Received 14 March 2017; revised 31 May 2017; accepted 26 July 2017.
Date of publication 21 August 2017; date of current version 9 June 2020.

Digital Object Identifier 10.1109/TETC.2017.2740364

Removal Attacks on Logic Locking and Camouflaging Techniques

MUHAMMAD YASIN, (Student Member, IEEE), BODHISATWA MAZUMDAR, (Member, IEEE), OZGUR SINANOGLU, (Senior Member, IEEE), AND JEYAVIJAYAN RAJENDRAN, (Member, IEEE)

M. Yasin is with the Electrical and Computer Engineering, New York University, Tandon School of Engineering, Brooklyn, NY 11201
B. Mazumdar is with the Computer Science and Engineering, Indian Institute of Technology Indore, Indore, MP 453552, India
O. Sinanoglu is with the New York University in Abu Dhabi, Abu Dhabi 129188, United Arab Emirates
J. Rajendran is with the Department of Electrical Engineering, University of Texas at Dallas, Richardson , TX 75080
CORRESPONDING AUTHOR: M. YASIN (yasin@nyu.edu)

ABSTRACT With the adoption of a globalized and distributed IC design flow, IP piracy, reverse engineering, and counterfeiting threats are becoming more prevalent. Logic obfuscation techniques including logic locking and IC camouflaging have been developed to address these emergent challenges. A major challenge for logic locking and camouflaging techniques is to resist Boolean satisfiability (SAT) based attacks that can circumvent state-of-the-art solutions within minutes. Over the past year, multiple SAT attack resilient solutions such as Anti-SAT and AND-tree insertion (ATI) have been presented. In this paper, we perform a security analysis of these countermeasures and show that they leave structural traces behind in their attempts to thwart the SAT attack. We present three attacks, namely "signal probability skew" (SPS) attack, "AppSAT guided removal (AGR) attack, and "sensitization guided SAT" (SGS) attack", that can break Anti-SAT and ATI, within minutes.

INDEX TERMS Hardware security, logic locking, logic encryption, IC camouflaging, boolean satisfiability, SAT

I. INTRODUCTION

A. THE NEED FOR HARDWARE IP PROTECTION

In present-day semiconductor manufacturing, integrated circuits (ICs) are designed and fabricated in a globalized multi-vendor environment, leading to concerns such as IC piracy, overproduction and counterfeiting [2]. A malicious foundry can reverse-engineer a GDSII layout file to obtain its gate-level netlist, or overbuild ICs to sell them illegally, leading to a serious economic loss to IC design companies [3], [4]. Moreover, the design may be pirated during test/assembly stages [5], or malicious circuits in the form of Hardware trojans may be embedded in the design [4]. Even an end-user may pirate the design using the state-of-the-art reverse engineering tools [6]. Reverse engineering can can extract design/technology details of an IC using imaging techniques. It involves several steps that include: depackaging an IC, delayering and imaging individual layers, and analyzing the collected images to identify design/IP details [6].

B. DESIGN-FOR-TRUST TECHNIQUES

Several design-for-trust countermeasures, including logic locking [7], IC camouflaging [8], and split manufacturing [9] have been developed to prevent IP piracy and reverse engineering attacks [10], [11]. Among these countermeasures, logic locking [5], [7], [12],–[16] and IC camouflaging [8], [17], [18] have gained significant interest from the research community as they can be easily integrated within the existing IC design flow. Moreover, as opposed to split manufacturing, both of these countermeasures provide security against reverse engineering attacks carried out by a malicious end-user. Logic locking and IC camouflaging are typically referred to as *hardware obfuscation* techniques as they obfuscate/hide critical design details from the attacker.

Logic Locking. Logic locking is a gate-level technique; a design is locked by inserting additional locking circuitry post logic synthesis [11], [20], [21] as illustrated in Figure 1. The design can be unlocked/made functional by only loading the secret key onto on-chip tamper-proof memory. An example locked netlist constructed using XOR/XNOR *key gates* is shown in Figure 2.

IC Camouflaging. IC camouflaging is a layout level technique; selected gates in the design are replaced with their camouflaged counterparts [8], [17],–[19], [22]. Camouflaged gates look identical from the top-view but can implement

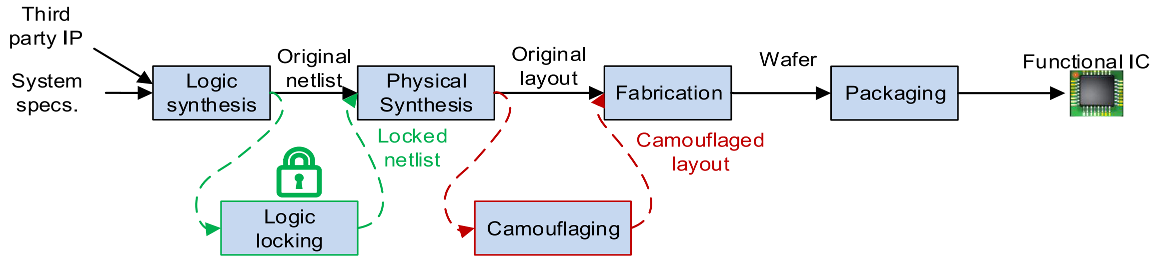

FIGURE 1. An illustration of logic locking and IC camouflaging in the context of IC design and fabrication flow.

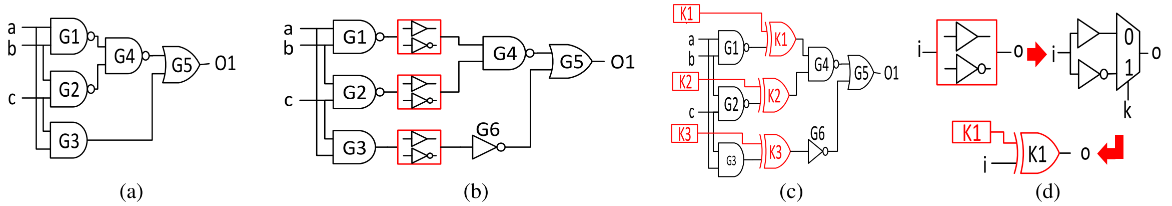

FIGURE 2. (a) The original circuit. (b) A camouflaged circuit consisting of INV/BUF camouflaged gates. (c) Equivalent logic locked circuit constructed by replacing INV/BUF camouflaged gates with XOR key gates. (d) Transforming an INV/BUF camouflaged to its logic locking counterpart using a MUX; further simplification of the MUX-based gate to an XOR key gate [19].

different functions. An example INV/BUF camouflaged gate is shown in Figure 3 [23]; the gate behaves either as a buffer or an inverter as dictated by the configuration of contacts 1 and 2 being either real or dummy. Transformations between logic locking and IC camouflaging have been proposed, enabling security analysis of both techniques using the same set of tools/algorithms [19]. Throughout this paper, we discuss the security aspects using the terminology associated with logic locking.

Traditional Locking Locking. An important research question in logic locking (and IC camouflaging) is to find the gate locations in a netlist that can be locked (or camouflaged) with maximum security benefits per unit implementation overhead. The earlier research efforts focused on developing gate selection strategies (e.g., random (RLL) [7], fault analysis-based (FLL) [14], or strong interference-based (SLL) [20]) that determine the gates to be locked (camouflaged) within the netlist [7], [8], [14], [20].

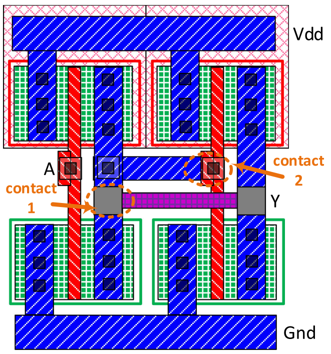

FIGURE 3. Camouflaged layout of an INV/BUF gate. The gate behaves as an inverter or a buffer based on the configuration of circled contacts. When contact 1 is real and contact 2 is dummy, the gate behaves as an inverter. The gate behaves as a buffer when contact 1 is dummy and contact 2 is real [23].

SAT Attack Resilient Logic Locking. Since the inception of a Boolean satisfiability (SAT) based attack against logic locking/camouflaging techniques, the focus of research has shifted towards developing countermeasures that offer strong resilience against the *SAT attack* [17], [18], [24] (see Section II-D for details.). The attack uses specialized *distinguishing input patterns* (DIPs) for iteratively refining the key search space. The techniques developed recently to mitigate the SAT attack include SARLock [25], Anti-SAT [26], CamoPerturb [23], and AND-tree insertion (ATI) [27] (See Section II-E for details). The fundamental theme underlying these techniques is to utilize point functions implemented by AND/NAND/OR/NOR trees to minimize the number of keys eliminated per DIP.

Removal Attack. In the SAT attack resilient techniques mentioned above, the protection circuitry (implementation of a point function) may be decoupled from the original circuit that needs to be protected, rendering these techniques vulnerable to the *removal attack*. The removal attack aims at retrieving the original circuit by identifying and removing/bypassing the protection circuitry. The first step is to identify the protection circuitry, which may be hampered due to layers of obfuscation in the design. This paper focuses on evaluating the resilience of the SAT attack resilient logic locking/camouflaging techniques against the removal attacks. The contributions of the paper are as follows:

1) We develop *signal probability skew (SPS)* attack that breaks Anti-SAT [26]. The SPS attack leverages the structural traces in the netlist to identify and remove the Anti-SAT block within minutes. The attack is scalable to large circuits; moreover, it becomes more effective with increasing key size.
2) We identify the security vulnerabilities in the ATI technique [27] and develop *sensitization guided SAT (SGS)* attack that circumvents ATI in most of the circuits by

exploiting the bias in the input distribution of the inserted AND-tree.

3) We demonstrate how SARLock [25] is vulnerable to simple removal attacks, whereas, CamoPerturb [23] exhibits resiliency against the aforementioned attacks.
4) The simple yet effective attacks we propose emphasize the importance of developing countermeasures without leaving structural traces, which could be exploited in ways much simpler than the main expected threat (i.e., the SAT attack).

II. BACKGROUND AND RELATED WORK

A. DEFINITIONS

Logic Locked Netlist. The original netlist F is a Boolean function $F : I \rightarrow O$, where $I = \{0, 1\}^n$ and $O = \{0, 1\}^m$. The locked netlist is a Boolean function $L : I \times K \rightarrow O$, where $K = \{0, 1\}^q$. Upon activation using the secret key k_s, $L(i, k_s) = F(i), \forall i \in I$. There are v key gates in L, each implementing p possible Boolean functions, determined by the key k (consisting of $\lceil log_2 p \rceil$-bits) .

Security of Logic Locking. A logic locking technique is considered secure if the effort required by an attacker to determine the correct key value k_s, or equivalently, retrieve the original circuit functionality is exponential in the number of key gates: $\mathcal{O}(2^v)$.

