are performed at the foundry.

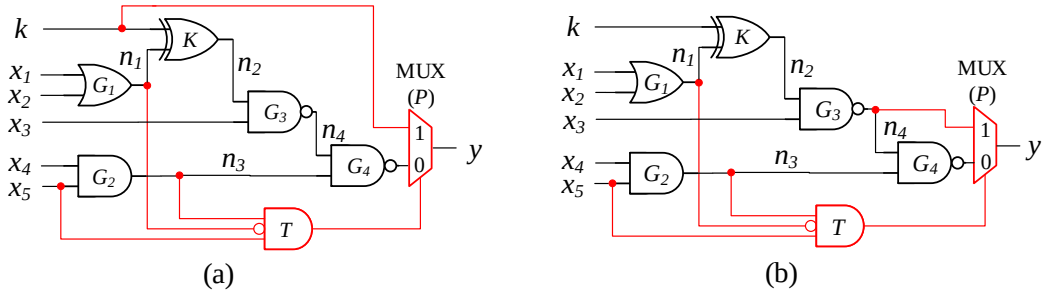

Fig. 2. (a) *T1 type TAAL attack*, where a *Type-3* combinational Trojan is inserted for key extraction directly from the connection between key gate and tamper-proof memory, (b) *T2 type TAAL attack*, where a *Type-3* combinational Trojan is inserted for the secret key extraction.

3.2 T1 Type TAAL Attack

The *T1 type TAAL* attack is the most straightforward attack among the other two types, which will be introduced in the successive sections. This attack is beneficial for the attacker who does not intend to gain knowledge regarding the security measures implemented for the circuit. The hardware Trojan assists in extracting out the secret key directly from the tamper-proof memory.

Figure 2.(a) shows the proposed modification to launch *T1 type TAAL* attack. A *Type-3* combinational Trojan (see details in Section 4.1) is designed and inserted in the netlist. A combination of 3-input AND gate with an inverter at one of its input served as the trigger and denoted as T, whereas the 2-input multiplexer delivers the payload to the primary output. One input of the multiplexer is the actual output of the locked netlist. The other input is connected to the line formed between the key gate (K) and the tamper-proof memory as a part of logic locking. Under normal operation for any activated chip, the multiplexer propagates the correct circuit functionality at the output. Once the Trojan gets activated with $[x_1\ x_2\ x_3\ x_4\ x_5] = [0\ 0\ X\ 1\ 1]$, the output of AND gate becomes 1, which leads to the extraction of the secret key through the multiplexer at the output. Note that the required number of multiplexers to extract the complete secret key is dependent on the key size.

The *T1 type TAAL* attack is very effective as it does not require any knowledge of the circuit netlist. This attack can also be applied to any logic locking techniques without knowing its implementation details as it directly leaks the key to the primary output. As there are no security measures undertaken in logic locking to protect the connection between the key gates and the tamper-proof memory, any locked circuit can be vulnerable for this attack. Note that an adversary can select any hardware Trojans (combinational or sequential Trojans) of its choice, and one can find the implementation details in Section 4.

3.3 T2 Type TAAL Attack

Instead of extracting the key directly to the primary output, an adversary can propagate it to the output. In *T1 type TAAL Attack*, once the Trojan is activated, the raw key values are transferred to primary output, which can raise a suspicion of a design being tampered. *T2 type TAAL Attack* primarily addresses this shortcoming of *T1 type attack*, by incorporating logic values in the key, which can easily be separated.

The attack involves tampering a netlist with a *Type-3* combinational Trojan. Similar to *T1 type*, the trigger is constructed using a 3-input AND gate along with an inverter placed before one of the AND gate inputs (see Figure 2.(b)) and the payload is delivered to the primary output of the circuit

using a 2-input MUX. An adversary can choose a net, whose logic value is impacted by the key gate for the input of the MUX. In Figure 2.(b), net n_4 is selected as the MUX input. One can also select n_2, however, n_3 cannot be selected. The key gate is considered as XOR gate having inputs as secret key (k) and G_1 gate output. In order to propagate the secret key at the output of the key gate (K), G_1 output needs to be specified (either 0 or 1). If $n_1 = 0$, the output of the key gate will be k, otherwise it will be $\overline{k}$. For this example, the trigger requires $n_1 = 0$; else, the Trojan will not be activated. This condition forces $[x_1\ x_2] = [0\ 0]$. Since net n_4 is selected for key extraction, input x_3 also plays a significant role in key propagation as the complementary value at net n_2 can propagate to n_4 only when x_3=1. To launch this attack, an adversary needs to perform the circuit analysis to sensitize the key. This attack requires an adversary to monitor the input pattern to extract the correct key. An adversary extracts $\overline{k}$ when $[x_1\ x_2\ x_3\ x_4\ x_5] = [0\ 0\ 1\ 1\ 1]$, in the example provided in Figure 2.(b). This attack shows the flexibility to identify individual key gate and target secret key through key propagation from consecutive node selection. The dependency of this attack on primary inputs increases the efficiency of *T2 type* attack where only the adversary knows the logical values of these inputs.

3.4 T3 Type TAAL Attack

A locked netlist typically consists of a large number (e.g., 128) of key gates, the effect of one key may affect the propagation of another key to the primary output. A secure logic locking technique can also insert keys in such a way that an adversary cannot propagate the key information to output using manufacturing tests [58]. In such scenarios, *T2 type TAAL attack* may be ineffective in extracting the key. *T3 type TAAL attack* is proposed to addresses this limitation encountered for *T2 type attack*.

Figure 3.(a) shows the locked netlist, where the propagation of the key (k_1) is prevented by inserting another key (k_2). The output of G_3 cannot be uniquely determined unless an adversary knows either k_1 or k_2 and *T2 type attack* will fail to determine either k_1 or k_2. It is thus necessary to help propagate one key and then determine the other. Figure 3.(b) shows our proposed *T3 type TAAL attack*, where net n_5 is selected to deliver the payload.

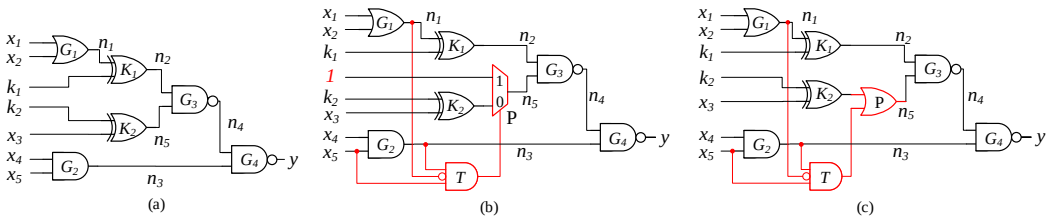

Fig. 3. T3 Type TAAL attacks. (a) Original netlist with k_2 inserted to prevent the propagation of k_1, (b) *T3 type TAAL attack* with a *Type-3* combinational Trojan with payload as multiplexer (MUX) and (c) *T3 type TAAL attack* with a *Type-3* combinational Trojan with payload as OR gate.

Figure 3.(b) shows the implantation of *T3 type TAAL attack* using a *Type-3* combinational Trojan. The trigger part for the Trojan can be designed as 3-input AND gate with inverter and payload is delivered through 2-input MUX as before. The key (k_1) propagation requires setting the node n_5 to 1 so that the signal value at node n_2 can propagate through G_3. The other input of the MUX is directly connected to V_{DD}, which is equivalent to logic 1.

The other input of the key gate (K_1) needs to be specified to propagate the key k_1 at node n_2. As one of the inputs to trigger requires $n_1 = 0$ to be satisfied, which results $[x_1\ x_2] = [0\ 0]$, and

the output of key gate (K_1) will be k_1. When the trigger condition is met, the output of the AND gate becomes 1, and logic 1 is delivered at the input of G_3 through the MUX. At this point, the Trojan activation nullifies the effect of key value k_2 at the output of $G3$. The signal at n_4 will be $\overline{k_1}$. Finally, setting node n_3 at logic 1 will expose the key at the output, y. As a result, input pattern $[x_1\ x_2\ x_3\ x_4\ x_5] = [0\ 0\ X\ 1\ 1]$ will expose the key k_1 at the output. Once the value of k_1 is known, an adversary can perform the signal propagation analysis to find k_2. Similarly, the payload MUX can be replaced with an OR gate as shown in Figure 3.(c), the attack works in the same manner where triggering of Trojan would force the output of the payload OR gate to logic 1 irrespective of its other input which will assist in propagating the key value k_1 at the primary output depending on the primary input values as discussed above.

