Anti-SAT: Mitigating SAT Attack on Logic Locking

Yang Xie, *Student Member, IEEE*, and Ankur Srivastava, *Senior Member, IEEE*

***Abstract*—Logic locking is a technique that is proposed to protect outsourced IC designs from piracy and counterfeiting by untrusted foundries. A locked IC preserves the correct functionality only when a correct key is provided. Recently, the security of logic locking is threatened by a new attack called SAT attack, which can decipher the correct key of most logic locking techniques within a few hours even for a reasonably large key-size. This attack iteratively solves SAT formulas which progressively eliminate the incorrect keys till the circuit is unlocked. In this paper, we present a circuit block (referred to as Anti-SAT block) to enhance the security of existing logic locking techniques against the SAT attack. We show using a mathematical proof that the number of SAT attack iterations to reveal the correct key in a circuit comprising an Anti-SAT block is an exponential function of the key-size thereby making the SAT attack computationally infeasible. Besides, we address the vulnerability of the Anti-SAT block to various removal attacks and investigate obfuscation techniques to prevent these removal attacks. More importantly, we provide a proof showing that these obfuscation techniques for making Anti-SAT un-removable would not weaken the Anti-SAT block's resistance to SAT attack. Through our experiments, we illustrate the effectiveness of our approach to securing modern chips fabricated in untrusted foundries.**

***Index Terms*—Hardware IP protection, hardware security, logic locking, SAT attack.**

I. Introduction

NOWADAYS, integrated circuit (IC) design is increasingly outsourced to an offshore foundry in order to access advanced semiconductor technology at low cost. However, the outsourced design faces various security threats since the offshore foundry might not be trustworthy. Attacks by untrusted foundries on the outsourced IC design can take on many forms, such as intellectual property (IP) piracy [2], counterfeiting [3] and hardware Trojans [4]. These security threats (also known as supply chain attacks) pose a significant economic risk to most IC design companies.

Logic locking [5] is a technique that is proposed to thwart the aforementioned supply chain attacks. The basic idea is to insert additional key-controlled logic gates (key-gates), key-inputs and an on-chip memory into an IC design to hide its original functionality, as shown in Fig. 1. The key-gate can be implemented using XOR/XNOR gates [5]–[8], MUX gates [7], [9]–[11], and look-up-tables (LUTs) [12]–[14]. The key-inputs are connected to the on-chip memory and the locked IC preserves the correct functionality only when a correct key is set to the on-chip memory. To prevent attackers from probing the key values of a running chip, a tamper-proof chip protection shall be implemented.

Manuscript received February 19, 2017; revised July 14, 2017 and October 14, 2017; accepted December 20, 2017. Date of publication February 2, 2018; date of current version January 18, 2019. This work was supported in part by NSF under Grant 1223233 and in part by AFOSR under Grant FA9550-14-1-0351. This paper was recommended by Associate Editor R. Karri. *(Corresponding author: Yang Xie.)*

The authors are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: yangxie@umd.edu; ankurs@umd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2801220

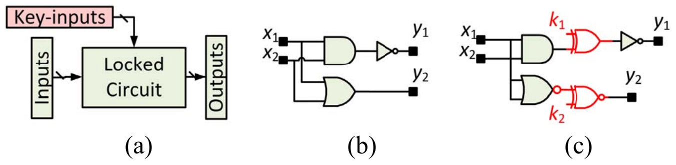


Fig. 1. Logic locking techniques. (a) Overview. (b) Original netlist. (c) XOR/XNOR-based logic locking.

A. Motivation

The security of logic locking is threatened if the correct keys can be learned by malicious foundries within a practical time. Various key-learning attacks have been proposed [1], [6], [10], [15]–[17]. Among all attacks, SAT-based attack [1] gains the most attention in recent years mainly because it can effectively break most logic locking techniques [5]–[7], [12] within a few hours even for a reasonably large key-size. The insight of the SAT attack is to iteratively solve a sequence of SAT formulas to find a set of distinguishing I/O patterns, which can progressively eliminate wrong key combinations till no one exists. The motivation of this paper is to develop a secure logic locking (SSL) design which can thwart the SAT attack.

B. Main Contribution

In this paper, we have developed a relatively lightweight circuit block (referred to as Anti-SAT block) that can be embedded into a design to mitigate the SAT attack. The Anti-SAT block is designed in a way that the total number of SAT attack iterations (and thus the total execution time) to obtain the correct key is an exponential function of the key-size in the Anti-SAT block. Thus, it can effectively prevent the SAT attack to obtain the correct key within a practical time limit. The contribution of this paper is as follows.

1) We propose an Anti-SAT circuit block to mitigate the SAT attack on logic locking. We illustrate how to construct the functionality of the Anti-SAT block and use a mathematically rigorous approach to prove that if chosen correctly, the Anti-SAT block makes SAT attack computationally infeasible (exponential in key-size).

2) The Anti-SAT block might be subject to attacks that intend to identify and nullify it, which are called *removal attacks*. We investigate a unified obfuscation technique to hide the functionality and structure of the Anti-SAT block. Also, we provide a proof showing that the obfuscation technique would not weaken the Anti-SAT block's resistance to the SAT attack.
3) Rigorous analysis and experiments on six circuits from ISCAS85 and MCNC benchmark suites have been conducted to validate the effectiveness of our proposed technique in improving the security of existing logic locking techniques.

The rest of this paper is organized as follows. Section II provides a review on SAT attack. Section III proposes the Anti-SAT block design and demonstrates its resistance against the SAT attack. Section IV outlines the unique functional and structural attributes of the Anti-SAT block that could be exploited for removal attacks and investigates corresponding obfuscation methods. Experiments and results are shown in Section V. Section VI discusses related works, followed by the conclusion in Section VII.

II. Background: SAT Attack

Logic locking is a technique that inserts a set of key-gates and key-inputs into an IC design to prevent supply chain attacks by untrusted foundries. Recently, the security of logic locking is threatened by a new attack called SAT attack [1]. Here we introduce the attack model, insight and algorithm of SAT attack.

1) Attack Model: The SAT attack model [1] assumes that the attacker is an untrusted foundry whose objective is to obtain the correct key of a locked circuit. The attacker has access the following.

1) A locked gate-level netlist, which can be obtained by reverse-engineering a GDSII layout file. This is available because the fabrication is done by the untrusted foundry which has the layout details provided by the designer. The locked netlist is represented as $\vec{Y} = f_l(\vec{X}, \vec{K})$ with primary inputs $\vec{X}$, key inputs $\vec{K}$ and primary outputs $\vec{Y}$. Its SAT formula in conjunctive normal form (CNF) is represented as $C(\vec{X}, \vec{K}, \vec{Y})$.
2) An activated functional chip, which can be obtained from open market. This IC can be used to evaluate a set of input patterns and observe their correct output patterns as a black box model.