Camouflaged Netlist. The camouflaged netlist $C : I \times A \rightarrow O$ consists of u camouflaged gates, where the assignment $A : [1, \ldots, u] \rightarrow G$ maps each camouflaged gate to an element in G, the set of possible Boolean functions that a camouflaged gate can implement. For the correct assignment A_s, $C(i, A_s) = F(i), \forall i \in I$.

Security of IC Camouflaging. An IC camouflaging technique is considered secure if the effort required by an attacker to determine the correct assignment value A_s, or equivalently, retrieve the original circuit functionality is exponential in the number of camouflaged gates: $\mathcal{O}(2^u)$.

Transformations. Transformations between logic locking and IC camouflaging enable security analysis of both techniques using the same set of algorithms [19]. The transformation $T : C \rightarrow L$ replaces each camouflaged gate with p gates (each implementing one of the functions in G) and a $p : 1$ MUX having $\lceil log_2 p \rceil$ select inputs. The transformation for an INV/BUF camouflaged gate is illustrated in Figure 2(d). The logic locked netlist in Figure 2(c) is generated by replacing each INV/BUF in the camouflaged netlist in Figure 2(b) with its logic locking counterpart, i.e., an XOR/XNOR key gate [19].

Removal Attack. A removal attack is a transformation $R : L(I, K) \rightarrow H(I)$ such that $H(i) = F(i), \forall i \in I$. Thus, upon the removal of the protection circuit, an attacker can obtain an implementation that produces the correct output for every input irrespective of the key value. For logic locking solutions that combine two or more logic locking techniques, it is essential that an attacker is not able to target the techniques on an individual basis.

B. TRADITIONAL OBFUSCATION TECHNIQUES

In this section, we present a summary of traditional obfuscation techniques and attacks.

Logic Locking Primitives. A wide variety of logic locking primitives have been used. The *combinational* primitives include XOR/XNOR gates [7], [14], [20], [21], AND/OR gates, multiplexers, whereas the sequential primitives include look-up tables [12] and obfuscated finite state machines (FSMs) [15], [30].

IC Camouflaging Primitives. Camouflaged gates can be constructed by using real/dummy contacts [8], [31], manipulating polarities of dopants in the active regions of transistors [32],–[34], or adjusting the threshold voltage of transistors in a circuit [22]. Available spaces in the design can also be filled using metal layers and filler cells to prevent insertion of malicious logic in the design [35].

Traditional Attacks. There exist multiple attacks, applicable to both logic locking and IC camouflaging, that can compromise their security. A summary of these attacks is presented in Table 1. In the sensitization attack, key bits are individually sensitized[1] to the outputs by applying judiciously crafted input patterns. Test-data mining [5] and hill climbing attack [21] leverage the vulnerabilities associated with test data. Differential power analysis attack exploits the correlation between power consumption and key value to extract the secret key [28]. The aforementioned attacks basically rely on divide-and-conquer approaches that are no more applicable to SAT attack resilient logic locking techniques, where standalone implementations of point functions (e.g., AND/NAND trees) are integrated at with the original circuit.

C. THREAT MODEL(S)

Logic locking and IC camouflaging have slightly different threat models that differ basically in only one aspect. Logic locking assumes an *untrusted* foundry, whereas IC camouflaging assumes a *trusted* foundry. However, both techniques assume that the attacker has access to the same set of assets: a reverse-engineered netlist and a functional IC. The attacker uses computational/simulation tools on the reverse-engineered (but obfuscated) netlist, while he/she exercises the functional IC (oracle) to produce chip outputs for input patterns of interest.

The difference between logic locking and camouflaging attacks lies in when/how the attacker gets access to the required assets. Thus, both techniques can be evaluated for security on a uniform basis. In this paper, we address security from a logic locking perspective.

D. SAT ATTACK

The SAT attack is applicable to both logic locking [24] and IC camouflaging [17], [18]. As per the SAT attack

[1] Sensitization of an internal line l to an output O refers to the condition (values applied from the primary inputs to justify the side input of gates on the path from l to O to the non-controllable values of the gates) which surjectively maps l to O and thus renders any change on l observable on O.

TABLE 1. A Comparison of the attacks against logic locking.

Attack	Attacker location	Attacker assets	Attack method	Proposed defense
Sensitization [8], [20]	Foundry / end-user	1) Locked netlist 2) Functional IC	Sensitization of key bits to circuit outputs	Key-interference based logic locking [13]
SAT [17], [24]	Foundry / end-user	1) Locked netlist 2) Functional IC	SAT-based algorithm that rules out incorrect keys iteratively	AES-based [13], Anti-SAT [26], SAR-Lock [25], CamoPerturb [23], ATI [27]
Hill climbing [21]	Foundry / test facility	1) Locked netlist 2) Test data	Start with a random key CK. Flip the bits in CK based on the Hamming distance	Test-aware logic locking [21]
Test-data mining [5]	Foundry / test facility	1) Locked netlist 2) Test data	Find the key that maximizes fault coverage and satisfies test data constraints	Post-test activation [5]
Differential power analysis [28]	Foundry / end-user	1) Locked netlist 2) Functional IC	Generate a differential trace from power samples for each key value	–
AppSAT [29]	Foundry / end-user	1) Locked netlist 2) Test data	Reduce a multi-layered defense to single-layered defense by augmenting SAT attack with random oracle queries	SARLock, Anti-SAT
Signal probability skew	Foundry / end-user	1) Locked netlist	Trace the Anti-SAT block using signal skew and remove it	CamoPerturb
AppSAT guided removal	Foundry / end-user	1) Locked netlist 2) Functional IC	Use AppSAT to find FLL key bits; trace keys to identify and remove Anti-SAT	CamoPerturb
Sensitization guided SAT	Foundry / end-user	1) Locked netlist 2) Functional IC	Guide the SAT attack using patterns from sensitization attack	CamoPerturb

The attacks are also applicable to the counterpart camouflaging techniques.

threat model, the attacker has access to a reverse-engineered netlist and a functional IC [17], [18], [24]. The main idea of the SAT attack is to reveal the correct key (or the correct functionality of camouflaged gates) by selectively applying the DIPs to a functional IC [24]. The attack rules out incorrect key values by using DIPs iteratively. A DIP is an input value for which at least two unique key values, k_1 and k_2, produce differing outputs, o_1 and o_2, respectively. Since o_1 and o_2 are different, at least one of the key values is incorrect. A single DIP may rule out multiple incorrect key values, reducing the computational effort of the attack.

Example. Let us consider an example SAT attack on the logic locked circuit shown in Figure 2(c). Table 2 represents the output values of the locked circuit for different key and input combinations. The values $(k0, \ldots, k7)$ represent all possible values for three key inputs $\{K1, K2, K3\}$. When the attack is launched, it takes four DIPs to obtain the correct key. The last column in the table lists the keys eliminated in each iteration. For example, in iteration 4, the pattern 010 is used that eliminates all incorrect keys, and thus identifies $k5$ as the correct key.

TABLE 2. Analysis of the SAT attack [24] against random logic locking [7].

abc	Y	k0	k1	k2	k3	k4	k5	k6	k7	Incorrect keys identified
000	0	1	1	1	1	1	0	1	1	
001	0	1	1	1	1	1	0	1	1	
010	0	1	1	1	1	1	0	1	1	iter 4: rest incorrect keys
011	1	1	1	1	1	1	1	1	1	
100	0	1	1	1	1	1	0	1	0	
101	1	1	0	1	1	1	1	1	1	iter 3 : k1
110	1	1	1	1	1	0	1	1	1	iter 1: k4
111	1	1	1	0	1	1	1	1	1	iter 2: k2

The red entries represent the keys identified as incorrect. k5 is the correct key; the columns with all correct output values are shaded blue.

The efficiency of the SAT attack depends on the order of choosing the DIPs. The total execution time of the SAT attack comprising λ iterations with t_i as the execution time for the ith iteration is $T = \sum_{i=1}^{\lambda} t_i$ [26]. The SAT attack can be mitigated if either t_i or λ increases exponentially with the key size.

E. SAT ATTACK RESILIENT OBFUSCATION

Figure 4 presents the recent SAT attack resilient logic locking/camouflaging techniques. The underlying idea of all these techniques is to utilize point functions to control the amount of error injected into a circuit on the application of incorrect key values. A point function is a Boolean function that produces the output value 1 at exactly one point. Example implementations include AND gates and password checkers.

SARLock. As shown in Figure 4(a), SARLock integrates a comparator and a mask block with the original circuit to be protected [25]. For the correct key value, no error is injected in the circuit and the correct output is retained. For each incorrect key value, an error is injected into the circuit for only one input pattern, leading to an incorrect output for the specific pattern. Assuming that $F(I)$ is the original circuit, the output O of the circuit locked using SARLock can be presented as $O = F(I) \oplus ((I == K) \oplus (I == k_s))$,

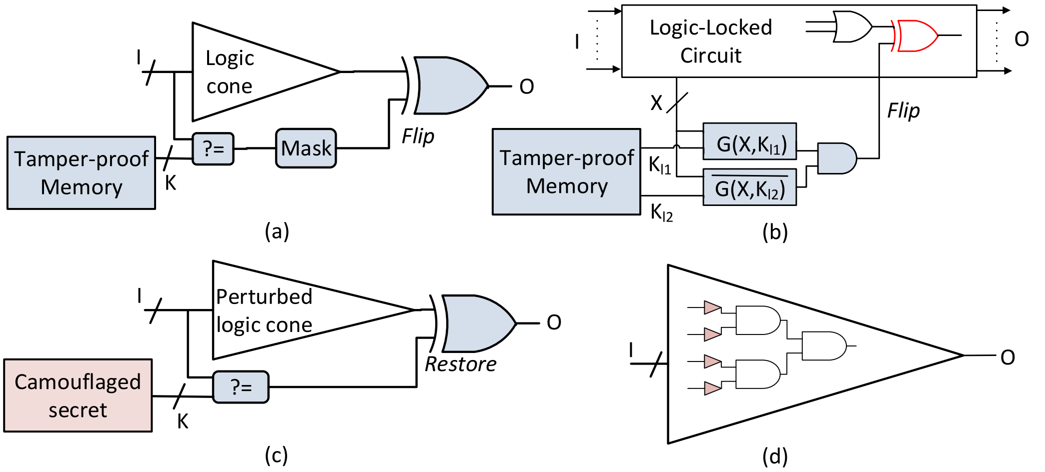

FIGURE 4. **SAT attack resilient logic locking/camouflaging techniques. a) SARLock [25], b) Anti-SAT [26], c) CamoPerturb [23], and d) ATI [27].**

where K denotes the key inputs, and k_s is the correct key value.

Anti-SAT. The Anti-SAT block shown in Figure 4(b) comprises two blocks, $B_1 = g(X, K_{l1})$ and $B_2 = \overline{g(X, K_{l2})}$ [26]. These blocks share the same inputs X, but are locked with different keys K_{l1} and K_{l2}. The outputs of B_1 and B_2 drive an AND gate to produce the output signal Y. The two blocks produce complementary outputs when correct key value is applied; for all inputs, $Y = 0$, leading to a correct output. For an incorrect key value, the output of B_1 and B_2 is 1 for a specific input pattern; for that pattern, $Y = 1$, leading to an incorrect output. Assuming that Anti-SAT protects one of the primary outputs of the original circuit $F(I)$, the protected output O can be represented as $O = F(I, K_{l0}) \oplus (g(X \oplus K_{l1}) \wedge \overline{g(X \oplus K_{l2})})$, where K_{l0} represents the key for the logic locked circuit. We elaborate on the security properties of Anti-SAT in Section III-C.