The attacks are explained using a combinational Trojan for the simplicity of understanding. All the attacks proposed can also be implemented using any *Type-p* sequential Trojan, which delivers at its targeted payload once the Trojan is activated R times. The design details for a sequential Trojan is discussed in Section 4.2.

Note that the *T3 Type* TAAL attack cannot be applied to DFS structure [25, 26] since DFS architecture prevents leaking of key through the secure cell. However, the adversary can implement either *T1 Type* and *T2 Type* TAAL attacks to bypass the secure cell and be extracted out through scan-chain.

4 DESIGN OF HARDWARE TROJANS FOR TAAL ATTACKS

A hardware Trojan can be described as intentional modifications in the original netlist of a design for malicious purposes [1, 10, 37, 72, 76, 77]. A Trojan can be inserted into a circuit during its design or manufacturing stages. In this paper, we only consider a Trojan, inserted by an untrusted foundry, which is relevant to logic locking (see the adversarial model in Section 3.1). As logic locking was proposed to address the threat from an untrusted foundry, it can practically thwart the locking mechanism by obtaining the secret key through inserting a hardware Trojan in the design.

A complete hardware Trojan classification can be found in [11]. In this paper, we only consider combinational and sequential hardware Trojans to demonstrate the attack. A combinational hardware Trojan generally comprises of a trigger and a payload, the detailed modeling can be found in [107]. On the other hand, sequential Trojans have a state element along with the trigger and payload [10, 87]. Any Trojans can be activated through trigger inputs, which can be taken from the primary inputs and/or internal nodes of a circuit so that manufacturing test patterns cannot trigger a Trojan and remain undetected. The trigger can be implemented as an AND gate. When a Trojan is activated, the output of this AND gate becomes 1 and it delivers the payload (selection input of the multiplexer shown in Figures 2-3) to the circuit to leak the secret key. The trigger can also be any logic function that provides 1 when activated. Note that a combinational Trojan manifests its effects upon the availability of the trigger inputs and effects the original netlist at the payload, on the other hand, sequential Trojan shows its effect after the occurrence of a sequence or a period of time upon triggered.

4.1 Design for a Combinational Hardware Trojan

The primary purpose of a hardware Trojan, once activated, is to modify the original functionality of a circuit to leak the secret key, which is unknown to the SoC designer. It is absolutely necessary that the Trojan must not get activated during scan-based structural or functional tests. In other words, the circuit should not come across any condition during tests that activates the trigger, which can lead to its detection. In this paper, we present a step-by-step process for designing a combinational hardware Trojan, initially presented in [107], for a locked netlist.

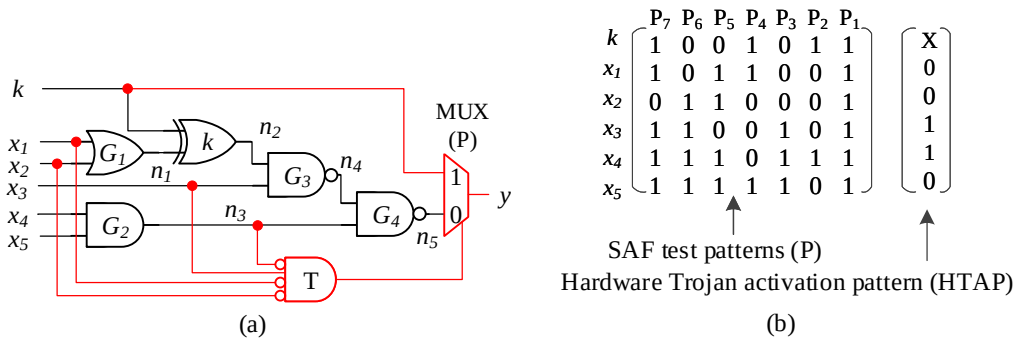

Fig. 4. Design for a combinational hardware Trojan that evades manufacturing tests. (a) A combinational circuit with a *Type-4* Trojan. (b) Stuck-at fault (SAF) test patterns for manufacturing tests. The hardware Trojan activation pattern (*HTAP*) is $[x_1\ x_2\ x_3\ x_4\ x_5]$ = [0 0 1 1 0] while treating the key input as unknown (X).

A hardware Trojan can be described based on its trigger inputs, and can be defined as *Type-p* Trojan when it has p trigger inputs. The trigger inputs can be selected from primary inputs and/or internal nodes, which are not affected by the key gates. If such a node is selected as a trigger input, an adversary cannot activate a Trojan as it does not know this secret key, and thus the internal signal value for an activation pattern. We call this pattern as hardware Trojan activation pattern (HTAP). The payload of the Trojan (a MUX) can be delivered to a location described in Figures 4 for launching our proposed TAAL attack.

Let us determine the number of Trojans, one can insert in a design to extract the secret key. This is basically a selection problem, where an adversary selects p nodes as the trigger inputs from N nodes of a circuit so that the Trojan is not activated during the manufacturing/production tests. The value of N can be determined based on the following equation:

$$N = PI + G + F - M \tag{1}$$

where,

PI : Number of primary inputs
G : Number of gates
F : Number of fanout branches
M : Number of lines impacted by the key gates

An upper bound of all possible *Type-p* Trojans (AT_p) can be given by:

$$AT_p = \binom{N}{p} \times 2^p \tag{2}$$

The right-hand side of Equation 2 constitutes of two products. The first one represents all possible combinations to select p lines from N. The second one denotes the trigger combinations, as one line can be applied directly or inverted to the trigger input. Note that the actual number of Trojans (denoted as VT_p) can be less than AT_p as few of them can be detected by the manufacturing test patterns (e.g., stuck-at fault patterns), and few may not be triggered from the primary inputs. However, for a reasonable size circuit, AT_p and VT_p are comparable.

Figure 4 shows an example of *TAAL: T1 type* using a *Type-4* Trojan inserted in the netlist. The circuit has five primary inputs (*PI*). The SoC designer can generate test patterns considering the key as input (the pattern generation is described in detail in [25, 26]. To detect all the stuck-at faults

(SAFs), seven test patterns (e.g., $P = \{P_1, P_2, \ldots, P_7\}$) are required and they are generated using Synopsys TetraMax [75] ATPG tool. To avoid a Trojan being activated by these manufacturing test patterns, the Trojan's trigger must remain quiet for all these input patterns. A hardware Trojan activation pattern is selected, where $HTAP = (X\ 0\ 0\ 1\ 1\ 0)^T \notin P$. As the logic values of nodes n_2, n_4, and n_5 are impacted by the key, k, these nodes are excluded in designing the Trojan. If one of these nodes is selected, an adversary may not activate the trigger as it does not know the key value. The upper bound of all possible *Type-4* Trojans (AT_4) can be given by:

$$AT_4 = \binom{7}{4} \times 2^4 = 560 \tag{3}$$

where, $N = 5 + 5 - 3 = 7$.

Out of these 560 possible Trojans, 188 will be detected by the test patterns P. The remaining 372 will be treated as valid Trojans, and an adversary can select one of them. In Figure 4, $\overline{x_1}$, x_2, $\overline{x_3}$ and $\overline{n_3}$ are selected as the trigger. Similarly, one can also design other types (*Type-1* through *Type-6*) of Trojans to launch TAAL attacks. Note that the Trojan needs to be quiet during normal operations so that no functional errors are observed at the primary output. An adversary can select rare nodes, whose value do not become identical with trigger pattern very often under continuous/normal operation of the IC, for the trigger inputs while designing a Trojan. One can perform controllability, and observability analysis [13] to find such rare nodes, and then select them as the trigger inputs. However, we do not include such analysis, as including rare nodes in the trigger for a sequential hardware Trojan is not a hard requirement. In this paper, a sequential Trojan is modeled using a combinational Trojan that needs to be triggered R times consecutively.