2) Attack Insight: We first describe the basic idea of SAT attack and its iterative process for finding the correct key. The basic idea of the SAT attack is to reveal the correct key by eliminating all *wrong key combinations*. The wrong keys are identified and eliminated using a set of carefully selected inputs and their correct outputs (which can be observed from an activated chip). These special input/output pairs are referred to as distinguishing input/output (DIO) pairs. Each DIO can identify a subset of wrong key combinations. The SAT attack learns the correct key by iteratively finding such DIOs until all the wrong keys are identified. Formal definitions of wrong key combination and DIO are as follows.

Definition 1 (Wrong Key Combination): Consider the logic function $\vec{Y} = f_l(\vec{X}, \vec{K})$ and its CNF SAT formula $C(\vec{X}, \vec{K}, \vec{Y})$. Let $(\vec{X}, \vec{Y}) = (\vec{X}_i, \vec{Y}_i)$, where $(\vec{X}_i, \vec{Y}_i)$ is a correct I/O pair. The set of key combinations WK_i which result in an incorrect output of the logic circuit [i.e., $\vec{Y}_i \neq f_l(\vec{X}_i, \vec{K})$, $\forall \vec{K} \in \mathrm{WK}_i$] is called the set of wrong key combinations identified by the I/O pair $(\vec{X}_i, \vec{Y}_i)$. In terms of SAT formula, it can be represented as $C(\vec{X}_i, \vec{K}, \vec{Y}_i) = \text{False}$, $\forall \vec{K} \in \mathrm{WK}_i$.

Definition 2 (DIO Pair): In each attack iteration, the SAT attack shall find a correct I/O pair to identify a subset of wrong key combinations until none of these are left. An I/O pair at ith iteration is a DIO, denoted as $(\vec{X}_i^d, \vec{Y}_i^d)$, if it can identify a *unique* subset of wrong key combinations that cannot be identified by the previous $i-1$ DIOs, i.e., $\mathrm{WK}_i \not\subset (\cup_{j=1}^{j=i-1} \mathrm{WK}_j)$, where WK_i is the set of wrong key combinations identified by the DIO at ith iteration.

3) Attack Algorithm: Here, we briefly introduce the SAT attack algorithm for finding the DIOs. First, the algorithm will formulate a SAT formula that can be solved by state-of-the-art SAT solvers. The SAT formula F_i at ith iteration is

$$F_i := C\big(\vec{X}, \vec{K}_1, \vec{Y}_1\big) \wedge C\big(\vec{X}, \vec{K}_2, \vec{Y}_2\big) \wedge \big(\vec{Y}_1 \neq \vec{Y}_2\big) \\ \left(\bigwedge_{j=1}^{j=i-1} C\Big(\vec{X}_j^d, \vec{K}_1, \vec{Y}_j^d\Big)\right) \wedge \left(\bigwedge_{j=1}^{j=i-1} C\Big(\vec{X}_j^d, \vec{K}_2, \vec{Y}_j^d\Big)\right) \tag{1}$$

where $\vec{X}, \vec{K}_1, \vec{K}_2, \vec{Y}_1$, and $\vec{Y}_2$ are variables, and $(\vec{X}_j^d, \vec{Y}_j^d), j = 1, \ldots, i-1$ are DIOs found in previous $i-1$ iterations. A detailed explanation of this formula can be found in [1]. Basically, this formula determines whether there still exist new wrong key combinations that have not been identified by all previous $i-1$ DIOs. If F_i is satisfiable, it means that such wrong key combinations still exist. By solving the SAT formula, the SAT solver will generate an assignment to the input variable $\vec{X} = \vec{X}_i^d$, which is the distinguishing input needed to form a new DIO. After $\vec{X}_i^d$ is obtained, it is fed into an activated functional chip obtained from the open market and the correct output $\vec{Y}_i^d$ is observed. This correct I/O pair forms the ith DIO $(\vec{X}_i^d, \vec{Y}_i^d)$ which can be used to eliminate new wrong key combinations. The processing of finding DIOs is continued till no new ones can be found (assuming after λ iterations). At this point, a correct key can be obtained by solving the following SAT formula G:

$$G := \bigwedge_{i=1}^{\lambda} C\Big(\vec{X}_i^d, \vec{K}, \vec{Y}_i^d\Big). \tag{2}$$

Basically it finds a key $\vec{K}$ which satisfies the correct functionality for all the DIOs. This must be the correct key since no other DIOs exist at this point (see Definition 2). The overall SAT attack algorithm is shown in Algorithm 1.

III. Anti-SAT Block Design

The efficiency of SAT attack can be evaluated by the total execution time: $T = \sum_{i=1}^{\lambda} t_i$, where λ is the total number of SAT attack iterations and t_i is the SAT solving time for ith iteration. Consequently, the SAT attack can be mitigated if t_i is

Algorithm 1 SAT Attack Algorithm [1]

```
Input: $C$ and $eval$
Output: $\vec{K}_C$
 1: $i := 1$;
 2: $G_i := True$;
 3: $F_i := C(\vec{X}, \vec{K}_1, \vec{Y}_1) \wedge C(\vec{X}, \vec{K}_2, \vec{Y}_2) \wedge (\vec{Y}_1 \neq \vec{Y}_2)$;
 4: while sat[$F_i$] do
 5:     $\vec{X}_i^d :=$ sat_assignment$_{\vec{X}}[F_i]$;
 6:     $\vec{Y}_i^d := eval(\vec{X}_i^d)$;
 7:     $G_{i+1} := G_i \wedge C(\vec{X}_i^d, \vec{K}, \vec{Y}_i^d)$;
 8:     $F_{i+1} := F_i \wedge C(\vec{X}_i^d, \vec{K}_1, \vec{Y}_i^d) \wedge C(\vec{X}_i^d, \vec{K}_2, \vec{Y}_i^d)$;
 9:     $i := i + 1$;
10: end while
11: $\vec{K}_C :=$ sat_assignment$_{\vec{K}}(G_i)$;
```

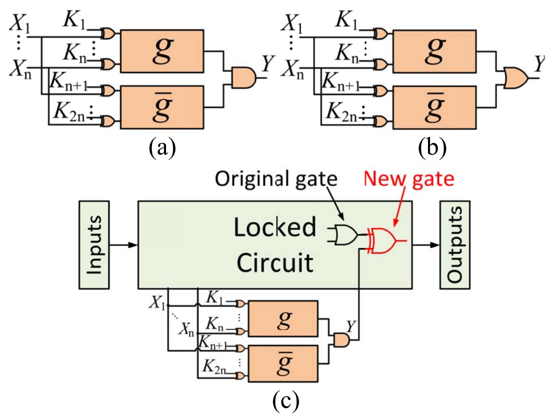


Fig. 2. Anti-SAT block configuration. (a) Type-0 Anti-SAT: always outputs 0 if key values are correct. (b) Type-1 Anti-SAT: always outputs 1 if key values are correct. (c) Integrating the Type-0 Anti-SAT block into a circuit.

large and/or λ is large. To mitigate the SAT attack, we propose to insert a relatively light-weight circuit block (referred to as Anti-SAT block) that can efficiently increase the number of iterations λ so as to increase the total execution time T.