CamoPertub. In CamoPerturb, the original logic cone $F(I)$ is perturbed for exactly one minterm i_s to hide the true implementation from an attacker [23]. The output of the logic cone for the perturbed minterm is then restored using a camouflaged secret and a comparator block, as illustrated in Figure 4(c). Let $F'(I)$ represent the Boolean function for the perturbed logic cone, then $O = F'(I) \oplus (I == c_s)$, where c_s is the camouflaged secret.

AND/OR-Tree Insertion (ATI). While Anti-SAT [26], SARLock [25], and CamoPerturb [23] add external point functions to the original netlist, ATI aims at identifying these structures inside the original netlist in an attempt to decrease the implementation overhead [27]. The inputs of the identified AND/OR tree are camouflaged by inserting INV/BUF camouflaged gates as illustrated in Figure 4(d). The INV/BUF gates can be replaced with the XOR/XNOR counterparts to obtain a logic locked AND-tree. Let us assume the original circuit can be represented as being composed of two functions, $F(I) = T_{and}(I) \circ F'(I)$, where $T_{and}(I)$ is the AND-tree, $F'(I)$ is the rest of the circuit, and $\circ$ is the Boolean operator integrating the two sub-circuits. The output of the ATI circuit with the locked AND-tree can be represented as $O = T_{and}(I, K) \circ F'(I)$. We discuss the security aspects of ATI in Section I V-A.

F. SIGNAL PROBABILITY SKEW

The signal probability skew attack, to be presented in Section III-C, is based on the notion of probability skew. We define *signal probability skew* s_x of a signal x as,

$$s_x = Pr[x = 1] - 0.5, \tag{1}$$

where, $Pr[x = 1]$ indicates the probability that signal x is 1. As $0 \leq Pr[x = 1] \leq 1$, the range of s is $[-0.5, 0.5]$. The SPS of a signal denotes the amount by which a signal is distinguishable from a random guess, i.e., $\Pr[x = 1] = 0.5$. An attacker has a negligible advantage of guessing the signal value over a random guess if the corresponding SPS s is close to zero. For instance, all primary inputs and key inputs (unknown to the attacker) are equiprobable, hence their skew is zero.

Consider a two-input AND gate with inputs in_1 and in_2 with the corresponding SPS values s_1 and s_2, respectively. The SPS of the output, s_{AND} is defined as,

$$\begin{aligned} s_{AND} &= Pr[y = 1] - 0.5 = Pr[in_1 = 1]Pr[in_2 = 1] - 0.5 \\ &= 0.5(s_1 + s_2) + s_1 s_2 - 0.25. \end{aligned} \tag{2}$$

If the inputs to an AND gate have zero SPS values, then $s_{AND} = -0.25$, demonstrating the skew that every AND gate introduces. The SPS of an OR gate and an XOR gate is shown in Figure 5. It can also be noted that OR gates add a positive skew, while XOR gates reduce the absolute skew, restoring it closer to zero. XOR/XNOR key gates, where the key inputs are treated as primary inputs, introduce a skew of zero.

In MUX-based logic locking [14], the select input of a MUX is a key input with zero skew; the data inputs are intermediate signals from the original circuit. The SPS of a MUX output can be derived as,

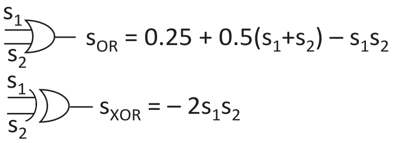

FIGURE 5. SPS of OR and XOR gate outputs where s_1 and s_2 are the SPS of the inputs of the gates.

$$s_{MUX} = 0.5(s_1 + s_2), \tag{3}$$

where s_1 and s_2 are the SPS of the inputs.

G. APPSAT ATTACK

AppSAT, a recent variant of SAT attack, aims at reducing a multi-layered defense to single- layer (e.g., Anti-SAT+FLL to Anti-SAT) [29]. The AppSAT attack builds upon the SAT attack by querying the functional IC with a fixed number of random DIPs at regular intervals and augmenting the CNF formula with new constraints based on these DIPs. The attack terminates when the Hamming distance between the correct output from the functional IC and the locked netlist is very low ($\approx \frac{1}{2^n}$), where n is the key size. Upon termination, the attack returns an approximately correct key that yields an approximate netlist [29].

While the AppSAT attack can produce only an approximate netlist, it can be used as a pre-processing attack to peel off defenses one at a time. Subsequently, other attacks can be used to obtain the *exact* netlist, as we will show in Section III-F.

III. REMOVAL ATTACK ON ANTI-SAT

A. ANTI-SAT

As already mentioned in Section 2.5, the Anti-SAT block consists of two complementary blocks $B_1 = g(X, K_{l1})$ and $B_2 = \overline{g(X, K_{l2})}$. The blocks integrated together render the SAT attack effort exponential in key size, i.e., in the number of key bits. An instance of the Anti-SAT block is shown in Figure 6 [26]. At the inputs of B_1 and B_2, a set of XOR/XNOR key gates is inserted. The number of key inputs is the same as the number of signals tapped from the logic locked circuit, i.e., $|K_{l1}| = |K_{l2}| = |X| = n$. The resulting key size is thus $2n$. The output Y is implemented as $Y = g(X \oplus K_{l1}) \wedge \overline{g(X \oplus K_{l2})}$. The output Y is 0 for all inputs when the correct keys K_{l1} and K_{l2} are applied. For incorrect keys, Y may take on the value 1, injecting error on an internal net in the netlist.

SAT Attack Resilience. The computational effort required by the SAT attack decode the $2n$ key bits is defined in terms of the number of input vectors that make the function g equal to 1, i.e., the on-set of g [26]. For an n-bit input vector $\mathbf{L} \in \{0, 1\}^n$, such input vectors are elements of the set,

$$L^T = \{\mathbf{L} | g(\mathbf{L}) = 1\}, |L^T| = p. \tag{4}$$

Anti-SAT constructs g in such a way that p is close to either 1 or $2^n - 1$. For the Anti-SAT block in Figure 6, $p = 1$. The lower bound on the number of SAT attack iterations (number of DIPs) to recover the $2n$ key bits of the Anti-SAT block is [26]

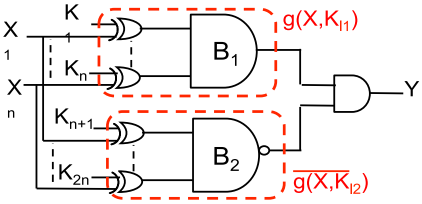

FIGURE 6. An instance of the Anti-SAT block consisting of AND/NAND trees [26].

$$\lambda_l = \frac{2^{2n} - 2^n}{p(2^n - p)}. \tag{5}$$

For $p \in \{1, 2^n - 1\}$, the number of required iterations λ_l is 2^n, i.e., exponential in the number of key bits in the Anti-SAT block. So, the SAT attack resilience of Anti-SAT hinges on p being either very small or very large. As Anti-SAT provides a provable measure to increase the SAT attack effort exponentially in key size, the conventional logic locking techniques need to be combined with the Anti-SAT block to obtain foolproof logic locking.

Secure and Random Integration. The SAT attack resilience of Anti-SAT also depends on the internal nets that drive the inputs of Anti-SAT block. Two integrations of Anti-SAT with original logic locked circuit are considered in [26]: *secure integration* and *random integration*.

Secure Integration. In this scheme, the n inputs of the Anti-SAT block are driven by n primary inputs of the logic locked circuit. The output Y is connected to a wire in the original logic locked circuit that is among the top 30 percent in observability.

Random Integration. In this scheme, the inputs as well as the output of the Anti-SAT block are connected to random wires in the logic locked circuit. The SAT attack results show that secure integration provides a higher resilience than random integration as it requires more iterations, resulting in a larger execution time to reveal the secret key [26].

B. SECURITY VULNERABILITIES IN ANTI-SAT

The main vulnerability of Anti-SAT is that it is incorporated into the netlist at a single point, where its output Y is XORed with an internal net. Therefore, Anti-SAT defense has to rely on different obfuscation schemes that make the identification of the block (and, thus, signal Y) difficult for an attacker. At the same time, SAT attack resilience is ensured by choosing a skewed p value, as dictated by Equation (5), irrespective of the structural and functional obfuscation. This basic construction principle inevitably leads to structural traces that help identify the Anti-SAT block output in a given netlist; the proposed SPS attack exploits these traces to break Anti-SAT.

C. SIGNAL PROBABILITY SKEW ATTACK

In this section, we present signal probability skew attack that detects the output signal Y of the Anti-SAT block. We show that the *absolute difference of the probability skew (ADS)* of the inputs of a gate is the maximum for the gate G, which produces the output Y of the Anti-SAT block.

Threat Model. The threat model of the SPS attack is weaker than that of the SAT attack [24] and Anti-SAT [26]. *SPS attack does not require access to a functional IC; the attack requires only a reverse-engineered netlist.* In contrast, the SAT attack requires a functional IC as well.

Let us consider the skew of individual gates in the Anti-SAT block shown in Figure 6. The XOR key gates produce zero skew signals. The blocks $g(X, K_{l1})$ and $\overline{g(X, K_{l2})}$ comprise an n-input AND and an n-input NAND gate, respectively. The SPS s_{n-AND} for the AND gate is defined as,

$$s_{n-AND} = \prod_{i=1}^{n}(0.5 + s_i) - 0.5, \tag{6}$$

where s_i is the SPS of the ith input. As $s_i = 0$, the SPS of n-input AND gate in $g(X, K_{l1})$ is,

$$s_{g(X,K_{l1})} = 0.5^n - 0.5. \tag{7}$$

For large n, $s_{g(X,K_{l1})} \approx -0.5$, indicating $p \approx 1$. Similarly, for the n-input NAND gate output in $\overline{g(X, K_{l2})}$, the SPS is,

$$s_{n-NAND} = 0.5 - \prod_{i=1}^{n}(0.5 + s_i). \tag{8}$$

As $s_i = 0$, the SPS of the NAND gate in $\overline{g(X, K_{l1})}$ is,

$$s_{\overline{g(X,K_{l1})}} = 0.5 - 0.5^n. \tag{9}$$

For large n, $s_{\overline{g(X,K_{l1})}} \approx 0.5$, indicating $p \approx 2^n - 1$. The absolute difference of the probability skew of the inputs of the AND gate G, ADS_G, can be computed as,

$$ADS_G = \left| s_{g(X,K_{l1})} - s_{\overline{g(X,K_{l1})}} \right| = 1 - 2 \times 0.5^n. \tag{10}$$

If the number of inputs to the Anti-SAT block is high, $ADS_G = |s_{g(X,K_{l1})} - s_{\overline{g(X,K_{l1})}}| \cong 1$. ADS_G close to 1 indicates that the two inputs of the gate G exhibit the highest skews but with opposite polarity. This property of gate G distinguishes it from the rest of the gates not only in the Anti-SAT block but also in the entire circuit. *The SPS attack on a logic locked circuit with the Anti-SAT block comprises computing the SPS values of all the gates in the circuit. The gate with the highest SPS value, i.e., a gate with oppositely skewed inputs is the suspect gate G, the output gate of the Anti-SAT block.* The SPS attack is described in Algorithm 1.