Algorithm 1: Design of Type-p Trojan

input : Locked Netlist (C), test pattern set (P), *Type-p Trojan*
output: Hardware Trojan activation pattern ($HTAP$), Trigger inputs (T)

1 Read the locked netlist (C);
2 Read production test patterns (P);
3 Perform logic simulation using P to form a matrix (A) of all the internal node values;
4 Select a hardware Trojan activation pattern ($HTAP$), where $HTAP \notin P$;
5 Perform logic simulation using $HTAP$ and form a matrix (H) of all the internal node values;
6 Select p random nodes that are not affected by the key gates of C for the trigger inputs;
7 Construct a new matrix A_p that corresponds to the trigger locations for all test patterns;
8 Construct a new vector H_p that corresponds to the trigger locations for $HTAP$;
9 **if** $H_p \notin A_p$ **then**
10 | Choose selected p nodes as trigger, T;
11 **else**
12 | Discard selected p nodes, as it would activate the Trojan during tests;
13 | Go to Step 6;
14 **end**
15 Report $HTAP$ and T;

An automated process is developed to design a combinational hardware Trojan. Algorithm 1 provides steps to be followed for designing a *Type-p* Trojan that eludes activation during the manufacturing test. The inputs of the algorithm are locked netlist(C), M production/manufacturing

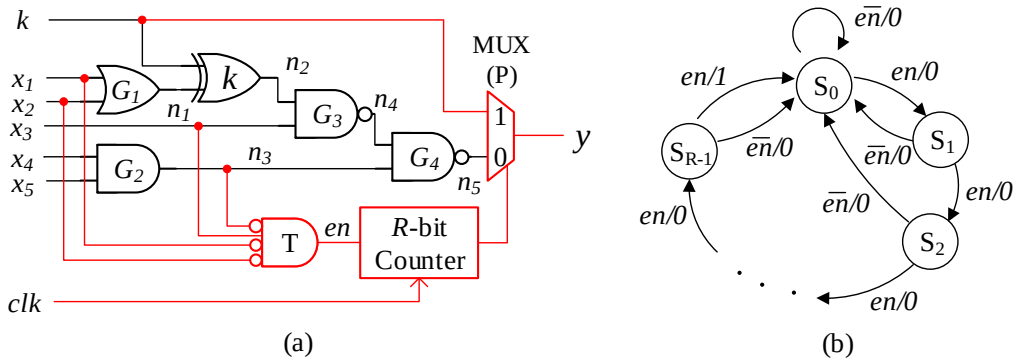

Fig. 5. (a) The netlist of a sequential Trojan with a *R-bit* counter, (b) The finite state machine (FSM) of the counter used in a sequential Trojan.

test patterns ($P = \{P_1, P_2, \ldots, P_M\}$) and the Trojan type ($p$). The algorithm results in the hardware Trojan activation pattern (HTAP) and the trigger inputs. Initially, it reads the Trojan-free locked netlist (C) and the set of manufacturing test patterns (P) (Lines 1-2). Logic simulation is performed using these patterns, and store the internal node values in a matrix, A (Line 3). It is unnecessary to store the nodes that are impacted by the key gates, as their values will be unknown (Xs) during the simulation. Note that A is a $N \times M$ matrix. A hardware Trojan activation pattern of an adversary's choice is selected (Line 4). Similarly, matrix H is formed due to logic simulation using HTAP (Line 5). Here, H is a $N \times 1$ vector. To select the trigger inputs, one can select p random nodes that are not affected by the key gates of the locked circuit (Line 6). To perform the search whether the trigger values are presented in the production set, the matrix A_p and vector H_p are constructed (Lines 7-8). If H_p is not in A_p, the trigger (T) is selected (Line 10), otherwise, drop the selected p locations as the Trojan will be activated during the production tests (Lines 12-13) and new p locations are selected (Line 6). Finally, the algorithm reports $HTAP$ and T (Line 15).

4.2 Design for a Sequential Hardware Trojan

A sequential Trojan modifies the functionality of a circuit until a specified time has elapsed after the trigger condition is satisfied. However, in this paper, we designed a sequential Trojan that needs to be triggered R times to deliver the payload. We designed a sequential Trojan in this way so that it can be modeled using a combinational Trojan, described in detail in the previous section.

A sequential Trojan also consists of a trigger and payload similar to a combinational Trojan. Additionally, the trigger part contains state elements that ascertain the payload in future time. In our sequential Trojan design, a *R-bit* counter is implemented as the state elements. This counter is enabled (*en*) once the trigger condition is fulfilled, i.e., the output of the p-input AND gate becomes 1. The counter increments by one unit, every-time the Trojan is triggered using the Trojan activation pattern (HTAP). The Trojan delivers at the payload (MUX) only after reaching the maximum count value (R). The circuit tampered with Sequential Trojan will show the intended malfunction, key extraction in TAAL attacks, only upon applying the activation pattern successively R-times to the circuit.

Figure 5.(a) shows the *TAAL attack* using a sequential Trojan. The trigger consists of a p-input AND gate and a R-bit counter. The finite-state machine (FSM) of the counter is shown in Figure 5.(b).

The FSM goes to the next states when $en = 1$; otherwise, it returns to the initial state, S_0. The counter produces an output of 1, once *en* is hold to 1 consecutively R clock cycles, as it takes $(R-1)$ cycles to reach S_{R-1}. Note that this sequential Trojan can be modeled as R number of combinational Trojans. An adversary can also design a different sequential Trojan, already in the literature, to launch the *TAAL attack*. The sequential Trojan increases the complexity compared to a combinational Trojan as it manifests its effect to the payload only after the sequence of repeated application of trigger inputs. Only the adversary has the knowledge regarding the maximum counter value making it very difficult for detection.

Note that the Trojan flip-flops (*FFs*) cannot be a part of the scan chain, since a Trojan is implanted after the test pattern generation (if this process is performed at the design house). Any modifications in the scan chain will be detected with a stuck-at fault test. As the Trojan circuitry is extremely small (one counter and AND gate for the Trigger), it is possible to verify the proper operation using a functional test (i.e., applying the Trigger pattern R times and observe the response). There is no need to add Trojan *FFs* in the scan chain and perform scan tests. Now, one can argue that test patterns are generated at the foundry, and Trojan *FFs* are a part of the scan chain. In such a situation, the scan shift will not change the state of the original *FFs* in the circuit. However, it is highly unlikely that an adversary will add these malicious *FFs* in the scan chains, as one can easily find the response of the combinational part of the Trojan and determine tampering.

4.3 Design for an Analog/RF Trojan

Analog Trojans [39, 48, 96] can also be used to launch different TAAL attacks. In this section, we provide a brief description of different analog Trojans. The payload can be implemented using a multiplexer or an OR gate (see Figures 2-3) like a combinational or sequential hardware Trojan. The implementation of the trigger, however, be different depending on the Trojan design. The analog or RF-leaking Trojans can detect an extremely rare event with just a handful of transistors added to the circuit. Moreover, analog techniques and characteristics to design the stealthy trigger for Trojans are usually difficult to uncover due to their small footprint [96]. The authors used the switched capacitor to design a trigger circuit, which is activated when the frequency of toggling on a victim wire goes above a certain threshold. When the wire toggles frequently, charge accumulates on the capacitor faster than it leaks away, eventually the voltage of the capacitor rises above the threshold. This deploys the payload to cause intentional malicious activity. A similar notion is utilized to introduce triggers that are activated after some delay or operate on a specific voltage threshold [48]. Kison et al. exploited the capacitor coupling in sub-micron process technologies to design Trojans [39]. This technique uses rerouting and extending existing layout tracks to increase the capacitive coupling between a victim and an aggressor wire in a way that a low to high transition on the aggressor can adequately affect the victim wire and flip its digital value. Similarly, RF-leaking Trojans leak the information through the Trojan-induced channel without affecting the legitimate signal/channel [18, 73].

5 ANALYSIS

The hardware Trojans presented in Section 4 pose a unique challenge to the SoC designers for securing their designs.

5.1 Complexity Analysis

In this section, we show that an adversary can implement a Trojan in a very large number of ways, and it is practically infeasible to detect all of them with absolute certainty. We choose six benchmark circuits from ISCAS'85 benchmark suites [12] to show the complexity of Trojan detection even for these small benchmark circuits.

