Special Session: Novel Attacks on Logic-Locking

Ayush Jain
and Ujjwal Guin
Dept. of Electrical and
Computer Engineering
Auburn University
Emails: {ayush.jain, ujjwal.guin}@auburn.edu

M Tanjidur Rahman
and Navid Asadizanjani
Dept. of Electrical and
Computer Engineering
University of Florida
{mir.rahman, nasadi}@ufl.edu

Danielle Duvalsaint
and R. D. (Shawn) Blanton
Dept. of Electrical and
Computer Engineering
Carnegie Mellon University
{dduvalsa, rblanton}@andrew.cmu.edu

***Abstract*—The outsourcing of the design and manufacturing of integrated circuits (IC) involves various untrusted entities, which can pose many security threats such as overproduction of ICs, sale of out-of-specification/rejected ICs, and piracy of Intellectual Properties (IPs). As a result, various design-for-trust techniques have been developed. Logic locking has recently gained significant interest from the research community due to its capability to provide defense against the threats from untrusted manufacturing. In logic locking, the original circuit is locked using a secret key to make it into a key-dependent circuit. However, various attacks on the extraction of secret keys associated with locking have undermined the security of logic locking techniques. Even after a decade of research, the security of logic locking is still under risk as none of the countermeasures can simultaneously provide resiliency against different attacks, such as tampering, probing, and oracle or oracle-less attacks. This paper presents an overview of novel attacks on logic locking apart from SAT-based analysis. We will present three different techniques to break a secure lock, and they are hardware Trojan based attacks, optical probing based attacks, and the ATPG oriented attacks.**

***Index Terms*—Logic Locking, Obfuscation, Hardware Trojan, Optical Probing, ATPG**

I. Introduction

In the current globalized and distributed semiconductor manufacturing and test processes, various security threats have emerged from untrusted manufacturing, such as Intellectual Property (IP) piracy/theft [1]–[4], sale of out-of-specification/rejected integrated circuits (ICs) [5], [6] and Overproduction [5], [7]–[10]. The challenges for protecting a circuit against these security threats have been the driving force for the development of different techniques to limit the amount of circuit information that can be recovered by an adversary. Logic locking emerged as a well-accepted hardware obfuscation technique to protect a secure design from different untrusted entities in the design and manufacturing process of ICs. The underlying principle of logic locking is to incorporate the additional key gates in the original netlist/design to obtain a key-dependent netlist. This secret key is loaded into the on-chip tamper-proof memory by the designer after fabrication and test. Only upon the application of the correct key, the circuit will show correct functionality, else, erroneous outputs will be observed. With the advent of Boolean Satisfiability (SAT) analysis proposed by Subramanyan et al. [11], almost all the initial locking mechanisms were broken in real-time. In recent times, researchers are focusing on achieving complete security for the logic locking techniques along with SAT resiliency. This is mainly done to explore the capabilities of an adversary as it may have many efficient methods to attack a locked design apart from performing SAT analysis.

An adversary such as an untrusted foundry who has access to most advanced equipment, such as a micro-probing station, scanning electron microscope (SEM) etc. is capable of attacking any chip with physical attacks. Moreover, an untrusted foundry can intentionally add malicious modifications to the original design without the knowledge of an SoC designer or design-house to leak the secret key. Amongst the many attacks in this direction, implanting hardware Trojans attacks [12], probing based attacks [13]–[15], oracle or oracle-less attacks (TGA [16], CLIC-A [17], [18]) can be identified as prominent attack choices for an adversary. Research groups have addressed these attacks with different defense and detection approaches, which include restricting the scan-access and preventing key leakage through scan-chain [19], [20], nanopyramid and shielding architectures to prevent probing attacks [21], [22] and optical probing based reverse engineering to detect hardware Trojans [23]. Despite many solutions to restrict these attacks, logic locking techniques have not achieved complete security against physical attacks than previously thought due to the possibility of the key extraction. As a result, currently, none of the logic locking techniques can be categorized to provide absolute defense against IP piracy.

The rest of the paper is organized as follows. The 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 II. This section also includes the model for combinational and sequential Trojan design. In Section III, we present the vulnerability of logic locking against the optical analysis. Section IV presents the overview of CLIC-A attack on different logic locking techniques along with a performance comparison to traditional SAT attack. Finally, we conclude the paper in Section V.

II. Tampering Attack on Logic Locking

The malicious modifications, described as hardware Trojans, can pose a serious threat to the logic locked circuits. A circuit can be tampered with implanting different types of hardware

Trojans in any key-based logic locked design. An adversary, who intends to obtain the correct functionality of the design can extract the secret key by triggering a hardware Trojan inserted by an untrusted foundry before fabrication. Note that an untrusted foundry has the capability to extract the netlist of a design from layout/mask information, making it feasible to implement a hardware Trojan. Since, the production tests are performed at the foundry, an adversary also has access to all the manufacturing test (e.g., stuck-at-fault and delay fault) patterns to be utilized for designing an efficient hardware Trojan. Once the valid key information is extracted out from an activated IC, an untrusted foundry can recover the original netlist for IP piracy, unlock any number of chips, and sell overproduced and defective chips in the market. As the attack is applicable to any key-based locked circuits, an adversary can undermine any secure solutions proposed so far to prevent threats originated from untrusted manufacturing.

A. Approach

A locked circuit can be targeted with three types of tampering attacks that can be launched to extract the secret key using hardware Trojans. A straightforward tampering attack directly leaks the secret key to the primary output (PO) once the Trojan is activated, whereas, the complexity of attack can be increased with dependency on the activation and propagation of the secret key to the primary output. The adversary has the freedom of choosing one of these attacks implemented using either a combinational or sequential hardware Trojan. For simplicity, these attacks are explained using a *Type-3* combinational Trojan, which consists of a 3-input AND gate which serves as the trigger (T).

The simplest form of tampering attack can be launched by an adversary who does not intend to gain knowledge regarding the security measures implemented for the circuit. Figure 1. (a) shows the proposed modifications represented through the dashed lines for launching this attack. As shown, one input of the multiplexer is the original output of the locked netlist, whereas, the other input is tapped on the wire connection between the key gate (K) and the tamper-proof memory. Under normal operation for any activated chip, the multiplexer propagates the correct circuit functionality at the output. Once the Trojan gets activated, the output of AND gate becomes 1 and the payload (P) is delivered to the primary output of the circuit using a 2-input MUX, which leads to the extraction of the secret key at the output. Note that the required number of multiplexers to extract the complete secret key is dependent on the key size.