SPS attack applies to arbitrary g and $\bar{g}$. In case of n-input OR gate and n-input NOR gate for the functions g and $\bar{g}$, the corresponding SPS values are,

$$s_{n-OR} = 0.5 - \prod_{i=1}^{n}(0.5 - s_i), \tag{11}$$

$$s_{n-NOR} = \prod_{i=1}^{n}(0.5 - s_i) - 0.5. \tag{12}$$

The ADS_G value will again be close to 1 for large n.

SPS versus SAT Resilience. SPS attack is highly effective when $p \in \{1, 2^n - 1\}$; these values of p lead to the maximum ADS_G. One option to reduce the effectiveness of the attack is to use a value of p far from 1 and $2^n - 1$, reducing the signal skew values. However, any such attempt would make Anti-SAT vulnerable to SAT attacks as dictated by Equation (5). *Anti-SAT is thus cornered by SAT attack and the proposed SPS attack.* This is further illustrated in Section III-D3.

Algorithm 1. Signal Probability Skew Attack

```
Input: $C_{antisat}$ // Locked netlist with Anti-SAT
Output: $C_{lock}$ // Locked netlist after removing Anti-SAT block
$ADS_{arr} \leftarrow \{\}$
for $g_j \in C_{antisat}$ do
    $ADS_{arr}(g_j) \leftarrow$compute_ADS($C_{antisat}, g_j$)
end
$G \leftarrow$ find_maximum ($ADS_{arr}$) // Anti-SAT output
$Y \leftarrow$ find_value_from_skew ($G$) // Correct value of $Y$
$C_{lock} \leftarrow$ remove_TFI($C_{antisat}, G, Y$) // Remove the gates that are in TFI of gate G alone
```

Removing the Anti-SAT Block. In SPS attack, the gate G is identified using the highest ADS trace. The logic locked circuit may contain a few signals that exhibit high ADS values, close to ADS_G. These false candidates can be filtered out by checking for simple structural traces. By analyzing the transitive fan-in (TFI) of the candidate gates and eliminating the gates whose TFI does not include at least $2n$ key inputs, we can correctly identify the gate G.

Identifying Value of Y. Once G has been identified, the value of the output signal Y can be determined from s_Y. If $s_Y < 0$, the value of Y in the functional IC is 0; otherwise, it is 1. Knowing the correct value of Y, one can trace back and discard the gates that are in the fan-in of signal Y alone. The remaining circuit is re-synthesized the circuit using the correct value of Y. Upon removal, the Anti-SAT stripped circuit can be represented as $O = F(I, K_{l0})$. To identify K_{l0} for the logic locked circuit (which is locked using traditional SAT attack-vulnerable techniques such as fault analysis-based logic locking), SAT attack can be launched.

Example. The objective of the SPS attack on the circuit in Figure 10 is to identify the output gate of the Anti-SAT block, i.e., $G11$. The highest five ADS values for the circuit are shown in Table 4. The pair of complementary signals, $G8$ and $G10$ with opposite SPS values leads to the highest ADS for $G11$, enabling the precise detection of the output of the Anti-SAT block. The SPS for the output of $G11$ is $s_Y = -0.398$, implying that the signal is skewed towards 0.

D. SPS ATTACK RESULTS

1) EXPERIMENTAL SETUP

The SPS attack experiments are conducted using ISCAS benchmark circuits [36] and OpenSPARC microprocessor controllers [37]. The SPS attack and the SAT attack are executed on a server with 6-core Intel Xeon W3690 CPU,

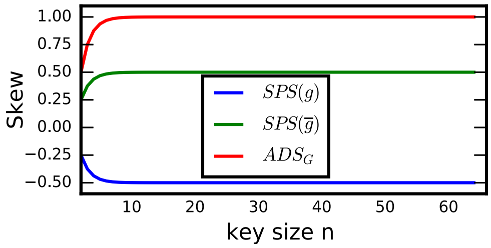

FIGURE 7. **Impact of n on ADS_G, the absolute difference of skew at the inputs of gate G, the output of Anti-SAT block, for $p = 1$. $SPS(g)$ and $SPS(\bar{g})$ represent the skew of the AND and NAND tree in the Anti-SAT block.**

running at 3.47 GHz, with 24 GB RAM [24]. The Anti-SAT block is integrated with *fault analysis based logic locking* [14], which is referred to as TOC'13(5 percent), following the convention used in [26].

2) IMPACT OF KEY SIZE (n)

The number of keys in the basic Anti-SAT block is $2n$, where n is the number of keys in the individual blocks g and $\bar{g}$. For the SPS attack to be effective, ADS_G must increase with n. Figure 7 demonstrates that as n increases, ADS_G increases exponentially initially and then saturates close to a value of 1. The SPS attack is successful when ADS_G is close to 1, representing a gate whose inputs are skewed towards opposite values. As an example, for $n = 16$, the skew at the output of the block g (an AND tree) will be ≈ -0.5, whereas the skew at the output of the block $\bar{g}$ (a NAND tree) will be ≈ 0.5. The ADS_G will be ≈ 1. For larger n values, ADS_G approaches 1 even further. Thus, *the attack effectiveness increases with n*, which is counter-intuitive for any attack.

3) SAT ATTACK VERSUS SPS ATTACK

Impact of p on Attack Success. The Anti-SAT block offers the highest resistance against the SAT attack when $p \approx 1$ or $p \approx 2^n$; then, the number of iterations for the SAT attack is $\approx 2^n$. The resistance is the least when $p \approx 2^{n-1}$. Figure 8 displays the SAT attack resistance normalized by $2^n = 65536$ for $n = 16$.

The resistance to the SPS attack can be represented as $1 - ADS_G$. When $ADS_G \approx 0$, the resistance is the maximum; this also implies p $\approx 2^{n-1}$ and the minimum resistance to the SAT attack. The resistance to the SPS attack is the minimum when $p \approx 1$ or $p \approx 2^n$ as demonstrated in Figure 8; for these values of p, the SAT attack resistance is the maximum. Thus, *the two attacks are complementary to each other. One of the attacks is highly effective for any value of p*. The regions of effectiveness of the SPS and the SAT attack are shown as red and blue regions, respectively, in Figure 8.

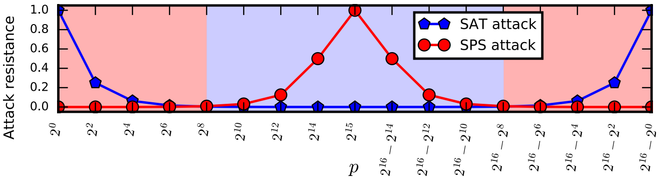

FIGURE 8. **Normalized attack resistance of the Anti-SAT block for $n = 16$. For the SAT attack, the resistance is the number of iterations of the attack normalized by 65536. For the SPS attack, the resistance is specified as $1 - ADS_G$. The SPS attack is highly effective in region shaded red; the SAT attack is effective in the region shaded blue.**

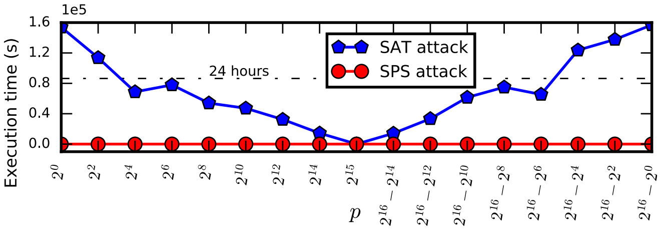

FIGURE 9. **Execution time of the SAT attack and the SPS attack on basic Anti-SAT block for $n = 16$. The execution time of the SAT attack is more than a day for $p \in \{1, 2^n - 1\}$, whereas, the execution time of the SPS attack is less than 2 minutes for all values of p.**

Attack Execution Time. Figure 9 shows that the execution time of the SAT attack depends on the value of p, which dictates the number of iterations of the attack. For $p = 1$ and $p = 65535$, the attack takes more than a day to complete. For the SPS attack, which involves computing the signal probabilities of few gates (≈ 100 for $n = 16$), the attack time is a few seconds, and practically negligible compared to the execution time of the SAT attack.

4) SPS ATTACK ON MULTI-LAYERED DEFENSE

In practical settings, the Anti-SAT block is integrated with an existing (SAT attack vulnerable) logic locking technique

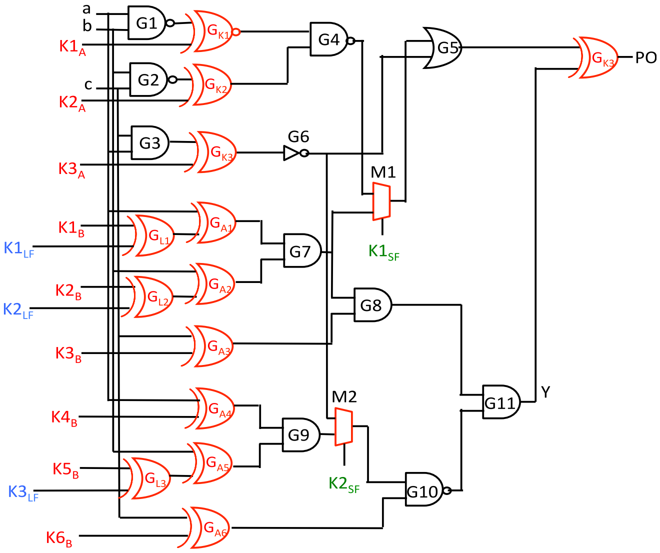

FIGURE 10. **Functional and structural obfuscation between Anti-SAT and logic locked circuit of Figure 2(c). $\{K1_A, \ldots, K3_A\}$ are the key inputs to the logic locked circuit, $\{K1_B, \ldots, K6_B\}$ are the key inputs to the Anti-SAT circuit, $\{K1_{LF}, \ldots, K3_{LF}\}$ (blue) are the key inputs for functional obfuscation, and $\{K1_{SF}, K2_{SF}\}$ (green) are the key inputs for structural obfuscation. $M1$ and $M2$ are used for MUX-based logic locking.**

TABLE 3. SPS attack on secure integration for $p = 1$.