Table 1. Circuit parameters.

Benchmarks	# Gates	Key Size ($\|K\|$)	# Total Lines ($N+M$)	# Net Lines (N)	# Test Patterns	Fault coverage
C432	160	30	349	233	58	100%
C499	202	30	491	226	78	100%
C880	383	30	594	350	86	100%
C1908	880	30	552	223	83	100%
C3540	1669	83	1826	1114	173	100%
C6288	2416	128	5621	1335	77	100%

Table 1 shows the design details for different locked benchmark circuits. The number of logic gates and key size for these circuits is shown in Columns 2 and 3. The number of key bits is selected in such a way that the total area overhead does not exceed 5%. However, for industrial design with millions of gates, the key gates will merely add any overhead. Column 4 represents the total number of nets in these circuits (see Equation 1). The number of nets that are not affected by key gates is shown in Column 5. A key gate is selected by analyzing the netlist and forward tracing is performed till the primary output(s) is reached. Simultaneously, removing all the nodes that belongs to these paths from the overall list of $N + M$. Note that this includes any fanout branches as well while performing forward tracing. This step is repeated for all the key gates to get the nodes that are not impacted by the key values. Note that these nets cannot be selected for trigger inputs. The manufacturing test patterns are generated using Synopsys TetraMax Automatic Test Pattern Generation (ATPG) tool [75] with targeted 100% fault coverage (Columns 6-7). For the C432 benchmark, we insert 30 key gates randomly in the netlist with 160 logic gates. There are 349 nets in the netlist, out of which 233 nets can be selected for the Trojan trigger as the remaining nets are affected by the key gates. The TetraMax ATPG tool generates 58 stuck-at fault patterns and reports 100% fault coverage. An adversary will use these test patterns to design the Trojans such that they are not activated during the manufacturing tests. A similar analysis can be performed for all other benchmark circuits through the details mentioned in respective rows.

Table 2. Number of hardware Trojans for launching TAAL attacks.

Benchmarks	Type-2 Trojan		Type-3 Trojan		Type-4 Trojan	
	AT_2	VT_2	AT_3	VT_3	AT_4	VT_4
C432	1.08×10^5	1.04×10^5	1.66×10^7	1.43×10^7	1.91×10^9	1.34×10^9
C499	1.02×10^5	0.27×10^5	1.52×10^7	2.11×10^6	1.69×10^9	1.21×10^8
C880	2.44×10^5	2.25×10^5	5.67×10^7	4.80×10^7	9.83×10^9	7.35×10^9
C1908	0.99×10^5	0.96×10^5	1.46×10^7	1.33×10^7	1.60×10^9	1.27×10^9
C3540	2.48×10^6	2.35×10^6	1.84×10^9	1.57×10^9	1.02×10^{12}	0.74×10^{12}
C6288	3.56×10^6	3.50×10^6	3.17×10^9	3.00×10^9	2.11×10^{12}	1.82×10^{12}

Table 2 shows the number of combinational hardware Trojans that can be designed to perform *TAAL* attacks (mentioned in Section 3) for different benchmark circuits. The upper bound (see Equation 2) for all possible Trojans that can be inserted in the circuit is denoted in Column 2, 4, and 6. Out of all possible Trojans, the valid Trojans that will not be detected during manufacturing tests are shown in Columns 3, 5, and 7. For C432 benchmark circuit, the total number of *Type-2* Trojans

is 1.08×10^5, whereas, the number of valid Trojans is 1.0×10^5. The number of Trojans increases exponentially with the increase of the Trojan type (p). Note that AT_p and VT_p are in the same order, which gives an adversary to select a Trojan of its choice from a large collection. It is worthwhile to mention that an adversary needs to choose a Trojan whose triggers are selected from the rare nodes such that it does not get activated during normal operation. However, it is not necessary to impose this condition for designing a sequential Trojan, as it is highly unlikely that a particular trigger condition will arrive R times consecutively during the normal operation of a chip.

Note that the result in Table 2 does not show the complexity for sequential hardware Trojans as the ISCAS'85 benchmark circuits are combinational in nature. This table is intended to show the number of possible combinational Trojans, resulted from an effective N number of nets. One can easily extend the results for industrial designs, where an adversary can insert any type of combinational, sequential, or analog Trojans to perform a TAAL attack. Note that we choose small ISCAS'85 benchmark circuits instead of larger benchmarks (e.g., opencores) to demonstrate the difficulty of detecting all the Trojans even for smaller circuits. It is obvious to infer that the difficulty will increase for larger circuits. The number of all possible valid Trojans is in the order of 10^{12}, which is so large even for a small benchmark circuit, *C6288*, with 2416 gates, when considering only four trigger inputs. If we show that detecting all possible valid Trojans for a very small benchmark is a complex problem, it will be sufficient to maintain that complexity for large circuits.

5.2 Overhead Analysis

The area for a hardware Trojan can be varied based on the trigger inputs. A *Type-p* combinational Trojan consists of an AND gate with p-trigger inputs and a multiplexer or an OR gate as payload. For a sequential Trojan, it is necessary to add a R-bit counter along with a p-input AND gate to implement the trigger. This subsection presents the area and power overhead for different ITC'99 benchmark circuits [22] locked with a 128-bit key. The simulation is performed by using Synopsys design compiler [23] with 32nm technology [50]. A single sequential hardware Trojan of *Type-2* to Type-4 is added to each benchmark circuits for launching the *T3 Type* TAAL attack. The output of a single trigger is routed to 128 payload locations (OR gates). This is the worst-case area overhead, as we often need less than 128 OR gates to expose all the 128 key bits. For example, shown in Figure 3.(c), we need only one OR gate to determine two key bits (e.g., K_1 and K_2). For a given benchmark circuit, A_O represents the original circuit area, whereas A_T represents the area of its Trojan inserted version. The area overhead (AO) is computed using the following formula:

$$AO = \frac{A_T - A_O}{A_O} \times 100\% \tag{4}$$

Table 3 shows the area overhead analysis for different ITC'99 benchmark circuits. The number of logic gates and flip-flops (FFs) for each synthesized benchmark circuits are shown in Columns 2 and 3, respectively. The Type of Trojan is represented in Column 4. It follows the same definition of *Type-p* Trojan, where p represents the number of trigger inputs. Columns 5 to 8 indicate the percentage area overhead (computed using Equation 4). The Trojan consists of a counter and a AND gate as the trigger, and 128 OR gate as the payload. The R represents the maximum count for the counter. For example, R = 8 means that the trigger has a maximum count of 8.

We observe an area overhead of less than 5% (e.g. b14 with only 2064 gates and 215 FFs) for a small benchmark circuit. However, it becomes significantly small for larger benchmark circuits. Considering *b19*, the percentage area overhead is 0.13% for *Type-2* sequential Trojan with $R = 8$. The area overhead remains almost constant even with an increased count (e.g. $R = 16$) and different Trojan Types for a large benchmark circuit. A similar analysis can be performed for all

Table 3. Area overhead for ITC'99 benchmark circuits.

Benchmarks	# Gates	# FFs	Trojan Type	Area Overhead (%)			
				R=2	R=4	R=8	R=16
b14	2064	215	Type-2	4.03	4.23	4.41	4.56
			Type-3	4.05	4.23	4.41	4.59
			Type-4	4.06	4.27	4.41	4.62
b15	2722	418	Type-2	3.02	3.16	3.30	3.41
			Type-3	3.03	3.17	3.30	3.43
			Type-4	3.04	3.19	3.30	3.45
b20	4190	430	Type-2	1.92	2.01	2.10	2.17
			Type-3	1.93	2.02	2.10	2.19
			Type-4	1.94	2.03	2.10	2.20
b21	4225	430	Type-2	1.90	1.99	2.08	2.15
			Type-3	1.91	1.99	2.08	2.16
			Type-4	1.91	2.01	2.06	2.17
b17	8629	1328	Type-2	0.89	0.93	0.97	1.00
			Type-3	0.90	0.94	0.97	1.01
			Type-4	0.90	0.94	0.97	1.02
b18	39845	3168	Type-2	0.24	0.25	0.26	0.27
			Type-3	0.24	0.25	0.26	0.27
			Type-4	0.24	0.25	0.26	0.27
b19	78076	6337	Type-2	0.12	0.12	0.13	0.13
			Type-3	0.12	0.12	0.13	0.13
			Type-4	0.12	0.12	0.13	0.13

the benchmark circuits as well. Note that the majority of the overhead results from the payload as we require 128 OR gates to determine 128 key bits.