A. *Configurations of Anti-SAT*

Fig. 2(a) and (b) illustrate two configurations of the proposed Anti-SAT block, referred to as *type-0 Anti-SAT* and *type-1 Anti-SAT*. They consist of two logic blocks g and $\overline{g}$, which share the same set of inputs $\vec{X} = (X_1 \dots X_n)$. The functionalities of g and $\overline{g}$ are complementary. A set of key-gates[1] are inserted at the inputs of two logic blocks, denoted as $\vec{K}_{l1} = (K_1 \dots K_n)$ and $\vec{K}_{l2} = (K_{n+1} \dots K_{2n})$. Hence, the key-size is $2n$. The output of g and $\overline{g}$ are fed into an AND2 gate [for Fig. 2(a)] or an OR2 gate [for Fig. 2(b)] to form the final single-bit output Y. As a result, we have $Y = g(\vec{X} \oplus \vec{K}_{l1}) \wedge \overline{g(\vec{X} \oplus \vec{K}_{l2})}$ for type-0 Anti-SAT and $Y = g(\vec{X} \oplus \vec{K}_{l1}) \vee \overline{g(\vec{X} \oplus \vec{K}_{l2})}$ for type-1 Anti-SAT.

1) Constant-Output Property: One basic property of Anti-SAT block is that when the key vector is correctly set, the output Y is a constant. Specifically, given a correct key, Y always outputs value 0 for type-0 Anti-SAT [Fig. 2(a)] and always outputs value 1 for type-1 Anti-SAT [Fig. 2(b)]. Otherwise, when a wrong key is given, Y can output either 1 or 0 depending on the inputs $\vec{X}$.

2) Correct Keys: To satisfy the constant-output property, the correct keys for the Anti-SAT block would be the ones that make type-0 Anti-SAT always output 0 and type-1 Anti-SAT always output 1. This happens when ith key-bit from $\vec{K}_{l1}$ and ith key-bit from $\vec{K}_{l2}$ have the same value, so the number of correct key combinations $c = 2^n$ for both types of Anti-SAT blocks. Since the Anti-SAT block has $2n$ keys, the total number of wrong key combinations is $2^{2n} - 2^n$.

B. *Complexity Analysis of SAT Attack on Anti-SAT*

In this section, we provide details on constructing the Anti-SAT block (i.e., the functionality of g) and analyze its impact on SAT attack complexity. We provide a rigorous mathematical analysis which gives a provable lower bound to the number of SAT attack iterations. For some constructions of g, this lower bound is exponential in the key-size thereby making the SAT-attack complexity very high. Here, we assume the Anti-SAT block alone is the circuit being attacked to decipher the $2n$ key bits.

Terminology: Given a Boolean function $g(\vec{L})$ with n inputs, assuming there exists p input vectors that make g equal to one $(1 \leq p \leq 2^n - 1)$, we can classify the input vectors $\vec{L}$ into two groups L^T and L^F, where

$$\begin{aligned} L^T &= \{\vec{L} | g(\vec{L}) = 1\}, \quad (|L^T| = p) \\ L^F &= \{\vec{L} | g(\vec{L}) = 0\}, \quad (|L^F| = 2^n - p). \end{aligned} \tag{3}$$

We denote L^T as the *on-set* of function g, L^F as the *off-set* of function g, and p as the on-set size.

Theorem 1: Assuming the on-set size p of function g is sufficiently close to 1 or sufficiently close to $2^n - 1$, the number of iterations needed by the SAT attack to decipher the correct key is lower bounded by 2^n.

Proof for Type-0 Anti-SAT [Fig. 2(a)]: Here, we show the outline of the proof. Basically, we analyze the number of wrong keys that can be identified by a DIO and use it to compute the number of iterations required to identify all the wrong keys. Detailed roof can be found in [18].

1) We first show that the wrong keys WK$_i$ that are identified by a DIO $(\vec{X}_i^d, Y_i^d)$ at ith iteration must satisfy the following conditions, because a wrong key must make a type-0 Anti-SAT output 1:

$$\begin{aligned} & g\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) \wedge \overline{g\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right)} = 1 \\ \Leftrightarrow & \left(g\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) = 1\right) \wedge \left(g\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right) = 0\right) \\ \Leftrightarrow & \left(\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) \in L^T\right) \wedge \left(\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right) \in L^F\right). \end{aligned} \tag{4}$$

Basically, (4) shows the form of wrong keys identified by the input $\vec{X}_i^d$ in ith iteration.

2) Note that $|L^T| = p$ and $|L^F| = 2^n - p$ in (3). Hence, for any given $\vec{X}_i^d$, we can select $\vec{K}_{l1}$ in p different ways and $\vec{K}_{l2}$ in $2^n - p$ different ways to satisfy the condition in (4). Thus, the total number of ways in which we can select such a wrong key is $p \cdot (2^n - p)$.

[1]Note that here we are using only XOR gates as key-gates for the sake of ease of explanation. The key-gates used could be either XOR or XNOR gates (+ inverters) based on a user-defined key [7].

3) Now we show that in any iteration i, for a given X_i^d, the *maximum* number of incorrect keys identified is $p \cdot (2^n - p)$. This is the *maximum* number because it is possible that some of these keys were identified in *previous* iterations. Hence, the number of *unique* wrong keys identified in iteration i is upper-bounded by $p \times (2^n - p)$.
4) In Section III-A2, we showed that the total number of wrong keys is $2^{2n} - 2^n$. Thus, the number of iterations λ required for SAT attack to identify all the wrong keys is lower-bounded by

$$\lambda \geq \frac{2^{2n} - 2^n}{p \times (2^n - p)}. \tag{5}$$

5) We denote this lower bound as λ_l. When $p \to 1$ or $p \to 2^n - 1$, we have the lower bound as follows:

$$\lambda \geq \lambda_l = \frac{2^{2n} - 2^n}{p \times (2^n - p)} \to \frac{2^{2n} - 2^n}{1 \times (2^n - 1)} = 2^n. \tag{6}$$

Hence, the number of iterations needed by the SAT attack to decipher the correct key is lower bounded by 2^n.