			$n = 16$		$n = 64$	
Benchmark	# gates	HC *ADS*	$\#cand$	Exec. time (s)	$\#cand$	Exec. time (s)
fpu in	1,501	0.8125	1	0.3	1	0.6
lsu rw	1,501	0.8125	1	0.7	1	1.1
lsu excp	1,651	0.81211	1	0.9	1	0.6
s9234	1,677	0.98526	1	0.6	1	0.8
fpu div	2,137	0.8125	1	0.5	1	1.0
lsu stb	2,371	0.93749	1	1	1	0.7
c5315	2,695	0.5616	1	0.6	1	0.8
c7552	2,697	0.58069	1	0.8	1	1.1
ifu ifq	3,663	0.92281	1	2	1	1.9
tlu mmu	5,559	0.98828	1	4.8	1	4.6
s13207	13,371	0.99994	1	18.2	1	20.1
s15850	15,876	0.99999	3	18.3	1	19.1
s35932	16,457	0.60127	1	47.8	1	43.4
s38584	19,511	0.99805	1	55.7	1	56.2
s38417	22,501	0.99644	1	54.4	1	57.7
b18	111,176	0.99512	1	1300.2	1	1351.1
b19	224,511	0.99512	1	5010.3	1	5028.4

HC ADS represents the highest ADS value for the gates in the locked circuit (excluding the gates in the Anti-SAT block). $\#cand$ represents the number of candidates for gate G.

such as FLL [14]. For maximum SAT attack resistance, secure integration is utilized. In secure integration of Anti-SAT (referred to as TOC'13(5%)+n-bit BA in [26]), n inputs of the Anti-SAT block are connected to n primary inputs of the logic locked circuit [26]. ADS_G is represented as $1 - 0.5^{n-1}$, irrespective of the logic locked circuit. For a successful attack, ADS_G must be higher than the ADS of all the other gates in the circuit.

Table 3 presents the results for the SPS attack on secure integration. The column "HC ADS" displays the highest ADS value for the gates in the original circuit (excluding the gates in the Anti-SAT block). With $n = 16$, the gates with $ADS \geq (1 - 0.5^{15})$ are candidates for the gate G. We observe that there is only one candidate for gate G in all the circuits except for s15850. The circuit s15850 has two other gates whose ADS values are higher than the ADS_G.

As mentioned in Section III-B, the false candidates for G are filtered out by analyzing the TFI of the candidate gates and eliminating the gates whose TFI does not include $2n$ key inputs. The attack then correctly identifies G in all of the circuits. The execution time of the SPS attack is in the order of seconds for most of the circuits. For the largest circuit b19, which has more than 200K gates, the attack completes within an hour and a half. Thus, the attack scales well for large circuits.

E. STRUCTURAL/FUNCTIONAL OBFUSCATION IN ANTI-SAT

A trivial attack could simulate the reverse-engineered netlist and find the complementary pair of signal outputs of g and $\bar{g}$, leading to the identification and removal of the Anti-SAT block [26]. To prevent this, n additional XOR/XNOR key gates are inserted randomly at the inputs of the Anti-SAT block, obscuring the complementary relations between signals, thereby, providing *functional obfuscation*.

TABLE 4. ADS values of the gates in the Anti-SAT block in Figure 10 in descending order.

Gate	$G11$	$M1$	$M2$	$G8$	$G10$
ADS	0.6875	0.5	0.25	0.25	0.125

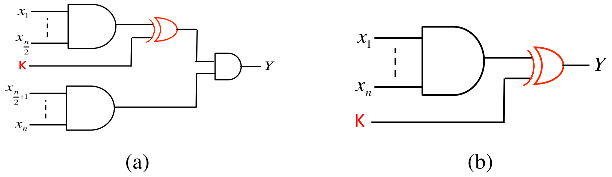

FIGURE 11. (a) Key gate inserted inside the tree of n-input AND gate; the change in probability skew assists the SPS attack. (b) Key gate inserted at the output of n-input AND gate assists the SAT attack [24].

Another simple attack could be in the form of circuit partitioning to identify the isolated Anti-SAT block and remove it from the netlist [26]. To thwart such attacks, *structural obfuscation* based on MUX-based logic locking was proposed to increase the inter-connectivity between the logic locked circuit and the basic Anti-SAT (BA) block [26]. The resultant obfuscated Anti-SAT (OA) block will have $4n$ key gates.

Example. Functional and structural obfuscation as applied to the logic locked circuit in Figure 2(c) is shown in Figure 10. The outputs of gates $G8$ and $G10$ form the output signals of the functions g and $\bar{g}$, and hence are complementary signals; an attacker can attempt to find the potential complementary pair of signals, leading to the identification of the Anti-SAT block. The Anti-SAT block, comprising an additional set of three key gates $\{G_{L1}, G_{L2}, G_{L3}\}$, obfuscates the pair of complementary signal outputs. Further, the MUXes $M1$ and $M2$ are used to increase the inter-connectivity of the logic locked circuit and the Anti-SAT block. This structural obfuscation of Anti-SAT renders the identification of the Anti-SAT block difficult for the attacker, as the boundary between the two blocks is obscured.

SPS Attack Effectiveness on Obfuscated Anti-SAT. The SPS attack is successful against obfuscated Anti-SAT (OA) as long as ADS_G values do not deviate significantly as a conseqence of obfuscation. Let us consider an n-input AND gate that constitutes the function g in the Anti-SAT block. In Figure 11(a), the XOR key gate is inserted at a net inside the AND-tree, at the input of final AND gate in this specific case. Let us assume s_1 and s_2 denote the skew at the inputs of the final AND gate. Prior to insertion of the key gate, $s_1 = s_2 = 0.5^{\frac{n}{2}} - 0.5$, and $s_{n-AND} = 0.5^n - 0.5$ for the AND-tree. After the insertion of the key gate, $s_1 = 0$, and hence the modified skew of the n-input AND becomes $s'_{n-AND} = 0.5^{\frac{n}{2}+1} - 0.5$. When the key gate is moved further to the output of AND gate as shown in Figure 11(b), $s_Y = 0$. The SPS attack, in its original form, would not be able to identify the gate G in such scenarios. Thus, by carefully inserting the key gates for functional/structural obfuscation, a designer can defend against the SPS attack. While one can develop stronger variants of the SPS attack that rely on better

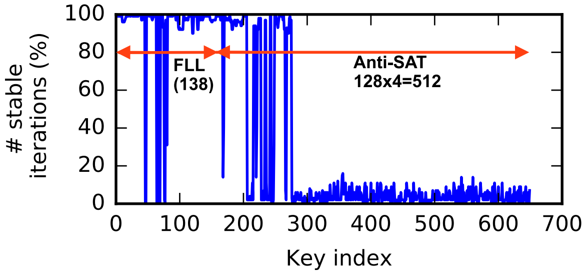

FIGURE 12. **Stability of key bits during AppSAT attack for the c5315 TOC'13(5%)+128-bit circuit. The FLL key bits mostly remain stable and are easily distinguishable from the Anti-SAT key bits.**

heuristics to guide the attack in the presence of obfuscation, in this paper, we focus our efforts on developing a strong removal attack against OA that makes use of the recently developed attack known as AppSAT [29].

F. APPSAT GUIDED REMOVAL ATTACK (AGR)

We propose AGR attack that integrates AppSAT with a simple structural analysis of the locked netlist to develop a strong removal attack on OA. As opposed to the AppSAT attack, the AGR attack recovers the *exact* netlist.

Threat Model. The threat model for the AGR attack is same as the threat model of the SAT attack [24] or ATI [27]. The attacker has access to a locked netlist and a functional IC.

The attack begins by applying AppSAT to reduce FLL+OA to OA. As the AppSAT attack terminates, the key bits corresponding to FLL settle; i.e., their values don't change over successive attack iterations. The key bit stability serves for distinguishing the Anti-SAT key bits from the FLL key bits.

Having peeled off the FLL layer, we next target the obfuscated Anti-SAT through a simple structural analysis. The Anti-SAT block has 4n key inputs, all of which converge at the gate G, the output of Anti-SAT block. We determine the gate G by tracing the transitive fan out of the Anti-SAT key inputs; it is the gate where all the 4n key bits converge.

In a real setting, AppSAT can only partially distinguish the FLL key bits from the Anti-SAT key bits. Similar to the FLL key bits, certain Anti-SAT key bits (particularly those close to the AntiSAT output) remain relatively stable over many iterations. Since the stable key bits could belong to either Anti-SAT or FLL, we use only the fluctuating key bits for structural analysis. We expect close to $C_g = 4n$ fluctuating key bits to converge at the gate G, and about $2n$ keys bits to converge at each of its inputs, which are driven by the two trees that produce the complementary functions in the Anti-SAT block. At the inputs of gate G, the ratios $R_1 = \frac{C_{in1}}{C_g}$ and $R_2 = \frac{C_{in2}}{C_g}$ are close to 0.5; here C_x represents the number of fluctuating keys that converge at a given gate. We identify the candidates for gate G by checking for this property for each gate in the circuit. If the attack yields multiple candidate gates, we sort them based on the number of key inputs that converge at a gate and pick the top-ranking candidate as the gate G. Algorithm 2 describes the AGR attack. The attack further demonstrates that simple heuristics could be used to build powerful attacks even on "provably-secure" hardware implementations.

Algorithm 2. AppSAT Guided Removal Attack

```
Input: $C_{antisat}$ // Locked netlist with Anti-SAT
Input: $n$ // Key size for Anti-SAT
Output: $C_{lock}$ // Locked netlist after removing Anti-SAT
$\#cand \leftarrow$ num_gates($C_{antisat}$)
while ($\#cand > 1$ and $!timeout$) do
    launch_appsat(4); // make 4 appsat calls
    $candidates = \{\}$
    for $g_j \in C_{antisat}$ do
        If $C_{g_j} \approx 4n$ and $R_1(g_j) \approx R_2(g_j) \approx 0.5$
            add $g_j$ to $candidates$
    end
end
$G \leftarrow$ find_maximum_key_count ($candidates$) // sort
    candidates by $C_g$ and pick the top-ranking one
$C_{lock} \leftarrow$ remove_TFI($C_{antisat}, G$) // Remove the gates that
    are exclusively in the TFI of the gate $G$
```

G. AGR ATTACK RESULTS

In this section, we present the results for the proposed AGR attack against obfuscated Anti-SAT. Following the convention used by [26], the attack results are presented for the secure integration of OA with FLL, referred to as TOC'13 (5%) + n-bit OA. Apart from the $2n$ key gates at the inputs for the Anti-SAT block, n additional XOR/XNOR key gates and n MUX key gates are inserted at the internal wires of the Anti-SAT block for functional and structural obfuscation, respectively. In our implementation, each gate in Anti-SAT has two inputs.

Key Bit Stability. Figure 12 demonstrates the stability of the key bits for the circuit c5315 when AppSAT attack is launched. The figure displays the percentage of consecutive previous iterations over which the value of a key bit has remained stable during the attack; as soon as a key bit value flips, the count for the bit is reset to zero. It can be observed that most of the Anti-SAT key bits keep fluctuating and are easily distinguishable.