Instead of extracting the key directly to the primary output, an adversary can also propagate it to the output. An adversary can choose a net, whose logic value is impacted by the key gate for the MUX input. For example, net n_2 can be selected as the MUX input for the circuit shown in Figure 1. (a). Upon activation of Trojan, k or $\overline{k}$ can be observed at the primary output (PO) by forcing net n_1 as logic 0 or 1 respectively. The notion of increasing the dependence of key extraction with its propagation from internal nodes results in increased

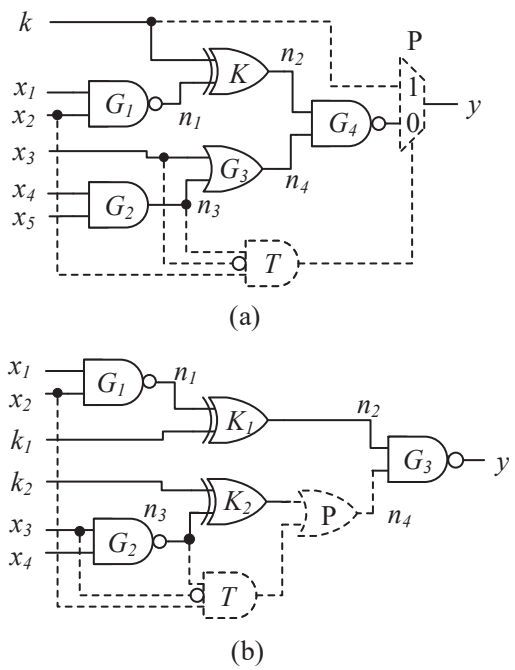

Figure 1. (a) *Tampering attack*, where a *Type-3* combinational Trojan and payload (MUX) is inserted for key extraction directly from the connection between key gate and tamper-proof memory, (b) *Tampering attack* with a *Type-3* combinational Trojan with payload as OR gate located inside the locked circuit

complexity from the simple form of tampering attack. A similar case is explained through the Figure 1. (b), where a secure logic locking technique is implemented to insert the key gates in such a way that key propagation is not possible due to the key inter-dependency using manufacturing tests [24]. For the locked netlist shown in Figure 1. (b), the propagation of the key (K_1) is prevented by inserting another key (K_2). The primary output cannot be uniquely determined unless an adversary knows either k_1 or k_2. Thus, it is necessary to help propagate one key and then determine the other. To achieve this aim, the circuit is tampered with a combinational Trojan consisting of a 3-input AND gate, which serves as the trigger (T) and an OR gate (shown using dashed lines) to deliver the payload at net n_4. The value of k_1 needs to obtained first to determine the other key k_2. The Trojan delivers the payload of logic 1 at node n_4, once it gets activated. This helps to propagate the key k_1 at the output ($\overline{k_1}$) with an input pattern of $[x_1\ x_2\ x_3\ x_4] = [1\ 1\ 0\ X]$. Once the value of k_1 is known, an adversary can perform the signal propagation analysis to find the value of k_2.

As mentioned before, the attacks are explained using a combinational Trojan for the simplicity of understanding. These proposed attacks can also be implemented using any *Type-p* sequential Trojan which delivers at its targeted payload once the Trojan is activated R times, consecutively. The design details for a combinational and sequential Trojan are discussed in the following section.

B. Design of Hardware Trojans

The primary purpose of a hardware Trojan is to modify the original functionality of a circuit to leak the secret key once activated without the knowledge of an SoC designer. It is absolutely necessary that the Trojan must not get activated during scan-based manufacturing tests. In other words, the circuit should not come across any condition during tests that activate the Trojan, which can lead to its detection. A hardware Trojan can be inserted into a circuit during its design or manufacturing stages. In relevance to logic locking, a Trojan inserted by an untrusted foundry should be considered to show the effectiveness of the attack as logic locking was proposed to address the threat from an untrusted foundry. An adversary can tamper the netlist with any category of Trojan e.g. combinational, sequential or analog types. 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 remains undetected. For a logic locked circuit, the trigger inputs need to be selected from nodes that are not affected by key gates. Otherwise, an adversary cannot activate a Trojan as it does not know this secret key, and thus the internal signal value for an activation pattern. The trigger can be implemented as an AND gate (e.g. 3-input AND gate for *Type-3* Trojan). When a Trojan is activated, the output of this AND gate becomes 1 and it delivers the payload (selection input of the multiplexer or OR gate shown in Figure 1) to the circuit to leak the secret key. The trigger can also be any logic function that provides logic 1 when activated.

1) Design of a Combinational Hardware Trojan: A combinational hardware Trojan generally comprises of a trigger and a payload, the detailed modeling can be found in [25]. Functionally, a combinational Trojan manifests its effects instantly upon the availability of the trigger inputs and effects the original netlist at the payload. 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.

For an adversary, the number of Trojan choices to break the security of logic locking is very large. In order to determine all possible Trojan choices for any given netlist, *p* nodes as trigger inputs from *N* nodes of the circuit needs to be selected. The value of *N* can be determined as :

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

where, *PI* represents the number of primary inputs, *G* denotes the number of gates and *F* is the number of fanout branches in the netlist. Note that the number of lines impacted by the key gates (*M*) needs to excluded as their value is unknown to the adversary and hence, cannot be selected as trigger inputs.

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

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

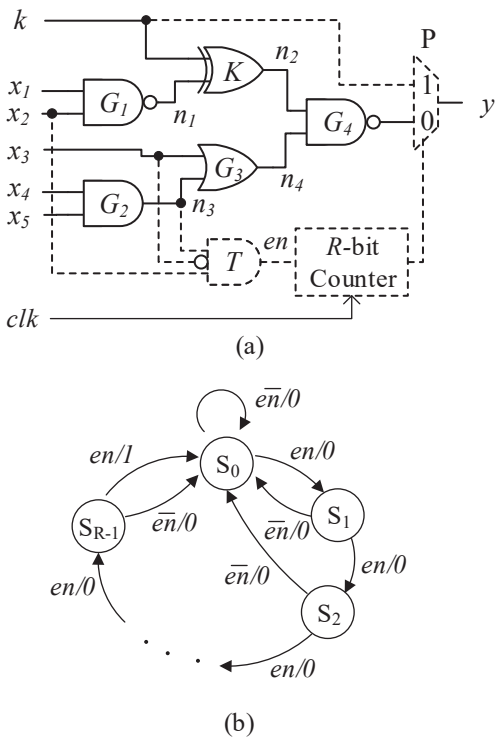

Figure 2. (a) The netlist of a sequential Trojan with a *R-bit* counter, (b) The finite state machine (FSM) of the counter used in a sequential Trojan.