Proof for Type-1 Anti-SAT [Fig. 2(b)]: For type-1 Anti-SAT, the correct output (when provided the correct key) is always 1. Therefore, a wrong key combination $\vec{K} = (\vec{K}_{l1}, \vec{K}_{l2}) \in \mathrm{WK}_i$ which was identified by $(\vec{X}_i^d, Y_i^d)$ must produce the incorrect output as 0. This condition is described below

$$\begin{aligned} Y_i^d &= g\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) \vee \overline{g\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right)} = 0 \\ &\Leftrightarrow \left(g\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) = 0\right) \wedge \left(g\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right) = 1\right) \\ &\Leftrightarrow \left(\left(\vec{X}_i^d \oplus \vec{K}_{l1}\right) \in L^F\right) \wedge \left(\left(\vec{X}_i^d \oplus \vec{K}_{l2}\right) \in L^T\right). \end{aligned} \tag{7}$$

Based on the proof for type-0 Anti-SAT, we know that for any given $\vec{X}_i^d$, we can select $\vec{K}_{l1}$ in $2^n - p$ different ways and $\vec{K}_{l2}$ in p different ways such that $\vec{X}_i^d \oplus \vec{K}_{l1} \in L^F$ and $\vec{X}_i^d \oplus \vec{K}_{l2} \in L^T$. The total number of ways in which we can select such a wrong key is still $p \cdot (2^n - p)$, which is exactly the same as the one for type-0 Anti-SAT. Therefore, the subsequent analysis would be the same as the analysis for type-0 Anti-SAT and we can obtain the same lower bound λ_l in (6). Hence, the number of iterations required is also lower bounded by 2^n. ■

As seen in (6), if we choose a g function such that p is either very low or very high then the SAT attack would at least require an exponential number of iterations in n. Since the key-size of Anti-SAT is $2n$, the number of SAT attack iterations is also an exponential number in the key-size of Anit-SAT when g is correctly configured. One possible choice of g is indicated in Fig. 3(a), where g is chosen to be a simple n-input AND gate. For AND gates, $p = 1$ which clearly results in exponential complexity of SAT attack in n.

C. Integrating Circuit With Anti-SAT

When the Anti-SAT block is integrated into a circuit, a set of wires in the original circuit are connected to the Anti-SAT inputs $\vec{X}$ and the AntiSAT output Y is integrated to a wire in the original circuit [as shown in Fig. 2(c)]. The interconnection between the original circuit and the Anti-SAT block will affect the overall SAT attack complexity. Here, we propose a *secure integration* method: the n-bit Anti-SAT inputs $\vec{X}$ are connected to n PIs of the original circuit, and the Anti-SAT output Y is connected to a wire which is randomly selected from wires that have the top 30% observability. The impact of the Anti-SAT integration location on the overall security shall be evaluated in the experiments (Section V-A3).

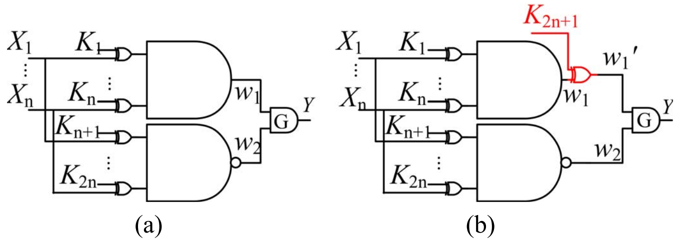


Fig. 3. Anti-SAT block design and obfuscation. (a) One possible construction of function g to ensure large number of SAT attack iterations. (b) Additional key-gate is inserted for functional obfuscation.

D. Combined With Conventional Logic Locking Techniques

While our Anti-SAT block can provide provable measures to increasing the SAT attack complexity, they may not necessarily cause substantial deviation in the chip functionality for incorrect keys. Hence, an unauthorized end user may still be able to use the chip correctly for "many" inputs (but not all). Therefore, conventional logic locking techniques need to be combined with our Anti-SAT block designs for achieving foolproof logic locking. In this paper, the original circuit is locked using the SLL, an interference-based logic locking algorithm [6]. This technique has been shown to be secure against ATPG attack [6] while obfuscating the original functionality. Let us denote the keys in the original netlist as the *conventional keys* and the keys in the Anti-SAT as the *Anti-SAT keys*. The conventional keys inserted at the original circuit can also make the Anti-SAT block less distinguishable. Without these key-gates in the original circuit, an attacker has less difficulty to locate the Anti-SAT block by inspecting the only key-inputs into the Anti-SAT block.

IV. Anti-SAT Block Obfuscation

In this section, we first analyze the security of Anti-SAT against a new type of attack called *removal attack*, which aims at identifying and nullifying the Anti-SAT block. Then, we investigate a unified obfuscation based on [13] which hides the functional and structural traces of the Anti-SAT block.

A. Removal Attacks on Anti-SAT

1) Functional Attributes-Based Removal Attacks: In Anti-SAT, the logic blocks g and $\overline{g}$ have complementary functionality. An attacker can simulate the circuit and find potential complementary pairs of signals leading to potential identification of the Anti-SAT block. Moreover, in order to guarantee exponential number of SAT attack iterations, the function g shall be configured to have very small on-set size p. Assuming $p = 1$, the outputs of $g/\overline{g}$ would be 0/1 for most of the time

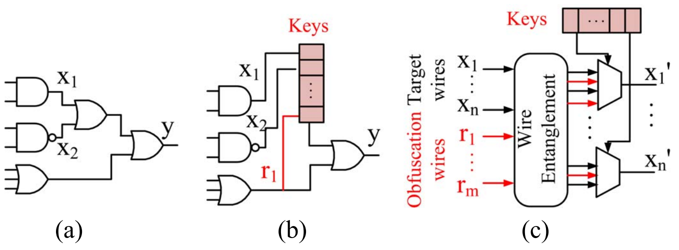


Fig. 4. Design withholding and entanglement technique. (a) Original circuit. (b) Design withholding. (c) Wire entanglement.

even when wrong keys are provided for the Anti-SAT block. In other words, the outputs of g and $\overline{g}$ will have very high signal skews of opposite polarities. This functional attribute is exploited by signal probability skew (SPS) attack citecryptoeprint:2016:896 to identify the Anti-SAT block. Another functionality attribute of Anti-SAT block is its low functionality corruptibility. Due to the construction of g and $\overline{g}$, the Anti-SAT keys normally have lower output corruptibility than the conventional keys in the original circuit. Recently, an approximate de-obfuscation technique called AppSAT [15] was proposed to learn the high-corruptibility conventional keys. An *AppSAT + netlist analysis-based removal attack* [20] has also been proposed to identify the Anti-SAT block.

2) Structural Attributes-Based Removal Attacks: In the Anti-SAT block, the internal wires in g and $\overline{g}$ do not have connections with the locked circuit. This makes the Anti-SAT block a relatively isolated and separable structure. When the size of the Anti-SAT block is roughly known, it is possible for an attacker to utilize a partitioning algorithm to partition the whole circuit into two parts while ensuring that small partition has about the same size as the Anti-SAT block. If a large portion of gates of the Anti-SAT block is moved to the small partition, then the attacker will have less difficulty to identify the Anti-SAT block using this *partitioning-based removal attack*.

B. Unified Anti-SAT Obfuscation Technique

To mitigate the removal attacks, we propose a unified obfuscation technique that obfuscates the functionality and structural attributes of Anti-SAT using design withholding and wire entanglement [13]. Fig. 4 illustrate the basic idea of design withholding and wire entanglement. In design withholding [Fig. 4(a)], a portion of design is replaced with a set of LUT's to ensure that the original design detail is not available to the untrusted foundry. Hence, design withholding technique can be used to hide both the functionality and implementation detail of the Anti-SAT. Design entanglement is another obfuscation technique that aims at obfuscating the interconnect structure of an IC design by using a wire-entanglement module, as shown in Fig. 4(b). The basic idea of wire-entanglement module is to entangle n target wires with r obfuscation wires using MUX-based interconnect network. When the selection bits of the MUXes are correctly configured, the wire-entanglement module will represent the original interconnection. The wire-entanglement module is useful for obfuscating the interconnect structure between the Anti-SAT block and the original netlist.