Attack Success. Table 5 presents the results of the AGR attack. $\#cand$ denotes the number of valid candidates for gate G. We report $\#cand$ upon timeout of one hour to provide insights into the attack behaviour. In most of the circuits, there is only one candidate for G, demonstrating the effectiveness of the AGR attack. In a few cases, the attack may return more than one candidate for G, since certain FLL bits may not have settled yet. We observe that these candidates are often the Anti-SAT gates located close to the gate G. Upon sorting the candidate gates based on the number of key inputs that converge at the gate (C_g), we identify gate G at the top of the ranked list of candidates. Gate G was therefore identified successfully in 100 percent of the cases.

Execution Time. The attack execution time is dominated by the time for AppSAT. In our experiments, we set the

TABLE 5. AGR attack on TOC'13(5%) + n-bit OA for $n = 128$ and $p = 1$.

Benchmark	#*cand*	Rank of *G*	Exec. time (s)
s5378	1	1	8.5
ifu dcl	1	1	8.9
fpu in	1	1	10.7
lsu rw	1	1	10.7
lsu excp	1	1	8.9
s9234	1	1	9.4
fpu div	1	1	9.4
lsu stb	1	1	10.2
c5315	1	1	11.7
c7552	1	1	10.5
ifu ifq	1	1	1.0
tlu mmu	14	1	3,600.0
s13207	10	1	3,600.0
s15850	28	1	3,600.0
s35932	1	1	57.6
s38584	22	1	3,600.0
b18	710	1	3,600.0
b19	938	1	3,600.0

#cand denotes the number of valid candidates for the gate G upon a timeout of one hour. Upon sorting the list of candidates based on C_g, the gate G always has the first rank, implying 100 percent success rate.

AGR attack timeout to one hour. This is sufficient for the attack to terminate successfully since we are not interested in the values of the key bits. We rather need to classify key bits as stable or fluctuating based on their activity over successive attack iterations.

For the smaller circuits such as s5378, the attack terminates successfully within a few seconds with a single candidate. Even for the circuit b19 with more than 200K gates, the attack reduces the valid candidates to 938 within one hour. These 938 candidates are then sorted to identify gate *G* successfully.

IV. REMOVAL ATTACK ON ATI

A. AND TREE INSERTION

As opposed to Anti-SAT [26], SARLock [25], and CamoPerturb [23] that integrate external point functions with the original netlist, ATI identifies and reuses such structures inside the original netlist in order to decrease the implementation overhead [27]. Once an AND/OR tree is identified in the netlist, the inputs of the tree are camouflaged by inserting INV/BUF camouflaged gates. Alternatively, using the transformations described in Section I-B, the same tree may be *locked* by inserting XOR/XNOR key-gates at the inputs, delivering the same level of security against the SAT attack [19]. To be consistent with the previous discussion on SAT attack, we will discuss the security of ATI from a logic locking perspective. Figure 13 shows a camouflaged AND tree and its logic locked counterpart.

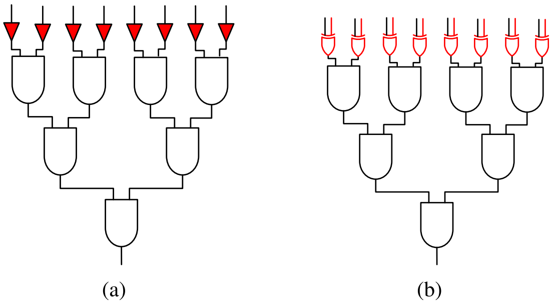
(a)
(b)

FIGURE 13. Examples of AND/OR-tree insertion: a) A camouflaged AND-tree with camouflaged INV/BUF gates inserted at its inputs [27], b) The locked counterpart of the AND-tree with XOR/XNOR key gates inserted at its inputs, using the transformations in [19]. Both trees achieve the same level of security against the SAT attack [17], [24].

TABLE 6. SAT attack resilience of ATI [27] for a 3-input AND gate with XOR key gates inserted at the inputs.

Key/DIP	0	1	2	3	4	5	6	7
k0	✓	✓	✓	✓	✓	✓	✓	✓
k1	✓	✓	✓	✓	✓	✓	×	×
k2	✓	✓	✓	✓	✓	×	✓	×
k3	✓	✓	✓	✓	×	✓	✓	×
k4	✓	✓	✓	×	✓	✓	✓	×
k5	✓	✓	×	✓	✓	✓	✓	×
k6	✓	×	✓	✓	✓	✓	✓	×
k7	×	✓	✓	✓	✓	✓	✓	×

For any DIP, the SAT attack can eliminate at most one key value: the one that injects an error at the output. ✓ denotes correct output; × denotes incorrect output.

1) ATI RESILIENCE TO SAT ATTACK

Similar to other SAT attack resilient logic locking techniques, ATI attempts to render the number of DIPs exponential in the number of key gates by controlling the distinguishing ability of individual DIPs [27]. This is illustrated in Table 6 for a 3-input AND-tree. It can be noted that exactly one incorrect key value can be eliminated by any of the input patterns, except for one special input pattern which, if applied, can identify all incorrect keys. There exists no known algorithm that can identify the special DIP from the analysis of the logic locked neltist. The number of patterns that an attacker is expected to try (in a random trial approach) prior to exercising the special input pattern is 2^{n-1}.

B. SECURITY CHALLENGES FOR ATI

There are multiple aspects that need to be considered prior to identification/insertion of logic locked AND/OR trees in order to achieve strong resilience against SAT attack.

Existence of Large Non-Decomposable Trees. The security of ATI is dictated by the size of the *largest non-decomposable* AND/OR-tree in the circuit, i.e., a tree where all internal nodes have a fanout of 1. If the internal nodes of an AND/OR-tree have multiple fanouts, an attacker can partition the tree into subtrees and attack the sub-trees on an individual basis. An example non-decomposable AND tree and a decomposable tree are presented in Figures 14(a) and 14(b), respectively. For sufficient security against the SAT attack, large non-decomposable AND/OR trees, e.g., with 64 or 128

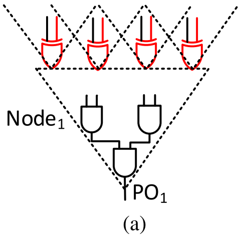

FIGURE 14. a) A non-decomposable AND tree, and b) a decomposable AND-tree [27]. Attacks on the decomposable tree can leverage divide-and-conquer strategies.

inputs, are required. Such large trees are rare in common benchmark circuits as will be illustrated in the experimental results (Section IV-D).

Bias in the Input Distribution. Contrary to the externally integrated AND/OR trees in Anti-SAT, the inputs of an internal AND/OR-tree may not be the primary inputs. Consequently, the input distribution of the tree will be biased; not all input values will be equiprobable at the tree inputs. An attacker may exploit this bias to reduce the attack effort.

Dummy AND/OR Trees. To ensure the formation of a large enough non-decomposable AND/OR tree, Li *et al.* [27] propose to insert dummy AND/OR trees in the circuit and integrate them with an original tree identified in the circuit, as illustrated in Figure 15. The dummy AND-tree $T_{dummy}(I, K1)$, with key input $K1$, is integrated with the original AND-tree in the circuit using a camouflaged OR gate. A permanent stuck-at-0 fault is introduced at the input of the OR gate by manipulating the dopant polarities [27]. With the addition of the dummy AND-tree, the output of the ATI-locked circuit can be represented as $O = F'(I) \circ T_{and}(I, K) \circ T_{dummy}(I, K1)$. However, since the inserted tree is fake and disconnected functionally from the circuit, it is prone to removal attacks. We elaborate on this in Section IV-C.

Flexibility. Another major drawback of ATI is that it can only protect the parts of a circuit where the desired AND/OR trees are present inherently. It does not offer a designer the flexibility to choose the logic to be protected.

C. SENSITIZATION-GUIDED SAT ATTACK

In this section, we present the *sensitization-guided SAT* attack that exploits the security vulnerabilities of ATI to discover the correct key values using a small number of DIPs ($\ll 2^n$). The attack consists of two main stages, *sensitization* and the *SAT attack* as illustrated in Figure 16. The sensitization stage computes *attack patterns* that are used to guide the SAT attack described in [24].

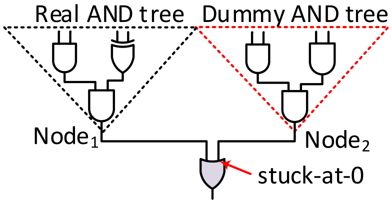

FIGURE 15. Insertion of dummy AND-tree in the circuit. A stuck-at-0 fault is introduced at the dummy input of the OR gate [27].

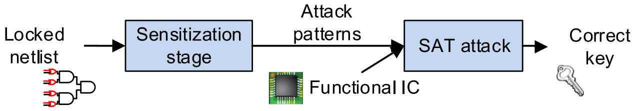

FIGURE 16. Proposed SGS attack on ATI-locked netlist. The sensitization attack computes a reduced set of attack patterns. The SAT attack uses the computed patterns in conjunction with the functional IC output to determine the correct key value.

Threat Model. The threat model for the SGS attack is same as the threat model of the SAT attack [24] or ATI [27]. The attacker has access to a locked netlist and a functional IC.

1) STAGE 1. SENSITIZATION

The objective of the sensitization stage is to compute attack patterns that are used as DIPs by the SAT attack. This stage exploits two observations about the inserted AND(/OR) tree, as illustrated in Figure 17:

1. Bias in the Input Distribution. The bias in the input distribution of an n-input AND-tree implies that the tree inputs take on only a subset of 2^n possible values. This reduction is due to the logic in the transitive fanin (TFI) of the AND-tree, i.e., the logic between the primary inputs of the circuit and the AND-tree inputs. This bias in input distribution allows an attacker to apply only a subset of DIPs, i.e., those that bring unique values to the AND-tree inputs.

2. Sensitization of the Injected Error. The AND/OR-tree introduces an error in the tree output for certain incorrect key values. However, even if an error is injected at tree output, it may not be sensitized to a primary output of the netlist; the effect of the error may be masked by the logic in the transitive fanout (TFO) of the AND-tree. In VLSI testing, detection of a stuck-at-0 (1) fault requires that the fault be a) activated by assigning a value 1 (0) to the fault location, and b) propagated to a primary output. Thus, the manifestation of the effect of an incorrect key at the primary outputs is analogous to the detection of a stuck-at fault at the output of the

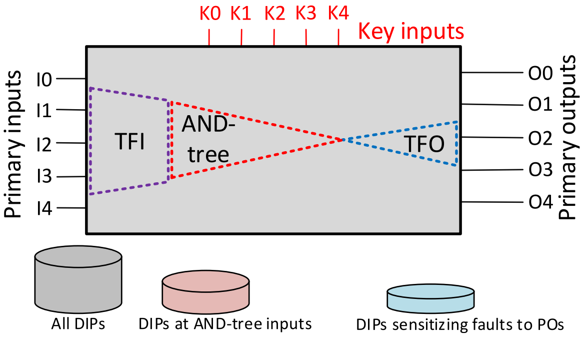

FIGURE 17. An illustration of how the sensitization stage reduces the number of the required DIPs. The gates in the TFI of the tree introduce bias and reduce the number of patterns received at the tree inputs. The gates in the TFO hamper the sensitization of errors activated at the output of the AND tree and further narrow down the attack pattern space.