The Equation 2 represents all possible combinations to select *p* lines from *N* with each applied directly or inverted to the trigger. The important aspect of the Trojan design is its stealthiness. Generally, the manufacturing tests are performed at the foundry. The SoC designer can generate test patterns considering the key as input (the pattern generation is described in detail in [6], [20]). To detect all the stuck-at faults (SAFs), different test patterns (e.g., $P = \{P_1, P_2, \ldots, P_n\}$) are required and they are generated using Synopsys TetraMax [26] ATPG tool. To avoid a Trojan being activated by these manufacturing test patterns, the trigger of the Trojan must remain quiet for all these input patterns. Once the test patterns are provided to the foundry, a hardware Trojan activation pattern is selected that does not belong to this set of test patterns. 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. The detailed steps and algorithm to design a combinational Trojan evading manufacturing test patterns can be found in [12].

2) Sequential Trojan Design: The model for designing a sequential hardware Trojan can be derived from the combinational Trojan model. A sequential Trojan modifies the functionality of a circuit until the specified trigger appears R times consecutively, where the value of R can be controlled

by the adversary. 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 once triggered R times as shown in Figure 2.(a). The dashed lines represent the added Trojan in order to tamper the netlist to launch the *tampering attack*. Figure 2.(b) shows the finite state machine (FSM) design of a ***R-bit*** counter used to implement the state element of a sequential Trojan. This counter is enabled (en) once the trigger condition is fulfilled, i.e., the output of the *p*-input AND gate becomes 1. The FSM goes to 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}. The Trojan delivers at the payload (MUX) only after reaching the maximum count value (R). 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 *Tampering attacks*. 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.

C. Analysis

The number of combinational hardware Trojans applicable for launching any type of tampering attack is obtained using the Equations 1 and 2. Out of all possible Trojans, the valid Trojans that will not be detected during manufacturing tests are obtained after performing the analysis using all the stuck-at fault patterns obtained from the TetraMax ATPG tool with 100% fault coverage. For a small benchmark circuit *c432*, the total number of *Type-2* Trojans is 1.08×10^5, whereas, the number of valid Trojans is 1.0×10^5. It is essential to be acquainted with a large number of valid Trojan choices for an adversary. These numbers for valid Trojans increase exponentially for the increase in *Type-p* of the Trojan and the circuit size. Additionally, detecting all the valid Trojans is a hard problem. Moreover, it is not necessary to impose the fault pattern detection condition for designing sequential Trojans as it is highly unlikely that a particular trigger condition will arrive R times consecutively during the normal operation or testing of a chip.

The area and power overhead 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 sequential Trojan, it is necessary to add a R-bit counter along with a p-input AND gate to implement the trigger. Note that both the area and power overhead will decrease with the increase in the size of the design. For a small circuit from ISCAS'85 benchmark suite [27], c6288, the area and power overhead for the netlist tampered with *Type-4* combinational Trojan is around 0.01% and 0.33%, respectively. It is safe to mention that these overheads can practically be negligible for a modern industrial design with millions of gates.

D. Existing Countermeasures and their Limitations

The security of a logic locking technique can be tied together with the hardware Trojan detection problem. 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 [28]–[30], and side-channel analysis [31]–[33]. Detection through logic testing is 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 the same trigger pattern at the input R times. Side-channel analysis is largely affected by the process and environmental variations which can mask the effect of Trojan. Moreover, it is difficult to acquire a golden chip for such an analysis. On the other hand, prevention methods can be categorized as design-for-trust measures [34]–[37] and split manufacturing [38]–[40]. However, different attacks on these approaches have also been proposed [41], [42]. The best solution so far requires reverse-engineering the entire SoC through SEM images to detect any additional nets or components, however, it is yet to be validated its effectiveness of detection when a chip is fabricated using recent technology nodes (10nm and beyond) [23]. 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.

III. Optical Attacks on logic locking

A. Optical Attack Techniques

Failure analysis (FA) based on optical debugging techniques, e.g., photon emission analysis, laser-voltage analysis, optical beam induced resistance change (OBRICH), etc. were developed to facilitate the yield analysis and fault localization. Since silicon is transparent to the photons at near-infrared (NIR) spectra, active and passive photon stimulation facilitated in run-time monitoring of chip for fault localization. However, the same optical inspection methods can also be used for extracting the assets stored in modern ICs [13], [15], [43], [44].

In the past decade, three major classes of optical attack methods have been used for extracting the crypto-keys stored in the SoCs. They are – a) photon emission (PE), b) electro-optical (EO)/laser-voltage (LV) techniques, and c) laser stimulation (LS).

1) Photon Emission Analysis (PEA): PEA is primarily developed for functional analysis and fault localization on the silicon die without any external stimuli. During the switching of logic gates, the charge carriers gain kinetic energy as the MOSFET transistors pass through saturation for a brief period. Then, the energy of the

carriers is released in the form of emitted photons at the drain edge of the transistor, i.e., the pinch-off region of the transistor's space-charge region. Photons are captured with a Si-CCD or InGaAs detector, and the signal fed to a computer to create a 2D image to map the location of switching activity of the logic gates. In addition, temporal information of the signal propagating through the chip can be detected if PEA is incorporated with picosecond image circuit analysis (PICA) [45], [46]. An adversary can threaten the confidentiality of the on-chip asset by using the aforementioned data-dependency of hot-carrier luminescence as a source of side-channel information [43], [46].

2) Electro-optical/ Laser-voltage Techniques: Electro-optical techniques are an active approach for optically probing the transistor state through two well-known approaches – electro-optical probing (EOP) and electro-optical frequency mapping (EOFM) [15], [47]. A laser stimulus is focused on a transistor. Since the absorption coefficient and refractive index of silicon are dependent on the space-charge densities caused by the time-varying electric field, the amplitude and phase of the incident laser beam are modulated by the electrical parameters of the transistor. EOP analysis can probe the transistor's electrical parameter based on the modulated reflected laser beam.

   Unlike EOP, in EOFM, the laser scans the region of interest (RoI) on the device under test (DUT) and the reflected light is evaluated by a spectrum analyzer, which acts as a narrowband frequency filter [48]. The frequency-filtered values are then sampled for every scanned pixel and used to construct a 2D image using a grayscale or false-color representation [15]. Nodes operating at the frequency of interest will modulate the laser with the same frequency and appear as a bright spot in the 2D image. Other approaches for optical probing are laser voltage probing (LVP) and laser voltage imaging (LVI). The LVP/LVI methods are equivalent to EOP/EOFM, respectively, except the light source used for the later ones is incoherent.