1) Security Implication: Fig. 5 illustrates the overall obfuscation for Anti-SAT based on design withholding and wire entanglement. The design withholding technique is used to hide the functionality of the Anti-SAT block and part of the original netlist. When obfuscated using LUTs, the signal skews of g and $\overline{g}$ would be significantly reduced, so the SPS attack cannot effectively identify the Anti-SAT block. Note that the obfuscation for g and $\overline{g}$ does not necessarily need to be balanced (i.e., the key-size of g and $\overline{g}$ can be different) as long as the outputs of g and $\overline{g}$ do not have a high signal skew. This can help mitigating the AppSAT + netlist analysis attack which assumes both g and $\overline{g}$ would have roughly the same key-size and use it as a hint to locate the Anti-SAT.

On the other hand, wire-entanglement technique is used to obfuscate the interconnection between the original circuit and the Anti-SAT block to prevent the partitioning-based attack. With the wire-entanglement module, the interconnections between the Anti-SAT block and the locked circuit will be increased and it is difficult for an attacker to partition and isolate the Anti-SAT block from the locked circuit. Besides, the design withholding and entanglement technique can be designed to mitigate the AppSAT + netlist analysis attack. This is because that after wire entanglement, new signal paths would be created which fanout many low-corruptibility keys to the original netlist. Therefore, gates in the original netlist can have many low-corruptibility keys in their fan-in cones. It increases the difficulty of the netlist analysis phase which tries to identify the Anti-SAT by counting the number of low-corruptibility keys in a gate's fan-in.

With regard to SAT attack, since these techniques are based on LUT-based and MUX-based logic locking, they can be modeled in SAT formula and attacked by the SAT attack. However, we will use a proof to show that the number of SAT attack iterations for unlocking a circuit with an obfuscated Anti-SAT will not be reduced. Experimental results in Section V-C also validate this analysis.

2) Performance Overhead: These two obfuscation techniques will inevitably increase the performance overhead. For example, an n-input, m-output LUT would require $O(m \times 2^n)$ gates. The key-size for such LUT is $m \times 2^n$. An n-input, m-output wire entanglement module would require m number of n-input MUXes. The key-size for such module is $m \times \log(n)$. To reduce overhead, we can use multiple small LUTs/MUXes (with less inputs and outputs) to form large LUTs/MUXes, as suggested in [13]. A more light-weight obfuscation solution may be explored in future research.

C. SAT-Attack Resistance of Anti-SAT After Obfuscation

In Section IV-B, a unified obfuscation technique for Anti-SAT block based on withholding and entanglement [13] is discussed. It basically obfuscates the Anti-SAT block by adding additional logic gates and key-inputs. Here, we use a rigorous proof to shows that the resistance of Anti-SAT block would not be weakened after when obfuscation technique is

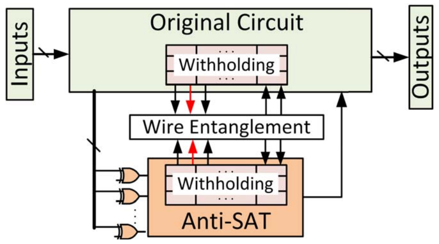


Fig. 5. Anti-SAT obfuscation based on design withholding and wire entanglement.

applied. In other words, adding addition key-gates and key-inputs will not reduce the number of SAT attack iterations required to decipher the Anti-SAT block.

We first show that adding one additional key-gate for obfuscation will not reduce the number of SAT attack iterations required for unlocking the Anti-SAT. Without loss of generality, we use XOR/XNOR-based key-gates instead of LUT-based key-gates to obfuscate the Anti-SAT in order to simplify the proof. Also, we insert the extra key-gate as shown in Fig. 3(b).

Theorem 2: Assuming a new key-gate K_{2n+1} is inserted into the Anti-SAT block (with $p = 1$) for obfuscation [as shown in Fig. 3(b)], the number of SAT attack iterations needed by the SAT attack to decipher the correct key will remain to be 2^n.

Proof: Let us first derive the equation which represents the wrong key combinations that can be identified by a DIO $(\vec{X}_i^d, \vec{Y}_i^d)$. Notice that for any input combinations (including the distinguishing inputs $\vec{X}_i^d$), the correct output (when provided a correct key) is 0 for type-0 Anti-SAT. Therefore, a wrong key combination which is identified by $(\vec{X}_i^d, \vec{Y}_i^d)$ must result in incorrect output as 1. This condition is described as

$$\Big[(K_{2n+1} = 0) \wedge \Big(g\Big(\vec{X}_i^d \oplus \vec{K}_{l1}\Big) = 1\Big) \wedge \Big(\overline{g\big(\vec{X}_i^d \oplus \vec{K}_{l2}\big) = 1}\Big)\Big] \vee \Big[(K_{2n+1} = 1) \wedge \Big(g\Big(\vec{X}_i^d \oplus \vec{K}_{l1}\Big) = 0\Big) \wedge \Big(\overline{g\big(\vec{X}_i^d \oplus \vec{K}_{l2}\big) = 1}\Big)\Big]. \tag{8}$$

This is equivalent to

$$\Big[(K_{2n+1} = 0) \wedge \Big(\vec{X}_i^d \oplus \vec{K}_{l1} \in L^T\Big) \wedge \Big(\vec{X}_i^d \oplus \vec{K}_{l2} \in L^F\Big)\Big] \vee \Big[(K_{2n+1} = 1) \wedge \Big(\vec{X}_i^d \oplus \vec{K}_{l1} \in L^F\Big) \wedge \Big(\vec{X}_i^d \oplus \vec{K}_{l2} \in L^F\Big)\Big]. \tag{9}$$

Note that the function g in Fig. 3(b) is an n-input AND gate

$$L^T = \{(11\ldots11)\}, L^F = \mathbb{B}^n \backslash L^T \tag{10}$$

where $\mathbb{B}^n$ is the set of all n-bit Boolean vectors, and $\mathbb{B}^n \backslash L^T$ means every n-bit Boolean vector except $(11\ldots11)$.

We can see that when $K_{2n+1} = 0$, to satisfy (9), we need to ensure $\vec{X}_i^d \oplus \vec{K}_{l1} \in L^T$ and $\vec{X}_i^d \oplus \vec{K}_{l2} \in L^F$. Since L^T has only one vector $(11\ldots11)$, for any $\vec{X}_i^d$, we only have one way of selecting $\vec{K}_{l1}$ to make $\vec{X}_i^d \oplus \vec{K}_{l1} = (11\ldots11)$, that is $\vec{K}_{l1} = \neg\vec{X}_i^d$ (bit-wise negation), i.e.,

$$\vec{K}_{l1}[j] = \neg\vec{X}_i^d[j], \ j = 1\ldots n. \tag{11}$$

On the other hand, since $L^F = \mathbb{B}^n \backslash L^T$, for any $\vec{X}_i^d$, we have $2^n - 1$ ways to select $\vec{K}_{l2}$ such that $\vec{X}_i^d \oplus \vec{K}_{l2} \in L^F$, those are $\vec{K}_{l2} \in \mathbb{B}^n \backslash \neg\vec{X}_i^d$.