Table 4 shows the power overhead analysis for the same benchmark circuits. We have computed two overhead for the dynamic power and the leakage power using Equation 5 and Equation 6, respectively. As the Trojan remains quiet most of the time unless triggered, leakage power overhead is of the Trojan designers' concern so that it evades detection.

$$DPO = \frac{DP_T - DP_O}{DP_O} \times 100\% \tag{5}$$

where, DP_T and DP_O represent the dynamic power of the Trojan inserted and Trojan free circuits, respectively.

$$SPO = \frac{SP_T - SP_O}{SP_O} \times 100\% \tag{6}$$

where, SP_T, SP_O represent the leakage power of the Trojan inserted and Trojan free circuits, respectively.

Columns 1 and 2 of Table 4 represent different benchmark circuits and Trojan types. Columns 3-6 show dynamic power overhead with a sequential Trojan with R =2, 4, 8 and 16, respectively. On the other hand, Columns 7-10 show leakage power overhead with a sequential Trojan with R =2, 4, 8 and 16, respectively. For example, For a small benchmark circuit, we observe a dynamic power overhead is around 10% (e.g. b15) and leakage power overhead is around 7% (e.g. b14). However, for larger benchmark circuits, the power overhead becomes very small. The percentage dynamic power and leakage power overhead for b19 are 0.10% and 0.22% for Type-2 sequential Trojan with

Table 4. Power overhead for benchmark circuits.

Benchmarks	Trojan Types	Dynamic Power Overhead (%)				Leakage Power Overhead (%)			
		R=2	*R=4*	*R=8*	*R=16*	*R=2*	*R=4*	*R=8*	*R=16*
b14	*Type-2*	9.61	8.46	8.63	8.86	6.02	6.63	6.89	7.05
	Type-3	9.70	8.34	8.60	8.80	5.91	6.62	6.77	7.03
	Type-4	9.63	8.38	8.52	8.70	5.95	6.64	6.82	7.05
b15	*Type-2*	10.22	8.99	9.18	9.42	4.78	5.27	5.48	5.60
	Type-3	10.31	8.87	9.14	9.36	4.69	5.26	5.37	5.58
	Type-4	10.24	8.90	9.05	9.24	4.73	5.28	5.42	5.60
b20	*Type-2*	7.42	6.53	6.67	6.84	3.01	3.32	3.45	3.53
	Type-3	7.49	6.44	6.64	6.80	2.96	3.31	3.39	3.52
	Type-4	7.44	6.47	6.58	6.71	2.98	3.33	3.41	3.53
b21	*Type-2*	7.67	6.75	6.89	7.07	2.99	3.30	3.43	3.51
	Type-3	7.74	6.66	6.86	7.03	2.94	3.29	3.36	3.49
	Type-4	7.69	6.68	6.80	6.94	2.96	3.30	3.39	3.51
b17	*Type-2*	3.36	2.96	3.02	3.10	1.52	1.67	1.74	1.78
	Type-3	3.39	2.92	3.01	3.08	1.49	1.67	1.71	1.77
	Type-4	3.37	2.93	2.98	3.04	1.50	1.68	1.72	1.78
b18	*Type-2*	0.24	0.21	0.21	0.22	0.38	0.42	0.44	0.45
	Type-3	0.24	0.21	0.21	0.22	0.37	0.42	0.43	0.44
	Type-4	0.24	0.21	0.21	0.21	0.38	0.42	0.43	0.45
b19	*Type-2*	0.12	0.10	0.11	0.11	0.20	0.22	0.22	0.23
	Type-3	0.12	0.10	0.11	0.11	0.19	0.21	0.22	0.23
	Type-4	0.12	0.10	0.10	0.11	0.19	0.22	0.22	0.23

R=4, respectively. Note that for a large circuit, an increased count (e.g. R=16) and different Types of a Trojan will not have a significant impact on dynamic power and leakage power overhead.

6 FUTURE RESEARCH DIRECTION FOR SECURE LOGIC LOCKING

The security of a logic locking technique can be tied together with the hardware Trojan detection problem. Developing a SAT-registrant logic locking is not sufficient enough to prevent IC overproduction or to protect IPs. It is required to address the detection of Trojans inserted at an untrusted manufacturing site. Researchers have already proposed different techniques to detect and prevent hardware Trojans. The detection methods can be grouped into two different categories, such as, logic testing [8, 17, 28, 42, 84], and side-channel analysis [3, 6, 7, 43, 45, 54]. On the other hand, prevention methods can be categorized as design-for-trust measures [16, 49, 57, 64, 91] and split manufacturing [61, 81, 89].

Logic testing by applying stimuli to primary inputs (PIs) and observe responses at primary outputs (POs) can be used to detect these Trojans [8, 10, 17, 28, 42, 70]. The decision is being made whether a chip is tampered with a hardware Trojan by observing a mismatch between the observed and expected responses. Note that the accuracy of the detection process does not depend on the manufacturing process variations. However, the detection will be extremely difficult as it is practically impossible to detect all types of combinational Trojans. In addition, it is not feasible to trigger a sequential Trojan, as it requires to apply the same trigger pattern at the input R times.

Side-channel information, such as, power [90], temperature [51], delay [34], and radiation [29] can be used to detect a hardware Trojan. The side-channel fingerprinting technique for detecting Analog/RF hardware Trojans in a wireless cryptographic IC has also been proposed [35, 46]. These

detection methods rely on the availability of Trojan-free golden circuits for creating Trojan free signature. It can be very difficult to acquire a golden sample as all the chips may have Trojans. Path delay-based testing has limitations on detecting a hardware Trojan as long as the Trojan is inactive. Note that activating a Trojan is challenging as an adversary can select a hard to activate Trojan. In addition, the path delay does not depend on the size of the trigger circuit or the number of payloads attached with each trigger as the Trojan remains quiet during the logic testing. In addition, process and environmental variations may mask the side-channel leakage, when a Trojan circuitry is small.

While dedicated towards hardware Trojan detection, researchers propose different measures to prevent a Trojan from being inserted into the design in the first place. These solutions involved characterization of ring-oscillator [57], shadow registers [43], and delay elements [62] to detect the delay deviation caused by hardware Trojans. Reducing the rare signal in the circuitry is another proposed method for designers to reduce the risk of being implanted with a Trojan [64, 106]. Camouflage fill techniques [20, 59] to create indistinguishable layouts for different gates by adding dummy contacts and connections can prevent the attacker from extracting a correct gate-level netlist of a circuit for Trojan insertion. Xiao et al. proposed to fill all the unused spaces using filler cells so that an untrusted foundry cannot insert a Trojan [69, 91]. However, this direction still lacks any firm solution as more emphasis is observed in Trojan detection.

Split Manufacturing can be an effective way to thwart Hardware Trojan insertion at an untrusted foundry. In split manufacturing, the production of ICs is carried out in two different foundries [32]. The design is divided into two parts – Front End of Line (FEOL) and Back End of Line (BEOL). An untrusted foundry is provided with the FEOL design, which contains partial information regarding the design that requires complex steps for fabricating and involves higher cost. Fabrication of BEOL does not incorporate complex fabrication steps and can be done by a smaller trusted foundry. The untrusted foundry sends the fabricated wafers directly to the smaller foundry for the complete fabrication. This way the untrusted foundry can be restricted to make any Trojan based modification as it does not have the complete information regarding the design. However, several attacks undermining the security achieved through split manufacturing have also been proposed in past [47, 88, 94].