3) Laser Stimulation (LS): IR laser is applied from chip backside in the LS method to induce perturbation in the circuitry. Depending on the wavelength of the injected photons and silicon bandgap energy, laser stimulation introduces photoelectric and/or thermal effects in the device. This perturbation in the device can expose the device parameters like the logical state of gates, registers, or memory.

   • *Laser Fault Injection (LFI)*: Lasers with photon energy greater than or equal to the silicon band-gap energy, 1.1 eV, generate electron-hole pairs in the silicon. This effect is commonly referred to as photoelectric laser stimulation (PLS) [44], [45]. Lasers with a wavelength less than 1100nm can introduce PLS in the device. The logic state of a CMOS circuit can be flipped if the PLS is focused on a transistor drain or source terminal [44], [49], thus injecting a fault into the circuitry. The success of laser fault injection depends on several factors such as the wavelength, power, and exposure time of the incident photons [50].

   • *Thermal Laser Stimulation (TLS)*: Due to the lower photon energy of the 1.1 μm laser, the radiation will majorly induce thermal stimuli in the device. The local heating caused by TLS induce resistance variation and generates an electromotive force due to the Seebeck effect. The effects mentioned above induce variation in device parameters like voltage, and current. During TLS, a device is biased at supply voltage, and the current is monitored between the supply pins via the current pre-amplifier (see Fig. 3.(c)). A PC simultaneously samples the current pre-amplifier output, and a 2D map of devices response is generated to localize the current variations in the circuit [45], [51].

4) Necessary Equipment for Optical Attacks: A successful optical attack requires a laser source with variable wavelengths for laser stimulation and an InGaAs detector for photon emission analysis. Recent advancements in FA instruments have increased the availability of various microscopes to aid optical analysis techniques. Laser scanning microscope (LSM) and photon emission microscope (PEM) are used for laser stimulation analysis and PEA. In addition, several FA instruments have incorporated the LSM and PEM to provide a single solution for all optical debugging techniques. These microscopes are available in different industrial/academic labs and widely used for FA analysis.

B. Security Threats of Optical Probing

The confidentiality and integrity of the Sensitive information protected by the SoC security architecture are considered violated if the assets are proven to be vulnerable against optical attacks. Therefore, ignoring the threat from optical attacks, leave a wide attack surface for an adversary to reveal the assets stored in the SoC. In SoC, security mesh is placed in the device. Moreover, a large number of interconnecting metal layers at frontside makes tracking the transistor activity impractical. The backside, which lacks such metal layers, is extremely vulnerable to optical attacks on the die or exposed silicon chip.

In recent years different security-sensitive components in SoC showed their susceptibility against optical attacks. Logic locking, a promising protection barrier against IP piracy, can not be considered secure against optical attacks. Secrecy of the locking key protects the IP design from an untrusted entity in the supply chain. In logic locking, it is assumed that the key is configured in a *read and tamper-proof* memory. However, optical attacks, e.g., PEA, LFI, and TLS, have exposed cryptographic cipher keys such as AES, RSA, etc.. stored in Flash and EEPROM by attacking the control circuitry or memory cells [44], [51]. Flash and EEPROM are widely used as on-chip memory to store other types of sensitive information as well such as soft IP, algorithms, and authentication keys.

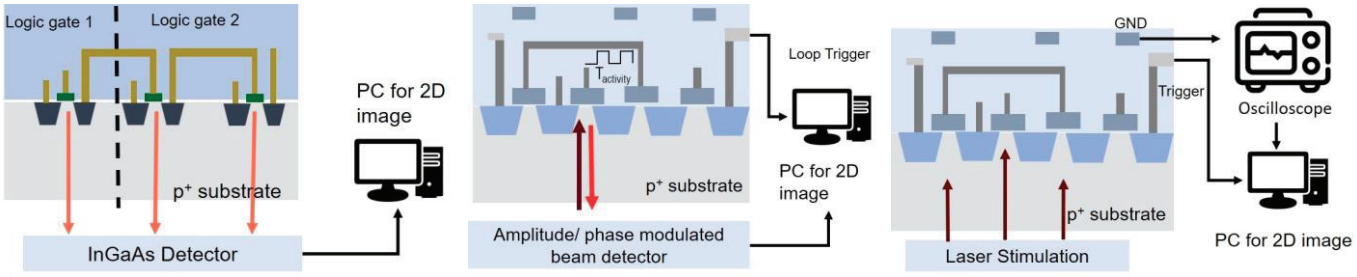

Figure 3. (a) Detector captures the photons emitted during switching of N-type MOSFET and generates a 2D mapping of activity, (b) optically probing a transistor operating at Tactive frequency using EFOM/EOP, (c) Device parameters are observed during laser stimulation technique. Depending on the application laser directed for a specific transistor (for LFI) or a region of interest.

Therefore, the common assumption about the existence of *tamper and read-proof* NVM is no longer valid.

Secured key storage, e.g., OTP memory (for example, efuse, antifuse) and battery-backed RAM (BBRAM), can also be read using active and passive photon stimulation. For example, an adversary can de-process the IC with FA tools and localize the efuse on the die, followed by OBIRCH analysis for reading the efuse value [52]. Physical unclonable function (PUF) generates keys from the intrinsic properties of the device [49]. PUF has demonstrated vulnerabilities against several non- and semi-invasive attacks, like photonic emission analysis and laser fault injection. Besides, TLS can read the output of SRAM PUF once the clock of the circuit is turned off [53]. Besides, PUF is not an option for the logic locking key.

Recent studies have shown that electro-optical analysis can be used to localize and probe the logic gates and flip-flops implemented in a design [13], [15]. Therefore, an adversary can always use advanced reverse engineering tools to extract the gate-level netlist of the locked IP and localize the key-gates, flip-flops connected to the key-gates (generally termed as key registers [13], [14]) and interconnects carrying the locking key of the IP. Thereafter, a malicious entity can use EOFM/EOP to extract the key bits. Hence, simply protecting the locked IP against Oracle-guided (e.g., SAT attack) and Oracleless (e.g., SAIL attack) does not prevent IP piracy.

C. Existing Countermeasure and their Limitation

Security against optical analysis can be addressed in two ways – detection and prevention of laser stimulation. The possible countermeasure for the backside of a chip can be, further, divided into three levels – material, device and circuit level.