Therefore, the wrong key combinations (with $K_{2n+1} = 0$) identified by $(\vec{X}_i^d, Y_i^d)$ has the following form:

$$\Big(\vec{K}_{l1} = \neg\vec{X}_i^d, \vec{K}_{l2} \in \Big(\mathbb{B}^n \backslash \neg\vec{X}_i^d\Big), K_{2n+1} = 0\Big). \tag{12}$$

We can see that since $\vec{K}_{l1} = \neg\vec{X}_i^d$, there exists an one-to-one matching between each pair of $\vec{X}_i^d$ and $\vec{K}_{l1}$. In other words, any $\vec{X}_i^d$ value (from 0 to $2^n - 1$) can identify a unique set of wrong key combinations in a form of (12). It is unique because that $\vec{K}_{l1} = \neg\vec{X}_i^d$ and different $\vec{X}_i^d$ would result in different $\vec{K}_{l1}$. Therefore, every input combination (from 0 to $2^n - 1$) is a distinguishing input because it can identify a unique set of wrong key combinations that can only be identified by it. Thus, the SAT attack requires 2^n DIOs (i.e., 2^n iterations) to identify all wrong key combinations. ■

We now show that adding more than one additional key-gates for obfuscation still does not reduce the number of SAT attack iterations required for unlocking the Anti-SAT.

Theorem 3: Assuming n_{obf} new key-gates are inserted into the Anti-SAT block (with $p = 1$) for obfuscation, the number of SAT attack iterations needed by the SAT attack to decipher the correct key will be 2^n.

Proof: The proof for Theorem 2 shows that after adding a new key-gate to the Anti-SAT block for obfuscation [Fig. 3(b)], the number of SAT attack iterations remains to be 2^n. Notice that this conclusion is also true when n_{obf} additional key-gates are inserted to the Anti-SAT for obfuscation. Let us denote the extra keys for obfuscation as $\vec{K}_{\text{obf}}$ and its correct key is $\vec{K}_{\text{obf}}^C$. Based on (12), we can conclude that any Anti-SAT input combination is a distinguishing input $\vec{X}_i^d$ because it can identify a unique set of wrong keys which is

$$\Big(\vec{K}_{l1} = \neg\vec{X}_i^d, \vec{K}_{l2} \in \Big(\mathbb{B}^n \backslash \neg\vec{X}_i^d\Big), \vec{K}_{\text{obf}} = \vec{K}_{\text{obf}}^C\Big) \tag{13}$$

and this set of wrong keys can only be identified by this input $\vec{X}_i^d$. Hence, the SAT attack requires at least 2^n DIOs (i.e., 2^n iterations) to identify all wrong key combinations. Therefore, adding additional key-gates at different locations for obfuscation will not weaken the Anti-SAT's resistance to the SAT attack. ■

V. Experiments and Results

In this section, we evaluate the security level of our proposed Anti-SAT blocks. The security level is evaluated by the number of *SAT attack iterations* as well as the *execution time* to infer the correct key. SAT attack tools and benchmarks used are from [1]. The SAT attack tool uses the Lingeling [21] SAT solver. The CPU time limit is set to 10 h as [1]. The experiments are running on an Intel Core i5-2400 CPU with 16 GB RAM.

A. Anti-SAT Block Design

1) On-Set Size p: Table I illustrates the impact of p on the security level of 16-bit Anti-SAT blocks (type-0 and type-1). For both types of Anti-SAT, when $p \to 1$ and $p \to 2^{16} - 1 =$

TABLE I
IMPACT OF p ON THE SECURITY LEVEL OF ANTI-SAT (WHEN $n = 16$)

	p	1	81	243	2187	30375	63349	65293	65455	65535
Type-0	# Iterations	-	10675	4760	901	273	898	4647	-	-
Anti-SAT	Time (s)	timeout	16555.8	8746.12	174.743	3.24	307.104	12932.3	timeout	timeout
Type-1	# Iterations	-	-	4853	877	285	881	4691	-	-
Anti-SAT	Time (s)	timeout	timeout	3559.96	55.108	3.148	187.896	1048.19	timeout	timeout

TABLE II
IMPACT OF n ON THE SECURITY LEVEL OF ANTI-SAT (WHEN $p = 1$)

	n	8	10	12	14	16
Type-0	# Iterations	255	1023	4095	16383	-
Anti-SAT	Time (s)	1.14569	20.024	324.727	4498.03	timout
Type-1	# Iterations	255	1023	4095	16383	-
Anti-SAT	Time (s)	1.06	14.612	273.1	3658.76	timeout

TABLE III
COMPARISON BETWEEN SECURE AND RANDOM INTEGRATION

	n	8	12	16
Random	Avg. # Iteration	151	1748	11461
	Avg. Time (s)	1.4296	162.529	10272.4
Secure	# Iteration	255	4095	-
	Time (s)	3.452	759.924	timeout

65535, the SAT attack algorithm fails to unlock the Anti-SAT block in 10 h. This is because that it requires a large number of iterations to rule out all the incorrect key combinations. As $p \to 2^{16}/2$ (the worst case), the SAT attack begins to succeed using less and less iterations and execution time for both types of Anti-SAT. This result validates out analysis in (6), which shows that for a fixed n, when p is close to 1 or $2^n - 1$, λ will be large and the SAT attack will fail within a practical time limit.

2) Input-Size n: As shown in (6), λ_l is an exponential function of n when p is very low ($p \to 1$) or very high ($p \to 2^n-1$). Table II shows the exponential relationship between λ and n when $p = 1$ for type-0 and type-1 Anti-SAT block. It can be seen that as n increases, the simulated SAT iterations and execution time grows exponentially.

In the following experiments, we select type-0 Anti-SAT and construct an *n-bit baseline Anti-SAT block* ($n - bitBA$) using an n-input AND gate ($p = 1$) as the logic block g to ensure large number of iterations. However, notice that this is not the only possible choice for g.

3) Secure Integration of Anti-SAT: Here, we compare two approaches of integrating the Anti-SAT block with the original circuit, namely *secure integration* and *random integration*. For the *secure integration*, n inputs of the Anti-SAT block $\vec{X}$ are connected to n PIs of the original circuit. The output Y is connected to a wire which is randomly selected from wires that have the top 30% observability. For the *random integration*, the inputs $\vec{X}$ are connected to random wires of the original circuit, and the output Y is connected to a random wire. For both cases, the wire for Y has a later topological order than that of the wires for $\vec{X}$ to prevent combinational loop. Table III shows the results for two integration approaches when three n-bit BA ($n = 8, 12, 16, p = 1$) are integrated into the c1355 circuit from ISCAS85. It can be seen that secure integration is better than random integration as the former requires more iterations ($\sim 2\times$) and execution time ($\sim 3\times$) for the SAT attack algorithm to reveal the key. Therefore, in the following experiments, we adopt the secure integration as the way to integrate the Anti-SAT block into a circuit.