Recent research contributions showed that machine learning and image processing can also be incorporated to detect hardware Trojans in the chip. Vashistha et al. presented Trojan scanner [83], which uses a trusted GDSII layout (golden layout) and scanning electron microscope (SEM) images to identify the malicious modifications made in the netlist during the manufacturing of a circuit. A unique descriptor for each type of gate is prepared based on different features using computer vision algorithms along with a machine-learning model of a golden layout and SEM images of an IC under authentication. These descriptors, when compared to each other can detect any modifications either in the form of additional gates or modified gates which might raise the suspicion for a potential hardware Trojan. Moreover, Trojan scanner also presents the trade-off between the accuracy and SEM parameters. The authors demonstrated the effectiveness of the scheme using a smart card die as a test sample (generally manufactured with 90 nm technology). It is yet to be validated its effectiveness of detection when a chip is fabricated using recent technology nodes (10 nm and beyond).

Research groups have also investigated vulnerabilities in the analog/RF front-end of a wireless device that can facilitate hardware Trojan attacks. Subramani et al. proposed defense to distinguish between channel-induced and Trojan-induced impact on the legitimate signal through its traits [73]. Acknowledging excessive toggling activity on a victim wire of Trojans such as A2 [96], Hou et al. proposed to add on-chip monitors to measure switching activity on potential victim wires during an adjustable time period and raises an alarm if such activity goes above a certain threshold [30]. This method can be effective for detecting A2 Trojan which comprises of a capacitor as a trigger.

Reverse engineering based detection methods have also been proposed by sorting trace lengths to realize the minimum capacitance needed for such Trojans [27]. However, it is possible to mitigate this requirement by an adversary, e.g. using multiple layers/higher voltages to evade detection by this kind of approaches.

Despite significant research have been performed on detecting hardware Trojans, we still lack efficient and accurate methods for modeling them and generating tests for their detection. Once the detection of hardware Trojans is ensured, an SoC designer can choose a SAT-resistant logic locking to prevent IC overproduction and IP piracy.

7 CONCLUSION

In this paper, we have demonstrated the vulnerability of logic locking techniques through a set of tampering attacks with hardware Trojans. Three types of proposed *TAAL attacks* can defeat any logic locking techniques that rely on storing the secret key in a tamper-proof memory. In *T1 type TAAL Attack*, we showed how an adversary could extract the key from a locked netlist without knowing the details of the logic locking technique used to protect the circuit. For *T2 type* and *T3 type TAAL* attacks, the complexity of detecting an attack has been improved. Only the attacker knows the specific values that can lead to key extraction, increasing the identification of a *TAAL* attack. To launch a *TAAL* attack, we develop models for combinational and sequential hardware Trojans. We also proposed an algorithm to design a hardware Trojan that cannot be detected by manufacturing tests. The results depict the range of Trojans selected by an adversary, which has a very high order of magnitude. Finally, we describe relevant detection and avoidance strategies for hardware Trojans to make logic locking secure.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under grant number CNS-1755733, and the United States Air Force/Air Force Materiel Command (USAF/AFMC) under grant AF-FA8650-19-1-1707. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF and USAF/AFMC.

REFERENCES

[1] Sally Adee. 2008. The Hunt for the Kill Switch. *IEEE Spectrum* 45, 5 (2008), 34–39.
[2] Age Yeh. 2012. Trends in the global IC design service market. DIGITIMES Research. (2012).
[3] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. 2007. Trojan Detection Using IC Fingerprinting. In *Proc. IEEE Symp. Security and Privacy (SP)*. 296–310.
[4] Yousra Alkabani and Farinaz Koushanfar. 2007. Active Hardware Metering for Intellectual Property Protection and Security.. In *USENIX security symposium*. 291–306.
[5] Yousra Alkabani, Farinaz Koushanfar, and Miodrag Potkonjak. 2007. Remote activation of ICs for piracy prevention and digital right management. In *Proc. of IEEE/ACM int. conf. on Computer-aided design*. 674–677.
[6] Mainak Banga and Michael S Hsiao. 2008. A Region Based Approach for the Identification of Hardware Trojans. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*. 40–47.
[7] Mainak Banga and Michael S Hsiao. 2009. A Novel Sustained Vector Technique for the Detection of Hardware Trojans. In *Proceedings of International Conference VLSI Design*. 327–332.
[8] Mainak Banga and Michael S Hsiao. 2011. Odette: A Non-Scan Design-for-Test Methodology for Trojan Detection in ICs. In *Proc. IEEE Int. Symp. Hardware-Oriented Security and Trust*. 18–23.
[9] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers* 27, 1 (2010), 66–75.
[10] Swarup Bhunia, Michael S Hsiao, Mainak Banga, and Seetharam Narasimhan. 2014. Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proc. IEEE* 102, 8 (2014), 1229–1247.
[11] Swarup Bhunia and Mark Tehranipoor. 2018. *Hardware Security: A Hands-on Learning Approach*. Morgan Kaufmann.
[12] David Bryan. 1985. The ISCAS'85 benchmark circuits and netlist format. *North Carolina State University* 25 (1985).

[13] Michael Bushnell and Vishwani Agrawal. 2004. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Vol. 17. Springer Science & Business Media.

[14] Encarnacin Castillo, Uwe Meyer-Baese, Antonio García, Luis Parrilla, and Antonio Lloris. 2007. IPP@ HDL: efficient intellectual property protection scheme for IP cores. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15, 5 (2007), 578–591.

[15] Rajat Subhra Chakraborty and Swarup Bhunia. 2008. Hardware protection and authentication through netlist level obfuscation. In *Proc. of IEEE/ACM International Conference on Computer-Aided Design*. 674–677.

[16] Rajat Subhra Chakraborty and Swarup Bhunia. 2009. Security Against Hardware Trojan Through a Novel Application of Design Obfuscation. In *Proc. Int. Conf. Computer-Aided Design*. 113–116.

[17] Rajat Subhra Chakraborty, Francis G Wolff, Somnath Paul, Christos A Papachristou, and Swarup Bhunia. 2009. MERO: A Statistical Approach for Hardware Trojan Detection. In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. 396–410.

[18] Doohwang Chang, Bertan Bakkaloglu, and Sule Ozev. 2015. Enabling unauthorized RF transmission below noise floor with no detectable impact on primary communication performance. In *VLSI Test Symposium (VTS)*. 1–4.

[19] Edoardo Charbon. 1998. Hierarchical watermarking in IC design. In *Proc. of the Custom Integrated Circuits*. 295–298.

[20] Ronald P Cocchi, James P Baukus, Lap Wai Chow, and Bryan J Wang. 2014. Circuit camouflage integration for hardware IP protection. In *Proceedings of Annual Design Automation Conference*. 1–5.

[21] Gustavo K Contreras, Md Tauhidur Rahman, and Mohammad Tehranipoor. 2013. Secure split-test for preventing IC piracy by untrusted foundry and assembly. In *IEEE International symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFTS)*. 196–203.

[22] Scott Davidson. 1999. ITC'99 benchmark circuits-preliminary results. In *International Test Conference 1999. Proceedings (IEEE Cat. No. 99CH37034)*. IEEE Computer Society, 1125–1125.

[23] DC Ultra: Concurrent Timing, Area, Power, and Test Optimization. 2019. [Online] Available at: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html. (2019).

[24] Ujjwal Guin, Qihang Shi, Domenic Forte, and Mark M Tehranipoor. 2016. FORTIS: a comprehensive solution for establishing forward trust for protecting IPs and ICs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 21, 4 (2016), 63.

[25] U. Guin, Ziqi Zhou, and A. Singh. 2017. A novel design-for-security (DFS) architecture to prevent unauthorized IC overproduction. In *Proc. of the IEEE VLSI Test Symposium (VTS)*. 1–6.

[26] Ujjwal Guin, Ziqi Zhou, and Adit Singh. 2018. Robust design-for-security architecture for enabling trust in IC manufacturing and test. *Trans. on Very Large Scale Integration (VLSI) Systems* 26, 5 (2018), 818–830.

[27] Xiaolong Guo, Huifeng Zhu, Yier Jin, and Xuan Zhang. 2019. When Capacitors Attack: Formal Method Driven Design and Detection of Charge-Domain Trojans. In *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*.

[28] Syed Kamran Haider, Chenglu Jin, Masab Ahmad, Devu Shila, Omer Khan, and Marten van Dijk. 2017. Advancing the State-of-the-Art in Hardware Trojans Detection. *IEEE Transactions on Dependable and Secure Computing* (2017).