One material level solution for preventing optical attack can be adding an active opaque layer at the backside of the chip. However, an adversary like an untrusted foundry can identify such a countermeasure and remove the opaque layer by polishing. Active monitoring of bulk silicon thickness, device capacitance, or adding fragile structure to prevent polishing the backside of the chip can be promising for opaque layer protection.

Integrating nanopyramid [22] structure between the device layer can scatter the incident/reflected laser beam in the device. Besides, implementing circuitry operating at an inverted signal can also be used to prevent laser modulation.

Traditionally, light sensors are used for identifying the laser stimulation on the chip as a circuit based laser stimulation detection mechanism. The light sensors can only detect laser with higher energy than silicon bandgap energy, hence, only applicable against LFI detection. However, TLS introduces temperature and current variations in the circuit, which can influence the RC delay in the circuits and can be used in sensors for detecting TLS [14]. Analog shields and sensors can be alternative approaches for detecting TLS. Analog shields and sensors utilize analog features, e.g. capacitance or RC delay, at specific chip locations to detect the attack, which can be used for detecting both LFI and TLS. However, the applicability of these analog shields remained untested.

IV. CLIC-A Methods

CLIC-A uses methods oriented around commercial ATPG to uncover key-input values in locked combinational and sequential circuits. There are four methods currently included in CLIC-A. The first three are applicable to combinational locking and are described in greater detail in [17]. The fourth is applicable to sequential locking and is described in [18]. The first method is key-input sensitization, the second is constraint-based ATPG, the third is targeting key-dependent faults, and the forth is sequential key extraction. The first two methods require the use of an oracle and a netlist of the locked circuit. The third and fourth methods are oracle-free, meaning they only require the netlist. More details and pseudo-code for all three combinational methods can be found in [17] and more information on the sequential method can be found in [18].

1) Key-Input Sensitization: CLIC-A includes the key-input sensitization method described in [54]. In this approach, ATPG is applied to the locked netlist to generate tests that sensitize individual key inputs to one or more primary outputs. By sensitization, we refer to the automatic test pattern generation of a test input pattern that propagates the targeted key-input

value to one or more primary outputs. In this method, a stuck-at fault is considered at each key input, one at a time, while constraining all the other key inputs to a do-not-care (X) value. Constraining the other key inputs to X, mutes their effects on circuit functionality. If a test is found under these circumstances, it is then applied to the oracle circuit. Based on which fault is used, and whether the response between the oracle and locked netlist match or mismatch, the key-input value is easily deduced. For example, when a test is generated for a single stuck-at zero fault, if the outputs of the oracle and the netlist match for that test, the key-input value is one. If there is a mismatch, the key-input value is zero.

The purpose of sensitization is to find individual key-input values that have at least one direct path to an output without the interference of other key inputs. Having key inputs that can be sensitized is most common for locked circuits using the random [55] or fault-based [56] locking techniques.

2) Constraint-based ATPG: CLIC-A also includes a novel ATPG constraint-based characterization method. This method utilizes the constraint functions built into commercial ATPG. An ATPG constraint is a logic function applied to an ATPG run where the signal-line values included in the constraint must satisfy that function for any test generated. If a test cannot be generated that satisfies the constraints, ATPG reports a failure. For example, if the constraint is $(s_1 + \overline{s_2} + \overline{s_3})$ then every test generated adhering to that constraint must have $s_1 = 1$, $s_2 = 0$, or $s_3 = 0$.

Constraint-based ATPG begins by performing ATPG on each key input, one at a time. ATPG targets a fault at each key input to generate a test which is then applied to the oracle. Based on whether the outputs match or mismatch, a constraint is constructed and applied to subsequent ATPG runs. If the outputs between the simulated netlist and oracle do not match, this means that the generated key-input values are incorrect. This results in a constraint function where all the key-input values are inverted and disjunctively combined. However, if instead the outputs between the simulated netlist and oracle match, it does not ensure that the key-input values generated are correct. Therefore, a constraint is added where all key-input values, except the targeted key input, are inverted and disjunctively combined. This new constraint ensures that ATPG can regenerate the same set of key-input values again if it is correct, while able to generate other key-input values as well. Additionally, a logic minimization tool is employed to keep the number of constraints at a minimum. The constraints are used to guide ATPG to the correct key value and solve key inputs that are clustered together. This method is most effective against strong locking [54].

3) Targeting Key-Dependent Faults: The third method included in CLIC-A targets faults that require a majority of key inputs for sensitization which is common in the SAT resistant methods. The SAT-resistant locking methods typically have a similar structure in that there exists at least one signal line in which all or a majority of the key inputs converge upon. Such a signal line has a corresponding fault that requires all or a majority of the key inputs for detection. Analysis of the generated test vectors locates these signal lines and extracts the correct key. This is possible because the correct key value is typically hard-coded into the circuit.

For example, in Figure 4, targeting a fault that sets the signal line *restore* to a zero would reveal the hard-coded key value in the test vectors.

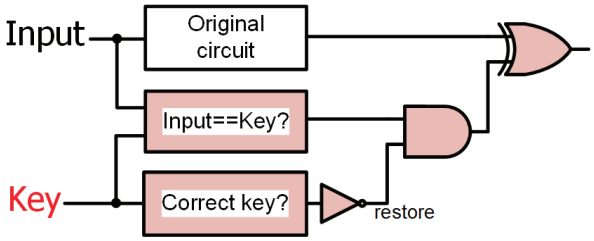

Figure 4. SARLock logic locking.

To first find the key-dependent faults, ATPG is run on the netlist with all unsolved key inputs constrained to do-not-care values (X). Faults that require the key inputs for detection will be reported as an ATPG failure. Each fault in this set of key-dependent faults is targeted using a second round of ATPG. If a test is found, the generated test is stored, along with the fault to be further analyzed for the key value. The test analysis method used for the generated tests differs depending on the lock type. More information on the key extraction is provided in [17].

4) Sequential Key Derivation: The last method included in CLIC-A solves a key sequence from a sequentially locked circuit. This method is specifically effective at solving a key from a circuit locked with an entrance FSM, such as the method proposed in [57]. With an entrance FSM lock, there is functionality that is only accessible when the key sequence is entered. Associated with this functionality are faults that can only be detected with a test vector containing the key sequence. As a result, to derive the key sequence from these circuits, we first apply sequential ATPG to the netlist, then search the generated tests for a repeatedly appearing pattern.