TABLE IV
BENCHMARK INFORMATION OF SIX CIRCUITS FROM ISCAS85 AND MCNC

Circuit	#PI	#PO	#Gates	Key-size	
				SLL	n-bit BA
c1355	41	32	546	29	2n
c1908	33	25	880	46	
c3540	50	22	1669	86	
dalu	75	16	2298	119	
des	256	245	6473	336	
i8	133	81	2464	130	

B. Anti-SAT Block Application

We evaluate the security level of the Anti-SAT block when it is applied to six circuits from ISCAS85 and MCNC benchmark suites. The benchmark information is shown in Table IV. We compare two logic locking configurations as follows.

1) *SLL:* The original circuit is locked only using the SLL, an interference-based logic locking algorithm [6]. This technique has been shown to be secure against ATPG attack [6] while obfuscating the original functionality.
2) *SLL (5%) + n-bit BA:* The original circuit is locked with SLL with 5% area overhead. Besides, an n-bit BA is integrated into the locked circuit using the secure integration (described in Section III-C).

We compare the security level of two configurations when the same number of keys are used in each configuration.[2] The SAT attack results of two configurations with respect to increasing key-size are shown in Fig. 6. It can be seen that for SLL, increasing the key-size cannot effectively increase SAT attack complexity. For all benchmarks locked with SLL, they can be easily unlocked using at most 67 iterations and 1070.85 s. On the other hand, when the Anti-SAT blocks are integrated, the SAT attack complexity increases exponentially with the key-size in the Anti-SAT block. The SAT attack fails to unlock the circuits within 10 h when a 16-bit BA is integrated (as shown by the fifth data point with respect to x-axis).

C. Anti-SAT Block Obfuscation

In Section IV-C, we have shown that after obfuscation, the security of Anti-SAT against SAT attack would not be undermined. To validate this proof, we obfuscate the Anti-SAT block using LUTs and MUXes. Fig. 7 shows the SAT attack results of c1355 circuit when obfuscation techniques are

[2]For SLL, the extra key-gates are inserted to the original circuit. For SLL(5%) + n-bit BA/OA, the extra key-gates are used in the Anti-SAT block and increasing the key-size also indicates increasing the input-size n because we construct the n-bit BA with key-size $k_{\text{BA}} = 2n$. In this experiment, we experiment the n-bit BA with $n = 8, 10, 12, 14, 16, 18, 20$. The key-sizes are shown in Table IV.

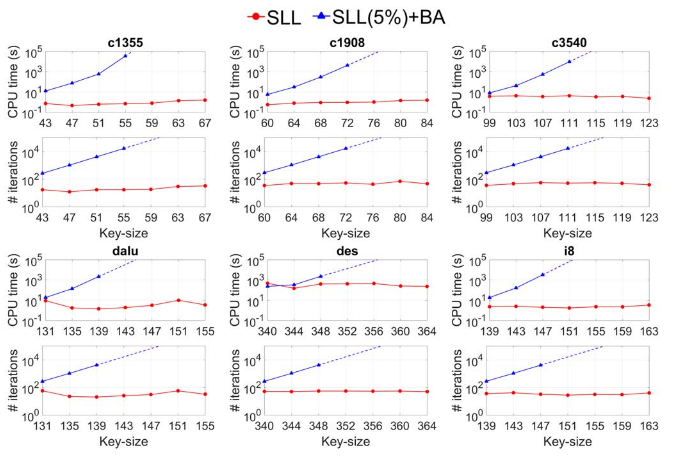


Fig. 6. SAT attack results on six benchmarks with three logic locking configurations: SLL and SLL(5%) + n-bit BA. Timeout is 10 h (3.6×10^4 s). The dashed lines are the curve fitting results when the SAT attack has time-outed after certain key-size.

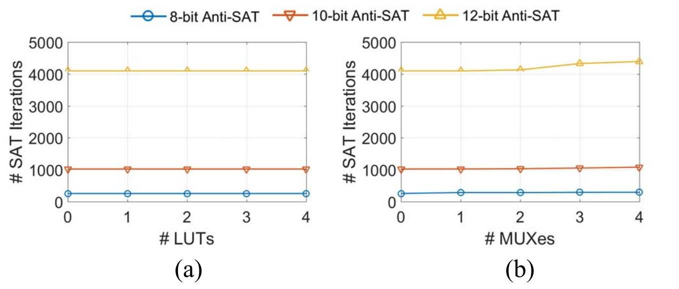


Fig. 7. SAT attack results of c1355 circuit when obfuscation techniques are applied. (a) Design withholding. (b) Wire entanglement. For both techniques, the number of SAT attack iterations required are $\geq 2^n$ after obfuscation, where n is the input-size of Anti-SAT.

applied. Fig. 7(a) shows the result for LUT-based design withholding technique. Here, we increasingly replace the 2-input AND gates in g and $\overline{g}$ with LUTs and evaluate its impact on the SAT attack iteration. As seen, when the number of LUTs is increased, the SAT attack iteration remains to be 2^n, where n is the input-size of Anti-SAT ($n = 8, 10, 12$). Fig. 7(b) shows the results for MUX-based wire entanglement technique. Here we use 2-input MUXes, where one input of MUX comes from the original circuit and the other input comes from the Anti-SAT block. As seen, the SAT attack iteration is $\geq 2^n$ for different choices of n. The number of iterations could be larger than 2^n because the MUXes enlarge the fan-in cones of the Anti-SAT block. These results confirm that the proposed obfuscation technique will not hamper the SAT attack resilience of Anti-SAT.

D. *Performance Overhead of the Anti-SAT Block*

Different implementation of g and $\overline{g}$ will result in different overhead. In our experiments, we utilize an n-bit AND gate and an n-bit NAND gate to implement the function g and $\overline{g}$, each consists of $n - 1$ AND2 gates. The estimated area for a n-bit BA is $4n$ additional gates. Since the number of SAT attack iteration required is 2^n, a slight increase in area overhead of the Anti-SAT block can result in exponential increase in SAT attack complexity. To counter removal attacks, we investigate both the design withholding and entanglement techniques. These two obfuscation techniques will inevitably increase the performance overhead. However, we present it as the first unified obfuscation technique to make various removal attacks harder. A more light-weight solution may be explored in future research.