[29] Jiaji He, Yiqiang Zhao, Xiaolong Guo, and Yier Jin. 2017. Hardware Trojan Detection Through Chip-Free Electromagnetic Side-Channel Statistical Analysis. *IEEE Trans. Very Large Scale Integration Sys.* 25, 10 (2017), 2939–2948.

[30] Yumin Hou, Hu He, Kaveh Shamsi, Yier Jin, Dong Wu, and Huaqiang Wu. 2018. R2d2: Runtime reassurance and detection of A2 Trojan. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 195–200.

[31] Jiawei Huang and John Lach. 2008. IC activation and user authentication for security-sensitive systems. In *IEEE International Workshop on Hardware-Oriented Security and Trust*. 76–80.

[32] Intelligence Advanced Research Projects Activity. 2011. Trusted Integrated Chips (TIC) Program. (2011).

[33] Richard Wayne Jarvis and Michael G McIntyre. 2007. Split manufacturing method for advanced semiconductor circuits. (2007). US Patent 7,195,931.

[34] Y. Jin and Y. Makris. 2008. Hardware Trojan Detection Using Path Delay Fingerprint. In *Proc. HOST*. 51—-57.

[35] Yier Jin and Yiorgos Makris. 2010. Hardware Trojans in wireless cryptographic ICs. *IEEE Design & Test of Computers* (2010), 26–35.

[36] Andrew B Kahng, John Lach, William H Mangione-Smith, Stefanus Mantik, Igor L Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. 2001. Constraint-based watermarking techniques for design IP protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20, 10 (2001), 1236–1252.

[37] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. 2010. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer* 43, 10 (2010), 39–46.

[38] Soroush Khaleghi, Kai Da Zhao, and Wenjing Rao. 2015. IC piracy prevention via design withholding and entanglement. In *Asia and South Pacific Design Automation Conference*. 821–826.

[39] Christian Kison, Omar Mohamed Awad, Marc Fyrbiak, and Christof Paar. 2019. Security Implications of Intentional Capacitive Crosstalk. *Trans. on Information Forensics and Security* (2019).

[40] Farinaz Koushanfar and Gang Qu. 2001. Hardware metering. In *Proceedings of Design Automation Conference*. 490–493.

[41] Yu-Wei Lee and Nur A Touba. 2015. Improving logic obfuscation via logic cone analysis. In *Latin-American Test Symposium (LATS)*. 1–6.
[42] Nicole Lesperance, Shrikant Kulkarni, and Kwang-Ting Cheng. 2015. Hardware Trojan Detection Using Exhaustive Testing of k-bit Subspaces. In *Proc. of Asia and South Pacific Design Automation Conf. (ASP-DAC)*. 755–760.
[43] Jie Li and John Lach. 2008. At-speed delay characterization for IC authentication and Trojan horse detection. In *IEEE International Workshop on Hardware-Oriented Security and Trust*. 8–14.
[44] Bao Liu and Brandon Wang. 2014. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6.
[45] Yu Liu, Ke Huang, and Yiorgos Makris. 2014. Hardware Trojan Detection Through Golden Chip-Free Statistical Side-Channel Fingerprinting. In *Proc. of Design Automation Conference*.
[46] Yu Liu, Georgios Volanis, Ke Huang, and Yiorgos Makris. 2015. Concurrent hardware Trojan detection in wireless cryptographic ICs. In *International Test Conference (ITC)*. 1–8.
[47] Jonathon Magaña, Daohang Shi, Jackson Melchert, and Azadeh Davoodi. 2017. Are proximity attacks a threat to the security of split manufacturing of integrated circuits? *Trans. on Very Large Scale Integration (VLSI) Systems* (2017), 3406–3419.
[48] Karthikeyan Nagarajan, Mohammad Nasim Imtiaz Khan, and Swaroop Ghosh. 2019. ENTT: A family of emerging NVM-based trojan triggers. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 51–60.
[49] Xuan Thuy Ngo, Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. 2015. Linear Complementary Dual Code Improvement to Strengthen Encoded Circuit Against Hardware Trojan Horses. In *Proc. IEEE Int. Symp. Hardware Oriented Security and Trust*. 82–87.
[50] Synopsys 32/28 nm Generic Library for Teaching IC Design. Accessed 2019. [online] Available at: https://www.synopsys.com/community/universityprogram/teaching-resources.html. (Accessed 2019).
[51] Abdullah Nazma Nowroz, Kangqiao Hu, Farinaz Koushanfar, and Sherief Reda. 2014. Novel Techniques for High-Sensitivity Hardware Trojan Detection Using Thermal and Power Maps. *Trans. Computer-Aided Design of Integrated Circuits and Systems* 33, 12 (2014), 1792–1805.
[52] Stephen M Plaza and Igor L Markov. 2015. Solving the third-shift problem in IC piracy with test-aware logic locking. *Trans. on Computer-Aided Design of Integrated Circuits and Systems* 34, 6 (2015), 961–971.
[53] Gang Qu and Miodrag Potkonjak. 2007. *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media.
[54] Reza Rad, Jim Plusquellic, and Mohammad Tehranipoor. 2008. Sensitivity Analysis to Hardware Trojans Using Power Supply Transient Signals. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*. 3–7.
[55] MT Rahman, S Tajik, MS Rahman, M Tehranipoor, and N Asadizanjani. 2019. *The key is left under the mat: On the inappropriate security assumption of logic locking schemes*. Technical Report.
[56] Md Tauhidur Rahman, Domenic Forte, Quihang Shi, Gustavo K Contreras, and Mohammad Tehranipoor. 2014. CSST: an efficient secure split-test for preventing IC piracy. In *IEEE 23rd North Atlantic Test Workshop*. 43–47.
[57] Jeyavijayan Rajendran, Vinayaka Jyothi, Ozgur Sinanoglu, and Ramesh Karri. 2011. Design and Analysis of Ring Oscillator Based Design-for-Trust Technique. In *Proc. of VLSI Test Symp.* 105–110.
[58] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2012. Security analysis of logic obfuscation. In *Proceedings of Annual Design Automation Conference*. 83–89.
[59] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security analysis of integrated circuit camouflaging. In *Proc. of ACM SIGSAC conference on Computer & communications security*. 709–720.
[60] Jeyavijayan Rajendran, Huan Zhang, Chi Zhang, Garrett S Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2015. Fault analysis-based logic encryption. *IEEE Transactions on computers* 64, 2 (2015), 410–424.
[61] J. J. V. Rajendran, Ozgur Sinanoglu, and Ramesh Karri. 2013. Is Split Manufacturing Secure?. In *Proc. Conf. Design, Automation and Test in Europe (DATE)*. 1259–1264.
[62] Abishek Ramdas, Samah Mohamed Saeed, and Ozgur Sinanoglu. 2014. Slack removal for enhanced reliability and trust. In *Int. Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. 1–4.
[63] Jarrod A Roy, Farinaz Koushanfar, and Igor L Markov. 2008. EPIC: Ending piracy of integrated circuits. In *Proceedings of the conference on Design, automation and test in Europe*. 1069–1074.
[64] Hassan Salmani, Mohammad Tehranipoor, and Jim Plusquellic. 2012. A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. *IEEE Trans. Very Large Scale Integration Sys.* (2012), 112–125.
[65] Anirban Sengupta and Saraju P Mohanty. 2018. Functional obfuscation of DSP cores using robust logic locking and encryption. In *Computer Society Annual Symposium on VLSI (ISVLSI)*. 709–713.
[66] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z Pan, and Yier Jin. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *Int. Symposium on Hardware Oriented Security and Trust (HOST)*. 95–100.
[67] Haoting Shen, Navid Asadizanjani, Mark Tehranipoor, and Domenic Forte. 2018. Nanopyramid: An Optical Scrambler Against Backside Probing Attacks. In *Proc. Int. Symposium for Testing and Failure Analysis(ISTFA)*. 280.