To perform sequential ATPG, the iterative logic array, ILA, model is used. An ILA model mimics N clock cycles of the behavior of a sequential circuit by making N copies of the combinational logic, and replacing memory elements with equivalent combinational circuits (*e.g.*, D-type flip flops are replaced with a single buffer and inverter). Moreover, the inputs to the flip-flops are added as secondary outputs to the circuit, and the outputs of the flip-flops are added as secondary inputs to the circuit. The secondary outputs of frame i (which corresponds to clock cycle i) are connected to the secondary inputs of frame $i + 1$. The secondary inputs for the first frame are held at the reset state, which must exist otherwise upon power-up, the circuit could easily be operating within the protected FSM. Figure 5 illustrates an ILA model consisting of N frames.

Combinational ATPG is then run on each time frame of the ILA model and the test vectors are stored. The test

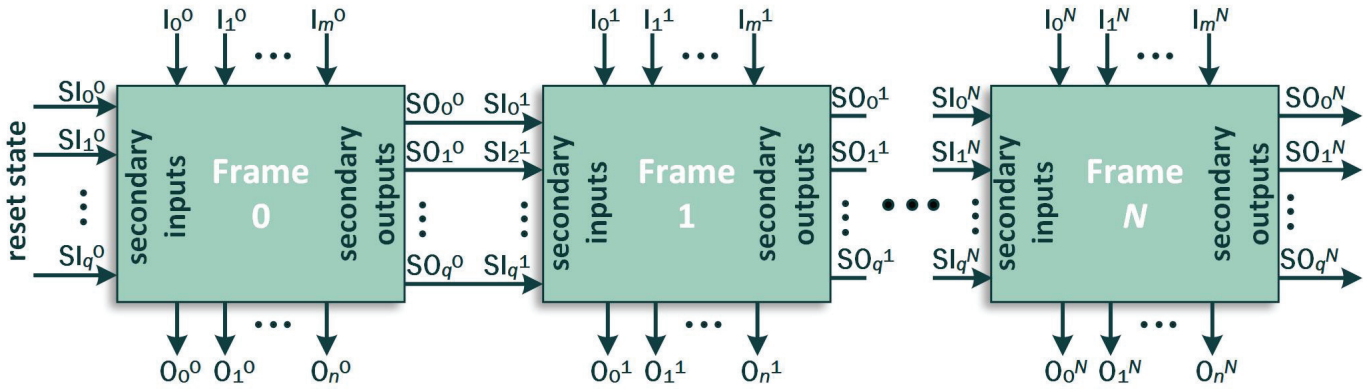

Figure 5. An ILA model of a sequential circuit consisting of N frames where the first set of secondary inputs are tied to the reset state.

vectors generated from each time frame are then searched for a repeatedly appearing sequence required to detect faults. When the same sequence appears for multiple time frames, that sequence is concluded to be the key sequence. More information on this method can be found in [18].

A. Experiments

This section presents the results of experiments that demonstrate the effectiveness of CLIC-A compared to the effectiveness of the SAT attack. In this section, key-input sensitization is referred to as method one, constraint-based ATPG is referred to as method two, targeting key-dependent faults is referred to as method three, and sequential key derivation is referred to as method four. With the exception of two benchmarks, the experiments in this section are performed on locked benchmarks created by others.

The experiments are performed using a machine with 64 CPU cores running at 2.20 GHz with 1,009GB of RAM. CLIC-A uses an off-the-shelf commercial ATPG tool and the results for the SAT attack stem from the publicly-available binaries reported in [11].

The results in Table I demonstrate the results of CLIC-A and SAT on the location-based locking methods. In particular, the experiments on randomly locked circuits show the results on a multiplier circuit, c6288, and a portion of the OpenSparc processor [63]. Both experiments demonstrate that for large/complex circuits, CLIC-A outperforms SAT. In the case of the processor, the SAT attack exits without solving a key. The rest of the rows in the table demonstrate that for smaller benchmark circuits, the performance of SAT and CLIC-A is comparable.

Table II demonstrate the results of running CLIC-A and SAT on the SAT-resistant locking methods. The experiments performed on circuits locked with cyclic, Anti-SAT, SARLock, and TTLock demonstrate that CLIC-A solves key values from circuits that SAT cannot. Additionally, the experiments performed on SFLL-HD circuits demonstrate that for smaller key sizes, the SAT attack and CLIC-A both solve a correct key. However, for larger key sizes, the results demonstrate that CLIC-A is more effective.

The results in Table III demonstrate that the application of CLIC-A method four to entrance FSM circuits results in the correct key sequence being derived. The use of the ILA method causes circuit size to increase each iteration, which in turn causes a runtime increase. The increase in runtime is an issue for larger circuits. However, experiments demonstrate that the increased runtime does not prevent CLIC-A from generating test sequences that reach the protected FSM.

B. Strengthening Combinational Locking

The introduction of the hard-coded "correct key" or hard-coded "protected pattern", as used in SARLock, TTLock, and SFLL-HD provides a particular point of attack that is exploited in CLIC-A. Because the key value is implemented as part of the design, when targeting faults that require those particular signal lines, the correct key/protected pattern has a high likelihood of appearing in test patterns. These methods could be considered more vulnerable than traditional locking methods because the key can be extracted without an oracle.

Additionally, strengthening the location-based locking methods (i.e., random, fault-based, and strong), requires mitigating the effectiveness of method two in CLIC-A. The time required to solve key-input values using constraint-based ATPG is dependent on the number of key inputs in a logic cone. As the number of key inputs in a cone grows, so does the time it takes to identify key-input values. Therefore increasing the number of key inputs in a cone effectively mitigates this attack. However, increasing the number of key inputs in each cone does come at the expense of increasing design overhead.

V. Conclusion

In this paper, we presented the security vulnerability of the various techniques of logic locking that can be exploited to extract the secret key. Specifically, the exposure of logic locking to attacks independent of its SAT resiliency. This includes the non-traditional physical attacks using optical probing (EOFM/EOP) to extract the key bits even after the chip is being manufactured incorporating the known state-of-art defenses. Additionally, tampering the locked circuit with hardware Trojans that remains undetected during manufacturing tests pose a strong security threat to any key-based locking technique. Only the attacker has the knowledge about the specific input pattern that can lead to key extraction, increasing the difficulty to identify the *Tampering Attack*. We

Table I
CLIC-A AND SAT COMPARISONS FOR LOCATION-BASED LOCKING METHODS.