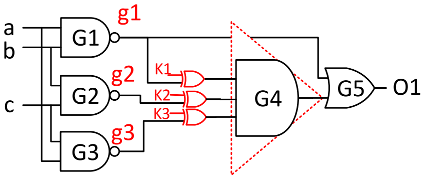

FIGURE 18. An example of pruned input pattern space as identified by the sensitization stage. The locked AND tree G4 has three inputs: g1, g2, and g3. The TFI logic prevents the tree inputs from taking on the values 001, 010, and 101. The TFO logic further reduces the number of feasible inputs; overall, only two out of eight possible input combinations, 011 and 000, are feasible for the AND tree inputs.

AND-tree. If the ATI defense was constrained to identify AND trees that directly feed a PO, the error would be guaranteed to be sensitized; however, there would be further need for dummy AND-trees as well.

Feasible Input Patterns. Overall, only a subset of total 2^n (n is number of inputs of the AND tree) DIPs are deemed feasible, i.e., they can manifest the error in the circuit output. The SAT attack uses the error at the output as a hint for identifying incorrect key values [17], [24]. The smaller the number of input patterns used by the attack, the lower the computational effort of the attack. The effectiveness of the sensitization stage is determined by the reduction in the number of attack patterns.

Example. For the netlist in Figure 18, the locked AND-tree has three inputs: g1, g2 and g3. Due to the impact of the TFI logic, the input values, 001, 010 and 101 cannot be assigned to the tree inputs. The TFO logic further narrows down the feasible input space; only two input patterns 011 and 000 are feasible for the tree inputs. Thus, the SAT attack can be launched using only two input patterns. While the reduction ratio is relatively small for this illustrative example, a significant reduction is achievable for larger circuits as will be demonstrated in Section IV-D.

2) STAGE 2. SAT ATTACK

The attack patterns computed by the sensitization stage are used to guide the SAT attack and extract the correct key by eliminating all the incorrect keys. The set of computed patterns is *sufficient for a successful SAT attack* since the set contains all the patterns that introduce observable error(s) in the circuit. The SAT attack does not need to compute any further DIPs and completes within a single iteration. The SAT solvers can inherently leverage the input bias and, apparently, render the sensitization stage redundant. However, as explained in the next section, the sensitization stage helps identify real/dummy AND trees and prevents the SAT attack from running into long trails.

3) IDENTIFYING DUMMY AND/OR TREES

To tackle the challenge of dummy AND/OR trees, we follow a simple divide-and-conquer strategy. We assume that

1) The attacker knows the location of the key gates (or alternatively, the camouflaged gates).
2) The dummy AND tree inputs are the primary inputs of the circuit (or wires close to the primary inputs) so that the issues related to the input bias are resolved [27].
3) The dummy AND-tree is large (e.g., 64 or 128) inputs.
4) None of the gates inside the dummy tree fan out to the gates in the original circuit. Only the output of AND (OR) tree is connected to a dummy OR (AND) gate; one input of the connecting OR (AND) gate is stuck-at-0 (1) [27] as illustrated in Figure 15.

Based on these realistic assumptions, which are in line with the threat model in [27], we identify a candidate dummy AND/OR tree in the netlist based on the input bias, and remove it from the netlist. To quantify the input bias precisely, we use the notion of feasible input patterns. In the sensitization stage, we compute the number of feasible input patterns DIP_{SGS} for each tree using sharpSAT solver. The tree with the higher DIP_{SGS} is assumed to be dummy. Compared to KL divergence, which is an approximate metric [27], DIP_{SGS} is a precise metric, derived from VLSI test principles, that can be efficiently computed using the sharpSAT solver. We could launch the SAT attack directly on a tree without pre-computation of DIP_{SGS}; but then the SAT attack would possibly run into long trails. Pre-computation of feasible input patterns prevents such situations. Upon removal of the dummy AND-tree, the ATI netlist reduces to $F'(I) \circ T_{and}(I, K)$, where T_{and} denotes the real AND-tree. Mounting removal attack on the real AND-tree T_{and} is not meaningful as it leads to extraction of $F'(I)$, as opposed to $F(I)$. We, therefore, proceed with the *SGS attack on the tree that is assumed to be real.* A successful SGS attack and the retrieval of the correct key validates the decision about the dummy AND-tree.

To verify the correctness of the key, we conduct the following simple test. From the correct key value returned by the attack, we can determine the input pattern for which the AND(OR)-tree will output a 1(0). We need to verify the circuit operation for only one input pattern; the tree output is a 0(1) for the rest of the input patterns. Otherwise, we repeat the experiment by switching the dummy/real trees.

D. SGS ATTACK RESULTS

In this section, we present the results for the SGS attack on ISCAS benchmark circuits [36], MCNC circuits [38], and OpenSPARC microprocessor controllers [37]. The experimental setup is the same as that for the Anti-SAT attack (presented in Section III-D1). The sensitization stage is launched using Minisat [39] solver. A miter circuit is constructed to find a pattern that can detect a stuck-at fault at the output of the AND/OR tree [40]. The CNF formula for the miter is fed to the SAT solver to compute the attack patterns.

Size of Typical AND/OR Trees (S_T). To evaluate the effectiveness of ATI, we first report the size of the largest AND/OR trees in the benchmarks circuits under study. The AND/OR trees are identified using the algorithm in [27]. We report only 22 circuits with the largest AND/OR trees. Table 7 shows that the size of the trees identified

TABLE 7. SGS attack results.

Benchmark	S_T	DIP_{EXP}	DIP_{SGS}	$R(\%)$	T_{exec}(s)
k2	104	1.01E+31	273	100.0	6.1
s38417	35	1.72E+10	NA	NA	NA
s15850	28	1.34E+08	NA	NA	NA
des	27	6.71E+07	28	100.0	1.3
s38584	27	6.71E+07	1,024	100.0	12.4
b18	26	3.35E+07	NA	NA	NA
b19	26	3.35E+07	NA	NA	NA
tlu_mmu	25	1.68E+07	NA	NA	NA
lsu_stb	24	8.39E+06	NA	NA	NA
s13207	21	1.05E+06	6	100.0	1.3
ifu_ifq	20	5.24E+05	39,680	92.4	6,269.0
c2670	18	1.31E+05	NA	NA	NA
lsu_excp	18	1.31E+05	2,624	98.0	29.3
c3540	17	65,536	4,096	93.8	100.1
c1908	14	8,192	2	100.0	1.3
c880	14	8,192	3,072	62.5	32.0
c5315	13	4,096	991	75.8	10.3
c432	12	2,048	257	87.5	3.3
fpu_in	10	512	59	88.5	2.5
ifu_dcl	10	512	510	0.4	4.6
lsu_rw	10	512	59	88.5	2.8
i8	9	256	70	72.7	9.9

S_T denotes the number of inputs of the largest AND/OR tree in the circuit. DIP_{SGS} denotes the number of attack patterns returned by the SGS attack. DIP_{EXP} is the expected number of DIPs required by the SAT attack [17], [18], [27]. R is the percentage reduction in the number of DIPs. T_{exec} denotes the execution time in seconds.

in the benchmark circuits is rather small. Only 11 out of the 22 reported circuits have a tree with 20 or more inputs. Thus, to attain sufficiently large trees, e.g., with 64 or 128 inputs, it becomes mandatory to add a dummy AND tree. In all experiments, we assume a target tree size of 64. To identify and remove the dummy AND tree, we follow the procedure described in Section IV-C3.

Percentage Reduction in DIPs. Table 7 also shows DIP_{SGS}, which denotes the number of DIPs computed by the SGS attack; these patterns are sufficient to retrieve the correct key for the target circuit. It can be observed that the number of attack patterns $\ll 2^{S_T}$, where S_T represents the size of the identified AND/OR tree. The percentage reduction R in DIP_{SGS} compared to DIP_{EXP}, computed as $R = \frac{DIP_{EXP}-DIP_{SGS}}{DIP_{EXP}} \times 100\%$, is close to 100 percent for about 50 percent of the circuits. Only a fraction of the DIP_{EXP} patterns are sufficient to break circuits, such as k2 and des, with the largest size of identified AND trees. For the same circuits, the SAT attack alone requires $DIP_{EXP} = 2^{S_T-1}$ patterns. For example, for the circuit k2 with $S_T = 104$, $DIP_{SGS} = 273$, compared to $DIP_{EXP} = 2^{103}$. The actual number of attack patterns used by the SAT attack is almost the same as DIP_{SGS}.

However, there are certain circuits, such as c2670, for which the SGS attack cannot complete within the allocated time of 10 hours (and are marked as NA). For these circuits, the bias in the input distribution of the tree is very small as most of the tree inputs are either the primary inputs of the circuit or the wires close to the primary inputs. As we discussed in Section IV-C1, the sensitization stage leverages the bias in the input distribution to attain a reduction in the number of the required DIPs. When there is zero or a very small bias in the input distribution, the attack effectiveness reduces. Alternatively, ATI can be utilized *only* for those circuits where large AND/OR trees exist close to the primary inputs. Our empirical evaluation shows large trees (with larger than 64 inputs) are rather rare; so, the designer has to resort to insertion of dummy AND trees, which can be easily removed using the proposed attacks.

Execution Time. The execution time of the SGS attack depends on the circuit size and the number of the iterations of the attack. Each iteration computes a single attack pattern. Thus, for the circuit ifu_ifq with 39,680 attack patterns, the execution time is the highest. For most of the circuits, the execution time of the attack is in the order of a few seconds. Even for the circuit k2 with a 104-input AND tree, the attack completes in 6 seconds as the number of computed attack patterns is only 273. The timeout was set to 10 hours.

V. REMOVAL ATTACKS ON SARLOCK AND CAMOPERTURB

A. SECURITY ANALYSIS OF SARLOCK

In SARLock circuit, shown in Figure 4(a), the original logic cone is implemented intact without any modifications, which makes it vulnerable to removal attacks. As already mentioned in Section II-E, in SARLock, $O = F(I) \oplus ((I == K) \wedge (I == k_s))$. An attacker has to isolate the protection circuitry comprising of an XOR, comparator and mask block; he/she can then remove the protection circuitry and extract/pirate the original IP. The comparator is functionally composed of XNOR gates and an AND tree, which can be easily identified using existing AND-tree identification algorithms [27] or the k-cut detection used in [26].

SARLock is vulnerable to the proposed SPS attack. The comparator logic comprises internally of an AND-tree, which can be identified using the skew values computed by the SPS attack. Upon the removal of the protection logic, the original function $O = F(I)$ is retrieved.

SARLock, however, is not vulnerable to the SGS attack. The effectiveness of the SGS attack depends on the bias in the input distribution. In SARLock, the comparator inputs are tied to primary inputs that do not exhibit any bias. The attack fails to achieve any reduction in the number of attack patterns.