[68] Yuanqi Shen and Hai Zhou. 2017. Double dip: Re-evaluating security of logic encryption algorithms. In *Proceedings of the Great Lakes Symposium on VLSI*. 179–184.
[69] Qihang Shi, Kan Xiao, Domenic Forte, and Mark M Tehranipoor. 2017. Obfuscated built-in self-authentication. In *Hardware Protection through Obfuscation*. Springer, 263–289.
[70] O. Sinanoglu, N. Karimi, J. Rajendran, R. Karri, Y. Jin, K. Huang, and Y. Makris. 2013. Reconciling the IC Test and Security Dichotomy. In *Proc. of IEEE European Test Symp.*
[71] Deepak Sirone and Pramod Subramanyan. 2018. Functional Analysis Attacks on Logic Locking. *arXiv preprint arXiv:1811.12088* (2018).
[72] Cynthia Sturton, Matthew Hicks, David Wagner, and Samuel T King. 2011. Defeating UCI: Building stealthy and malicious hardware. In *2011 IEEE Symposium on Security and Privacy*. IEEE, 64–77.
[73] Kiruba Sankaran Subramani, Angelos Antonopoulos, Ahmed Attia Abotabl, Aria Nosratinia, and Yiorgos Makris. 2017. ACE: Adaptive channel estimation for detecting analog/RF trojans in WLAN transceivers. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 722–727.
[74] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 137–143.
[75] Synopsys Inc., Mountain View, CA, USA. 2017. TetraMAX ATPG: Automatic Test Pattern Generation. (2017).
[76] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers* 27, 1 (2010).
[77] Mohammad Tehranipoor, Hassan Salmani, Xuehui Zhang, Michel Wang, Ramesh Karri, Jeyavijayan Rajendran, and Kurt Rosenfeld. 2011. Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges. *Computer* (2011).
[78] Mohammad Tehranipoor and Cliff Wang. 2011. *Introduction to hardware security and trust*. Springer Science & Business Media.
[79] Mark Mohammad Tehranipoor, Ujjwal Guin, and Domenic Forte. 2015. Counterfeit integrated circuits. In *Counterfeit Integrated Circuits*. Springer, 15–36.
[80] Randy Torrance and Dick James. 2009. The state-of-the-art in IC reverse engineering. In *International Workshop on Cryptographic Hardware and Embedded Systems*. 363–381.
[81] Kaushik Vaidyanathan, Bishnu P Das, and Larry Pileggi. 2014. Detecting Reliability Attacks During Split Fabrication Using Test-Only BEOL Stack. In *Proc. of Design Automation Conf.* 1–6.
[82] Kaushik Vaidyanathan, Renzhi Liu, Ekin Sumbul, Qiuling Zhu, Franz Franchetti, and Larry Pileggi. 2014. Efficient and secure intellectual property (IP) design with split fabrication. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 13–18.
[83] Nidish Vashistha, Hangwei Lu, Qihang Shi, M Tanjidur Rahman, Haoting Shen, Damon L Woodard, Navid Asadizanjani, and Mark Tehranipoor. 2018. Trojan Scanner: Detecting Hardware Trojans with Rapid SEM Imaging combined with Image Processing and Machine Learning. In *Proc. Int. Symposium for Testing and Failure Analysis*. 256.
[84] Adam Waksman, Matthew Suozzo, and Simha Sethumadhavan. 2013. FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis. In *Proc. ACM SIGSAC Conf. on Computer & Communications Security*. 697–708.
[85] Huanyu Wang, Domenic Forte, Mark M Tehranipoor, and Qihang Shi. 2017. Probing attacks on integrated circuits: Challenges and research opportunities. *IEEE Design & Test* 34, 5 (2017), 63–71.
[86] Huanyu Wang, Qihang Shi, Domenic Forte, and Mark M Tehranipoor. 2019. Probing Assessment Framework and Evaluation of Antiprobing Solutions. *Transactions on Very Large Scale Integration (VLSI) Systems* 27, 6 (2019), 1239–1252.
[87] Xinmu Wang, Seetharam Narasimhan, Aswin Krishna, Tatini Mal-Sarkar, and Swarup Bhunia. 2011. Sequential hardware trojan: Side-channel aware design and placement. In *International Conference on Computer Design (ICCD)*. 297–300.
[88] Yujie Wang, Tri Cao, Jiang Hu, and Jeyavijayan Rajendran. 2017. Front-end-of-line attacks in split manufacturing. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.
[89] Yujie Wang, Pu Chen, Jiang Hu, and Jeyavijayan JV Rajendran. 2016. The Cat and Mouse in Split Manufacturing. In *Proceedings of Design Automation Conference*. 1–6.
[90] S. Wei, S. Meguerdichian, and M.Potkonjak. 2011. Malicious Circuitry Detection Using Thermal Conditioning. *Trans. of Information, Forensics and Security* 6, 3 (2011), 1136—-1145.
[91] Kan Xiao and Mohammed Tehranipoor. 2013. BISA: Built-In Self-Authentication for Preventing Hardware Trojan Insertion. In *Proc. IEEE Int. Symp. Hardware-Oriented Security and Trust*. 45–50.
[92] Yang Xie and Ankur Srivastava. 2016. Mitigating SAT attack on logic locking. In *International Conference on Cryptographic Hardware and Embedded Systems*. 127–146.
[93] Yang Xie and Ankur Srivastava. 2019. Anti-sat: Mitigating sat attack on logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 2 (2019), 199–207.
[94] Wenbin Xu, Lang Feng, Jeyavijayan JV Rajendran, and Jiang Hu. 2019. Layout recognition attacks on split manufacturing. In *Proceedings of Asia and South Pacific Design Automation Conference*. 45–50.

[95] Xiaolin Xu, Bicky Shakya, Mark M Tehranipoor, and Domenic Forte. 2017. Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks. In *Int. Conf. on Cryptographic Hardware and Embedded Systems*.

[96] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. 2016. A2: Analog malicious hardware. In *IEEE symposium on security and privacy (SP)*. 18–37.

[97] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan JV Rajendran, and Ozgur Sinanoglu. 2016. SARLock: SAT attack resistant logic locking. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 236–241.

[98] Muhammad Yasin, Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. 2015. On improving the security of logic locking. *Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2015), 1411–1424.

[99] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan JV Rajendran, and Ozgur Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*. 1601–1618.

[100] Muhammad Yasin, Abhrajit Sengupta, Benjamin Carrion Schafer, Yiorgos Makris, Ozgur Sinanoglu, and Jeyavijayan JV Rajendran. 2017. What to lock?: Functional and parametric locking. In *Proc. of Great Lakes Symposium on VLSI*. 351–356.

[101] Muhammad Yasin and Ozgur Sinanoglu. 2015. Transforming between logic locking and IC camouflaging. In *International Design & Test Symposium (IDT)*. 1–4.

[102] Qiaoyan Yu, Jaya Dofe, and Zhiming Zhang. 2017. Exploiting hardware obfuscation methods to prevent and detect hardware trojans. In *International Midwest Symposium on Circuits and Systems (MWSCAS)*. 819–822.

[103] Qiaoyan Yu, Jaya Dofe, Zhiming Zhang, and Sean Kramer. 2018. Hardware Obfuscation Methods for Hardware Trojan Prevention and Detection. In *The Hardware Trojan War*. Springer, 291–325.

[104] Dongrong Zhang, Xiaoxiao Wang, Md Tauhidur Rahman, and Mark Tehranipoor. 2018. An On-Chip Dynamically Obfuscated Wrapper for Protecting Supply Chain Against IP and IC Piracies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 11 (2018), 2456–2469.

[105] Yuqiao Zhang, Pinchen Cui, Ziqi Zhou, and Ujjwal Guin. 2019. TGA: An Oracle-less and Topology-Guided Attack on Logic Locking. In *Proc. of the ACM Workshop on Attacks and Solutions in Hardware Security Workshop (ASHES)*. 75–83.

[106] Bin Zhou, Wei Zhang, Srikanthan Thambipillai, and JKJ Teo. 2014. A low cost acceleration method for hardware Trojan detection based on fan-out cone analysis. In *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis*. 28.

[107] Ziqi Zhou, Ujjwal Guin, and Vishwani D Agrawal. 2018. Modeling and test generation for combinational hardware Trojans. In *VLSI Test Symposium (VTS)*. 1–6.