Circuit (gates, keys)	Source	Lock Type	SAT (keys solved, runtime)	CLIC-A (keys solved, runtime)	CLIC-A Methods Applied
c6288 (2406, 128)	Trust-Hub [58]	Random	128, 2.3 days	128, 18 hours	1, 2
OpenSparc T2-NCU (84189, 5507)	Self-locked	Random	0, exits	4160, 27 hours	1, 2
c1355 (546, 29)	SAT attack [59]	Fault-based	29, 0.06 seconds	29, 6.7 seconds	1, 2
c1908 (971, 91)	SAT attack [59]	Fault-based	91, 0.08 seconds	91, 6.6 minutes	1, 2
c432 (168, 8)	SAT attack [59]	Strong	8, 0.06 seconds	8, 3.2 minutes	1, 2
c499 (212, 10)	SAT attack [59]	Strong	10, 0.1 seconds	10, 13.0 minutes	1, 2

Table II
CLIC-A AND SAT COMPARISONS FOR SAT RESISTANT LOCKING METHODS.

Circuit (gates, keys)	Source	Lock Type	SAT (keys solved, runtime)	CLIC-A (keys solved, runtime)	CLIC-A Methods Applied
c7552 (1428, 26)	Trust-Hub [58]	Cyclic	0, inf. loop	26, 8.85 minutes	1, 2
c7552 (1494, 50)	Trust-Hub [58]	Cyclic	0, inf. loop	50, 8.42 minutes	1, 2
c6288(2577, 128)	Trust-Hub [58]	Anti-SAT + Random	0, 3.0 days	128, 12.7 minutes	1, 2, 3
c7552 (4337, 685)	Trust-Hub [58]	Anti-SAT + Random	0, 3.0 days	685, 13.5 minutes	1, 2, 3
i7 (1736, 275)	Double Dip Attack [60]	SARLock + Random	0, 3.0 days	275, 8.0 minutes	1, 2, 3
k2 (2048, 139)	Double Dip Attack [60]	SARLock + Random	0, 3.0 days	139, 2.81 minutes	1, 2, 3
c7552 (1451, 32)	TTLock authors [61]	TTLock	0, 5.0 days	32, 1.4 minutes	3
b18 (57362, 64)	TTLock authors [61]	TTLock	64, 1.1 hours	64, 26.5 minutes	3
c1355 (621, 32)	Self-locked	SFLL-HD	32, 7.16 seconds	32, 4.1 minutes	3
dfx (42404, 256)	SFLL-HD authors [62]	SFLL-HD	0, 6.0 days	256, 3.0 days	3

Table III
CLIC-A RESULTS ON SEQUENTIAL LOCKING METHODS.

Circuit (gates, flip-flops)	Key Length (cycles)	Source	Lock Type	CLIC-A (keys solved, runtime)	CLIC-A Methods Applied
s9234 (733, 166)	31	HARPOON authors [57]	HARPOON	31 cycles, 48.4 minutes	4
s13207 (964, 358)	31	HARPOON authors [57]	HARPOON	31 cycles, 3.8 hours	4
s38584 (5987, 1273)	31	HARPOON authors [57]	HARPOON	31 cycles, 41.0 hours	4

also discuss *CLIC-A* attack methods oriented around ATPG to demonstrate the vulnerability of traditional locking methods and SAT-resistant methods that contain a "hard-coded" key. The result shows *CLIC-A* methods outperform the SAT attack. As a whole, we can conclude that the security of logic locking cannot be justified until resiliency against all forms of attacks is provided.

REFERENCES

[1] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris, "IPP@ HDL: efficient intellectual property protection scheme for IP cores," *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, pp. 578–591, 2007.

[2] M. Tehranipoor and C. Wang, *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

[3] M. M. Tehranipoor, U. Guin, and D. Forte, "Counterfeit integrated circuits," in *Counterfeit Integrated Circuits*. Springer, 2015, pp. 15–36.

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

[5] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "FORTIS: a comprehensive solution for establishing forward trust for protecting IPs and ICs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, p. 63, 2016.

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

[7] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. of the conf. on Design, automation and test in Europe*, 2008, pp. 1069–1074.

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

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

[10] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. of IEEE/ACM int. conf. on Computer-aided design*, 2007, pp. 674–677.

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

[12] A. Jain, Z. Zhou, and U. Guin, "TAAL: Tampering Attack on Any Key-based Logic Locked Circuits," *arXiv preprint arXiv:1909.07426*, 2019.

[13] M. Rahman, S. Tajik, M. Rahman, M. Tehranipoor, and N. Asadizanjani, "The key is left under the mat: On the inappropriate security assumption of logic locking schemes," Cryptology ePrint Archive, Report 2019/719, 2019, https://eprint. iacr. org ..., Tech. Rep., 2019.

[14] M. T. Rahman, M. S. Rahman, H. Wang, S. Tajik, W. Khalil, F. Farahmandi, D. Forte, N. Asadizanjani, and M. Tehranipoor, "Defense-in-depth: A recipe for logic locking to prevail," *Integration*, 2020.

[15] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit, "On the power of optical contactless probing: Attacking bitstream encryption of fpgas," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1661–1674.

[16] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "TGA: An Oracle-less and Topology-Guided Attack on Logic Locking," in *Proc. of the Workshop on Attacks and Solutions in Hardware Security Workshop (ASHES)*, 2019, pp. 75–83.

[17] D. Duvalsaint, X. Jin, B. Niewenhuis, and R. D. Blanton, "Characterization of Locked Combinational Circuits via ATPG," *International Test Conference*, 2019.

[18] D. Duvalsaint, Z. Liu, A. Ravikumar, and R. D. Blanton, "Characterization of Locked Sequential Circuits via ATPG," *International Test Conference Asia*, 2019.

[19] Rahman, M Sazadur and Nahiyan, Adib and Amir, Sarah and Rahman, Fahim and Farahmandi, Farimah and Forte, Domenic and Tehranipoor, Mark, "Dynamically Obfuscated Scan Chain To Resist Oracle-Guided Attacks On Logic Locked Design," 2019.

[20] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *Trans. on Very Large Scale Integration (VLSI) Systems*, pp. 818–830, 2018.

[21] H. Wang, Q. Shi, D. Forte, and M. M. Tehranipoor, "Probing Assessment Framework and Evaluation of Antiprobing Solutions," *Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1239–1252, 2019.

[22] H. Shen, N. Asadizanjani, M. Tehranipoor, and D. Forte, "Nanopyramid: An Optical Scrambler Against Backside Probing Attacks," in *Proc. Int. Symposium for Testing and Failure Analysis(ISTFA)*, 2018, p. 280.