VI. Related Work

A. *SAT-Attack Resilient Logic Locking*

Recent years have seen an increasing number of research work on mitigating the SAT attack on logic locking. Yasin *et al.* [22] proposed to add an AES circuit into a locked circuit which aims at increasing the SAT solving time. Although this approach is effective, the AES circuit leads to a significant performance and area overhead since a standard AES circuit implementation requires a large number of gates [23]. In [24], a technique called SARLock was proposed which can make the number of SAT attack iterations grow exponentially in key-size. SARLock is similar to Anti-SAT, however, it has been shown to be vulnerable to some variants of SAT attack called double-DIP [25] or bypass attack [26]. On the contrary, these attacks cannot break Anti-SAT (when the combination of SLL and Anti-SAT locking is used), as analyzed in [25]. Xu *et al.* [26] proposed a binary decision diagram (BDD) based design technique to achieve exponential number of SAT attack iterations. However, the disadvantage of the BDD-based technique is that it will result in a very significant area overhead, because the size of the BDD is almost always exponential in the key-size as shown in [26]. To increase the difficulty of SAT formulation, Shamsi *et al.* [27] proposed a cyclic logic locking technique which introduces nonreducible combinational loops to the locked circuit. However, the cyclic logic locking technique was shown vulnerable to a variant of SAT attack called CyclicSAT [28]. Besides conventional logic locking, a new set of locking techniques called parametric locking is proposed [29], [30]. The parametric locking techniques aim at obfuscating the parametric behavior of the circuit, such as power, delay, and reliability etc. For incorrect keys, the locked circuit will malfunction or have degraded performance.

B. *SAT Attack on IC Camouflaging*

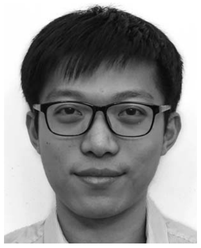
IC camouflaging is a reverse-engineering prevention technique that hides a circuit's functionality with camouflaging cells. It has been shown that SAT attack can also be applied to recover the functionality of the camouflaging cells [31], [32]. To counter the SAT attack, various countermeasures have been proposed [33], [34], which aim at making the de-camouflaging effort exponentially harder in the number of camouflaged gates.

VII. Conclusion

In this paper, we present a circuit block called Anti-SAT to mitigate the SAT attack on logic locking. We show that the iterations required by the SAT attack to reveal the correct key in the Anti-SAT block is an exponential function of the key-size in the Anti-SAT block. The Anti-SAT block is integrated to a locked circuit to increase its resistance to the SAT attack. A unified obfuscation technique has been proposed to protect the Anti-SAT block from removal attacks, such as the SPS attack and the partitioning-based attack. Overall, our proposed Anti-SAT-based logic locking can effectively thwart the SAT attack and various removal attacks.

REFERENCES

[1] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2015, pp. 137–143.
[2] 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.
[3] U. Guin *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.
[4] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
[5] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Conf. Design Autom. Test Europe*, 2008, pp. 1069–1074.
[6] J. J. V. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. 49th ACM Annu. Design Autom. Conf.*, 2012, pp. 83–89.
[7] J. J. V. Rajendran *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015.
[8] Y. Xie, C. Bao, Y. Liu, and A. Srivastava, "2.5D/3D integration technologies for circuit obfuscation," in *Proc. 17th Int. Workshop Microprocessor SOC Test Verification (MTV)*, 2016, pp. 39–44.
[9] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using PUF-based logic," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2014, pp. 270–277.
[10] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in IC piracy with test-aware logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 961–971, Jun. 2015.
[11] Y.-W. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Proc. 16th IEEE Latin-Amer. Test Symp. (LATS)*, 2015, pp. 1–6.
[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] S. Khaleghi, K. Da Zhao, and W. Rao, "IC piracy prevention via design withholding and entanglement," in *Proc. IEEE 20th Asia South Pac. Design Autom. Conf. (ASP DAC)*, 2015, pp. 821–826.
[14] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *Proc. Conf. Design Autom. Test Europe*, 2014, pp. 1–6.
[15] K. Shamsi *et al.*, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2017, pp. 95–100.
[16] A. Chakraborty, Y. Xie, and A. Srivastava, "Template attack based deobfuscation of integrated circuits," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2017, pp. 41–44.
[17] 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. (DFTS)*, 2015, pp. 97–102.
[18] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, 2016, pp. 127–146.
[19] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-sat," in *Proc. 22nd Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Chiba, Japan, 2017, pp. 342–347.
[20] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2017.2740364.
[21] A. Biere, "Lingeling, plingeling and treengeling entering the SAT competition 2013," in *Proc. SAT Competition*, vol. 51, 2013, pp. 51–52.
[22] M. Yasin, J. J. V. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.
[23] HelionTechnology. (2015). *High Performance AES (Rijndael) Cores for ASIC*. [Online]. Available: http://www.heliontech.com/downloads/aes_asic_helioncore.pdf
[24] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2016, pp. 236–241.
[25] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. ACM Great Lakes Symp. VLSI*, 2017, pp. 179–184.
[26] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, 2017, pp. 189–210.
[27] K. Shamsi *et al.*, "Cyclic obfuscation for creating SAT-unresolvable circuits," in *Proc. ACM Great Lakes Symp. VLSI*, 2017, pp. 173–178.
[28] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Irvine, CA, USA, 2017, pp. 49–56.
[29] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," in *Proc. 54th ACM Annu. Design Autom. Conf.*, 2017, pp. 1–6.
[30] M. Yasin *et al.*, "What to lock?: Functional and parametric locking," in *Proc. ACM Great Lakes Symp. VLSI*, 2017, pp. 351–356.
[31] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. 22nd Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2015, pp. 1–14. [Online]. Available: http://internetsociety.ong/sites/default/files/06_2_2.pdf
[32] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb, "Incremental SAT-based reverse engineering of camouflaged logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1647–1659, Oct. 2017.
[33] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. 35th ACM Int. Conf. Comput.-Aided Design*, 2016, p. 28.
[34] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. J. V. Rajendran, "CamoPerturb: Secure IC camouflaging for minterm protection," in *Proc. 35th ACM Int. Conf. Comput.-Aided Design*, 2016, p. 29.



Yang Xie (S'14) received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2013. He is currently pursuing the Ph.D. degree in the Electrical and Computer Engineering Department, University of Maryland, College Park, MD, USA.

His current research interests include hardware security, trustworthy hardware design, IP piracy prevention, and 3-D IC security.

Ankur Srivastava (S'00–M'02–SM'15) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, New Delhi, India, in 1998, and the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 2002.

He is currently a Professor with the ECE Department with joint appointment with the Institute for Systems Research, University of Maryland, College Park, MD, USA. His current research interests include high performance, low power and secure electronic systems and applications, such as computer vision, data and storage centers, and sensor networks. He has published numerous papers on these topics at prestigious venues.

Dr. Srivastava was a recipient of the Outstanding Dissertation Award from the CS Department of UCLA in 2002. He has been a part of the technical program and organizing committees of several conferences, such as International Conference on Computer Aided Design, Design Automation Conference, International Symposium on Physical Design, International Conference on Computer Design, Great Lakes Symposium on VLSI, International Symposium on Hardware Oriented Security and Trust, and others. He has served as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, *VLSI Journal*.