B. SECURITY ANALYSIS OF CAMOPERTURB

As shown in Figure 4(c), the restore circuitry in CamoPerturb [23] consists only of a comparator and an XOR gate. In CamoPerturb, $O = F'(I) \oplus (I == c_s)$. Although the SPS attack can identify the comparator logic comprising the AND-tree, the removal of the protection logic leads to the retrieval of the perturbed/modified netlist $F'(I)$, as opposed to the targeted original netlist $F(I)$. The comparator inputs are connected to the primary inputs of the circuit; thus, there is no bias in the input distribution, and the SGS attack is ineffective against CamoPerturb.

TABLE 8. **Attack/Defense Matrix for SAT Attack Resilient Logic Locking Techniques and the Proposed SPS and SGS Attacks.**

	SPS+Removal	AGR	SGS+Removal
SARLock [25]	✓	✓	SARLock inputs are PIs (no bias)
Anti-SAT [26]	Obfuscation may impact SPS values	✓	Anti-SAT inputs are PIs (no bias)
ATI [27]	Dummy AND-tree identified and removed, real AND-tree identified but removal failed	Dummy AND-tree identified and removed, real AND-tree identified but removal failed	✓
CamoPerturb [23]	Restore signal identified but removal failed	Restore signal identified but removal failed	CamoPerturb inputs are PIs (no bias)

✓ denotes that a technique is vulnerable to an attack. When a technique is resilient to an attack, we provide a brief explanation. All vulnerability and resiliency expectations in this table have been experimentally validated by running each attack on each defense for our largest benchmark circuits.

C. DISCUSSION

Table 8 summarizes the vulnerability of the existing SAT attack resilient locking techniques to the proposed attacks. The proposed SPS and SGS attacks are effective for specific countermeasures, Anti-SAT and ATI, respectively. However, as the empirical results demonstrate, the execution time of both attacks is rather small. The attacks together serve as an evaluation platform that can assist designers in quickly determining the possible vulnerabilities of their logic locking/ camouflaging solutions.

According to our analysis, CamoPerturb exhibits the best security properties among all SAT attack resilient countermeasures. However, CamoPerturb protects the circuit for only one minterm. Thus, CamoPertub has to be combined with traditional logic locking/camouflaging techniques.

VI. CONCLUSION

Several countermeasures such as Anti-SAT and ATI have been developed to thwart the SAT attack, and prevent IP piracy through reverse engineering. Our security analysis identifies security vulnerabilities in the existing countermeasures. We present three simple attacks, SPS, AGR, and SGS, that can break Anti-SAT and ATI, within minutes. The proposed attacks serve as a quick evaluation platform for future logic locking and camouflaging solutions. We also provide insights for developing SAT attack resilient solutions that can withstand the proposed attacks.

ACKNOWLEDGMENTS

This work was supported in part by the Army Research Office (ARO) under Grant number 65513-CS; the US National Science Foundation, Division Of Computer and Network Systems (NSF/CNS), under Grant number 1652842; and the New York University/New York University Abu Dhabi (NYU/ NYUAD) Center for Cyber Security (CCS). A preliminary version of this paper was presented at IEEE Asia and South Pacific Design Automation Conference 2017 [1].

REFERENCES

[1] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-SAT," in *Proc. IEEE Asia South Pacific Des. Autom. Conf.*, 2016, pp. 342–347.

[2] Defense Science Board (DSB) study on High Performance Microchip Supply, 2005. [Online]. Available: www.acq.osd.mil/dsb/reports/ADA435563.pdf, Accessed on: Mar. 16, 2015.

[3] SEMI, "Innovation is at risk losses of up to $4 billion annually due to IP infringement," 2008. [Online]. Available: www.semi.org/en/Issues/IntellectualProperty/ssLINK/P043785, Accessed on: Jun. 10, 2015.

[4] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.

[5] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of logic encrypted chips: Pre-test or post-test?" in *Proc. IEEE Des. Autom. Test Eur. Conf. Exhib.*, 2016, pp. 139–144.

[6] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2011, pp. 333–338.

[7] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *IEEE Comput.*, vol. 43, no. 10, pp. 30–38, Oct. 2010.

[8] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM/SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 709–720.

[9] R. W. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2007.

[10] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in VLSI design: Design-for-trust techniques," *Proc. IEEE*, vol. 102, no. 8, pp. 1266–1282, Aug. 2014.

[11] B. Colombier and L. Bossuet, "Survey of hardware protection of design data for integrated circuits and intellectual properties," *IET Comput. Digit. Techn.*, vol. 8, no. 6, pp. 274–287, 2014.

[12] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.

[13] M. Yasin, J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.

[14] J. Rajendran, *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015.

[15] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[16] F. Koushanfar, "Provably secure active IC metering techniques for piracy avoidance and digital rights management," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 1, pp. 51–63, Feb. 2012.

[17] M. E. Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–14.

[18] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb, "Incremental SAT-based reverse engineering of camouflaged logic circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. PP, no. 99, p. 1, 2017.

[19] M. Yasin and O. Sinanoglu, "Transforming between logic locking and IC camouflaging," in *Proc. IEEE Int. Des. Test Symp.*, 2015, pp. 1–4.

[20] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2012, pp. 83–89.

[21] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in IC piracy with test-aware logic locking," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 6, pp. 961–971, Jun. 2015.

[22] M. I. M. Collantes, M. El Massad, and S. Garg, "Threshold-dependent camouflaged cells to secure circuits against reverse engineering attacks," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2016, pp. 443–448.

[23] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "CamoPerturb: Secure IC camouflaging for minterm protection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2016, pp. 29:1–29:8.

[24] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2015, pp. 137–143.

[25] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "SARlock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2016, pp. 236–241.

[26] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. Int. Conf. Cryptographic Hardware Embedded Syst.*, 2016, pp. 127–146.

[27] M. Li, *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2016, pp. 28:1–28:8.

[28] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2015, pp. 97–102.

[29] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2017, pp. 95–100.

[30] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: An overview," in *Proc. Great Lakes Symp. VLSI*, 2011, pp. 449–454.

[31] J. P. Baukus, L. W. Chow, R. P. Cocchi, and B. J. Wang, "Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing," U.S. Patent 20 120 139 582, 2012.

[32] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *Proc. Int. Conf. Cryptographic Hardware Embedded Syst.*, 2013, pp. 197–214.

[33] S. Malik, G. Becker, C. Paar, and W. Burleson, "Development of a layout-level hardware obfuscation tool," in *Proc. IEEE Annu. Symp. VLSI*, 2015, pp. 204–209.

[34] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu, "Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 1, pp. 64–77, Jan. 2017.

[35] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware ip protection," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2014, pp. 1–5.

[36] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, no. 3, pp. 72–80, July 1999.

[37] OpenSPARC T1 Processor, 2015. [Online]. Available: http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-%1444609.html, Accessed on: Nov. 1, 2015.

[38] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1989, pp. 1929–1934.

[39] N. Sorensson and N. Een, "Minisat v1.13- A SAT solver with conflict-clause minimization," *SAT J.*, vol. 2005, no. 53, pp. 1–2, 2005.

[40] T. Larrabee, "Test pattern generation using boolean satisfiability," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

MUHAMMAD YASIN (S'13) received the BS degree in electrical engineering from the University of Engineering and Technology (UET) Lahore, Pakistan and the MS degree in microsystems engineering from Masdar Institute of Science and Technology, UAE. He is working toward the PhD degree in electrical engineering at New York University, Tandon School of Engineering; and a Global PhD fellow with New York University Abu Dhabi, UAE. He has worked as a research officer with the University Technologi Petronas, Malaysia and as a lecturer with the COMSATS Institute of IT, Lahore, Pakistan. He was awarded Dean's Award for Academic Excellence by Masdar Institute of Science and Technology. He holds two pending US patents, and has published more than 10 journal and conference papers. His research interests include hardware security and design-for-trust. He is a student member of the IEEE.

BODHISATWA MAZUMDAR is presently an Assistant Professor at Indian Institute of Technology Indore. Previosuly, he was a Post Doctoral Researcher at New York University Abu Dhabi. He earned his PhD at Indian Institute of Technology Kharagpur. His research interests is in hardware IP security and the design and construction of cryptoprimitives which have a good trade-off of cryptographic properties with side-channel analysis attack resilience. He received the Best Student Paper Award in the 25th IEEE International Conference on VLSI Design 2012, Hyderabad, India.

OZGUR SINANOGLU (M'10-SM'15) received the BS degrees, one in electrical and electronics engineering and one in computer engineering, both from Bogazici University, Turkey, in 1999, and the MS and PhD degrees in computer science and engineering from the University of California San Diego, in 2001 and 2004, respectively. He is an associate professor of electrical and computer engineering at New York University Abu Dhabi. He has industry experience at TI, IBM, and Qualcomm, and has been with NYU Abu Dhabi since 2010. During his PhD, he won the IBM PhD fellowship award twice. He is also the recipient of the best paper awards at IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013. His research interests include design-for-test, design-for-security, and design-for-trust for VLSI circuits, where he has around 160 conference and journal papers, and 20 issued and pending US Patents. He has given more than a dozen tutorials on hardware security and trust in leading CAD and test conferences, such as DAC, DATE, ITC, VTS, ETS, ICCD, ISQED, etc. He is serving as track/topic chair or technical program committee member in about 15 conferences, and as (guest) associate editor for the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, the *ACM Journal on Emerging Technologies in Computing Systems*, the *IEEE Transactions on Emerging Topics in Computing*, the *Elsevier Microelectronics Journal*, the *Journal of Electronic Testing: Theory and Applications*, and the *IET Computers & Digital Techniques* journals. He is the director of the Design-for-Excellence Lab at NYU Abu Dhabi. His recent research in hardware security and trust is being funded by US National Science Foundation, US Department of Defense, Semiconductor Research Corporation, and Mubadala Technology. He is a senior member of the IEEE.

JEYAVIJAYAN RAJENDRAN (S'09-M'15) received the PhD degree in the Electrical and Computer Engineering Department, New York University, in Aug. 2015. He is an assistant professor in the Department of Electrical and Computer Engineering, University of Texas, Dallas. His research interests include hardware security and emerging technologies. His research has won the NSF CAREER Award in 2017, the ACM SIGDA Outstanding PhD Dissertation Award in 2017, and the Alexander Hessel Award for the Best PhD dissertation in the Electrical and Computer Engineering Department, NYU in 2016. He has won three Student Paper Awards (ACM CCS 2013, IEEE DFTS 2013, and IEEE VLSI Design 2012); four ACM Student Research Competition Awards (DAC 2012, ICCAD 2013, DAC 2014, and the Grand Finals 2013); Service Recognition Award from Intel; Third place at Kaspersky American Cup, 2011; and Myron M. Rosenthal Award for Best Academic Performance in MS from NYU, 2011. He organizes the annual Embedded Security Challenge, a red-team/blue-team hardware security competition and has co-founded Hack@DAC, a student security competition co-located with DAC. He is a member of the IEEE and the ACM.