[23] N. Vashistha, H. Lu, Q. Shi, M. T. Rahman, H. Shen, D. L. Woodard, N. Asadizanjani, and M. Tehranipoor, "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*, 2018, p. 256.

[24] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. of Annual Design Automation Conference*, 2012, pp. 83–89.

[25] Z. Zhou, U. Guin, and V. D. Agrawal, "Modeling and test generation for combinational hardware Trojans," in *VLSI Test Symposium (VTS)*, 2018, pp. 1–6.

[26] Synopsys Inc., Mountain View, CA, USA, "TetraMAX ATPG: Automatic Test Pattern Generation," 2017.

[27] D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," *North Carolina State University*, vol. 25, 1985.

[28] N. Lesperance, S. Kulkarni, and K.-T. Cheng, "Hardware Trojan Detection Using Exhaustive Testing of k-bit Subspaces," in *Proc. of Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2015, pp. 755–760.

[29] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis," in *Proc. ACM SIGSAC Conf. on Computer & Communications Security*, 2013, pp. 697–708.

[30] S. K. Haider, C. Jin, M. Ahmad, D. Shila, O. Khan, and M. van Dijk, "Advancing the State-of-the-Art in Hardware Trojans Detection," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[31] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection Using IC Fingerprinting," in *Proc. IEEE Symp. Security and Privacy (SP)*, 2007, pp. 296–310.

[32] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity Analysis to Hardware Trojans Using Power Supply Transient Signals," in *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 3–7.

[33] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan Detection Through Golden Chip-Free Statistical Side-Channel Fingerprinting," in *Proc. of Design Automation Conference*, 2014.

[34] R. S. Chakraborty and S. Bhunia, "Security Against Hardware Trojan Through a Novel Application of Design Obfuscation," in *Proc. Int. Conf. Computer-Aided Design*, 2009, pp. 113–116.

[35] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," *IEEE Trans. of VLSI Systems*, pp. 112–125, 2012.

[36] K. Xiao and M. Tehranipoor, "BISA: Built-In Self-Authentication for Preventing Hardware Trojan Insertion," in *Proc. IEEE Int. Symp. Hardware-Oriented Security and Trust*, 2013, pp. 45–50.

[37] X. T. Ngo, S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "Linear Complementary Dual Code Improvement to Strengthen Encoded Circuit Against Hardware Trojan Horses," in *Proc. IEEE Int. Symp. Hardware Oriented Security and Trust*, 2015, pp. 82–87.

[38] K. Vaidyanathan, B. P. Das, and L. Pileggi, "Detecting Reliability Attacks During Split Fabrication Using Test-Only BEOL Stack," in *Proc. of Design Automation Conf.*, 2014, pp. 1–6.

[39] J. J. V. Rajendran, O. Sinanoglu, and R. Karri, "Is Split Manufacturing Secure?" in *Proc. Conf. Design, Automation and Test in Europe (DATE)*, 2013, pp. 1259–1264.

[41] J. Magaña, D. Shi, J. Melchert, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" *Trans. on Very Large Scale Integration Systems*, pp. 3406–3419, 2017.

[40] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The Cat and Mouse in Split Manufacturing," in *Proc. of DAC.*, 2016, pp. 1–6.

[42] W. Xu, L. Feng, J. J. Rajendran, and J. Hu, "Layout recognition attacks on split manufacturing," in *Proc. of Asia and South Pacific Design Automation Conference*, 2019, pp. 45–50.

[43] S. Tajik, D. Nedospasov, C. Helfmeier, J.-P. Seifert, and C. Boit, "Emission analysis of hardware implementations," in *Euromicro Conference on Digital System Design*, 2014, pp. 528–534.

[44] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 2–12.

[45] C. Boit, C. Helfmeier, and U. Kerst, "Security risks posed by modern IC debug and diagnosis tools," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2013, pp. 3–11.

[46] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," *Journal of Cryptographic Engineering*, pp. 3–15, 2013.

[47] Z. Song and L. Safran, "LVI and LVP Applications in In-Line Scan Chain Failure Analysis," *Elec. Device Failure Analysis*, pp. 4–14, 2016.

[48] H. Photonics, "Emission Microscopy: Phemos-1000," https://www.hamamatsu.com/resources/pdf/sys/SSMS0003E PHEMOS1000.pdf, accessed:2018-04-26.

[49] S. Tajik, H. Lohrke, F. Ganji, J.-P. Seifert, and C. Boit, "Laser fault attack on physically unclonable functions," in *2015 workshop on fault diagnosis and tolerance in cryptography (FDTC)*, 2015, pp. 85–96.

[50] F. Courbon, P. Loubet-Moundi, J. J. Fournier, and A. Tria, "Increasing the efficiency of laser fault injections using fast gate level reverse engineering," in *Int. Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 60–63.

[51] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert, "Key Extraction Using Thermal Laser Stimulation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 573–595, 2018.

[52] H. C. Chen, C. Y. Tsai, S. Y. Liu, Y. P. Chang, and J. C. Lin, "Defect Localization and Root Cause Analysis on e-Fuse Read Reliability Failure," in *Proc. Int. Sym. for Testing and Failure Analysis (ISTFA)*, 2014, p. 304.

[53] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical characterization of arbiter PUFs," in *Int. Workshop on Cryptographic Hardware and Embedded Systems*, 2014, pp. 493–509.

[54] O. S. J. Rajendran, Y. Pino and R. Karri, "Security Analysis of Logic Obfuscation," in *Design Automation Conference*, 2012.

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

[56] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *Transactions on computers*, pp. 410–424, 2013.

[57] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1493–1502, 2009.

[58] S. Amir, B. Shakya, X. Xu, Y. Jin, S. Bhunia, M. Tehranipoor, and D. Forte, "Development and evaluation of hardware obfuscation benchmarks," *Journal of Hardware and Systems Security*, pp. 142–161, 2018.

[59] S. R. P. Subramanyan and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2015.

[60] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Proc. of the on Great Lakes Symposium on VLSI*, 2017, pp. 179–184.

[61] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, "What to lock? Functional and parametric locking," in *Proc. of the on Great Lakes Symposium on VLSI*, 2017, pp. 351–356.

[62] M. Yasin et al., "SFLL Attack Framework," https://github.com/DfX-NYUAD/CCS17, 2017.

[63] "OpenSPARC T2," http://www.oracle.com/technetwork/systems/opensparc/, Oracle.