Provably Secure Camouflaging Strategy for IC Protection

Meng Li, *Student Member, IEEE*, Kaveh Shamsi, *Student Member, IEEE*, Travis Meade, *Student Member, IEEE*, Zheng Zhao, Bei Yu, *Member, IEEE*, Yier Jin, *Member, IEEE*, and David Z. Pan, *Fellow, IEEE*

***Abstract*—The advancing of reverse engineering techniques has complicated the efforts in intellectual property protection. Proactive methods have been developed recently, among which layout-level integrated circuit camouflaging is the leading example. However, existing camouflaging methods are rarely supported by provably secure criteria, which further leads to an over-estimation of the security level when countering latest de-camouflaging attacks, e.g., the SAT-based attack. In this paper, a quantitative security criterion is proposed for de-camouflaging complexity measurements and formally analyzed through the demonstration of the equivalence between the existing de-camouflaging strategy and the active learning scheme. Supported by the new security criterion, two camouflaging techniques are proposed, including the low-overhead camouflaging cell generation strategy and the AND-tree camouflaging strategy, to help achieve exponentially increasing security levels at the cost of linearly increasing performance overhead on the circuit under protection. A provably secure camouflaging framework is then developed combining these two techniques. The experimental results using the security criterion show that camouflaged circuits with the proposed framework are of high resilience against different attack schemes with only negligible performance overhead.**

***Index Terms*—Active learning, AND-tree, integrated circuit (IC) camouflaging, provably secure, SAT-based attack.**

I. Introduction

With the increase of integrated circuit (IC) design costs, intellectual property (IP) privacy and infringement becomes a significant concern for the semiconductor industry. One of the major threats arises from reverse engineering (RE) [1]–[4]. By stripping the IC layer by layer, the gate-level netlist can be extracted and duplicated without the authorization of the IP holder [4]. To protect IC design against RE, IC camouflaging is proposed as a layout-level technique to hide the circuit functionality [5]–[13]. By synthesizing circuits with logic cells that look alike but can have different functionalities (also known as camouflaging cells), the functionality of original circuits cannot be determined from physical RE.

Existing work on IC camouflaging mainly falls into the following three categories: 1) fabrication level [5]–[7], [14]; 2) cell level [8], [15]–[18]; and 3) gate netlist level [8], [19]. Fabrication-level camouflaging mainly focuses on developing fabrication techniques that can hide the circuit structure. In [5], a doping-based technique is proposed to create Always-on and Always-off transistors by changing the dopant polarity of the source and drain. A dummy contact-based method is also proposed to control the connection between two adjacent layers [7]. Both doping-based and contact-based methods are shown to be robust against existing RE techniques [5], [7]. Cell-level camouflaging leverages the fabrication techniques to build camouflaging cells that look alike but may have different functionalities. In [8] and [15], by controlling the doping scheme or contact configurations, camouflaging cells or lookup tables are created with multiple possible functionalities. Gate netlist level camouflaging seeks to develop camouflaging cell insertion algorithms to maximize the resilience of the circuit netlist against RE techniques given predefined overhead constraints. For example, Rajendran *et al.* [8], Lee and Touba [19], and Shamsi *et al.* [20] inserted interfered camouflaging cells or camouflaging connections to prevent RE. In [21]–[23], new low output-corruptibility camouflaging strategies are further proposed to protect the circuit from more advanced RE techniques [24], [25].

Despite the extensive researches on IC camouflaging, there are still fundamental problems that have not been properly solved. First, due to the lack of provably secure criteria to guide IC camouflaging, existing methods tend to over-estimate the provided security level and are shown vulnerable to the SAT-based de-camouflaging attacks as well as removal attacks based on structural and functional information [24]–[27]. Second, the insertion of camouflaging cells usually leads to large overhead, which places significant limits on their usage in commercial applications.

In this paper, we propose a new criterion, defined as de-camouflaging complexity, to directly quantify the security of the camouflaged netlist. We further identify two key factors that determine the security of a camouflaged netlist, i.e., the number of possible functionalities of a camouflaged netlist and the output Hamming distance between different functionalities.

Manuscript received January 29, 2017; revised May 4, 2017 and July 24, 2017; accepted September 1, 2017. Date of publication September 7, 2017; date of current version July 17, 2019. This paper was recommended by Associate Editor C. H. Chang. *(Corresponding author: Meng Li.)*

M. Li, Z. Zhao, and D. Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: meng_li@utexas.edu; zzhao@utexas.edu; dpan@utexas.edu).

K. Shamsi and Y. Jin are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: kshamsi@ufl.edu; yier.jin@ece.ufl.edu).

T. Meade is with the Department of Computer Science, University of Central Florida, Orlando, FL 32816 USA (e-mail: travm12@knights.ucf.edu).

B. Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

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.2017.2750088

To increase the security level of the camouflaged netlist, we propose two camouflaging strategies targeting at the two identified factors, including a new low-overhead camouflaging cell generation strategy and an AND-tree based strategy. We summarize our contributions as follows.

1) We investigate a new security criterion to quantify the de-camouflaging complexity and identify two key factors that can help enforce the security criterion in the camouflaged netlist.
2) We propose two novel camouflaging strategies to increase the two identified factors.
3) We develop an IC camouflaging framework combining the two strategies to further protect the camouflaged circuits against removal attacks.
4) We verify our proposed security criterion and framework against state-of-the-art de-camouflaging techniques and demonstrate great resilience with negligible overhead.

The rest of this paper is organized as follows. Section II provides a review of existing de-camouflaging attacks and the preliminaries on active learning scheme. Section III formally builds the equivalence between SAT-based de-camouflaging and active learning with key security factors identified. Sections IV and V describe the camouflaged cell generation strategy and the AND-tree structure. Section VI proposes an IC camouflaging framework. Section VII demonstrates the performance of the proposed camouflaging framework, followed by conclusion in Section VIII.

II. Background

In this section, the RE attack model and attack techniques are reviewed. We also talk about the active learning scheme, which lays the foundation for our analysis on de-camouflaging complexity in Section III.

A. Reverse Engineering Attacks

For an attacker, the main target of RE is to extract the original or equivalent circuit with RE techniques. We follow the widely used attack model and assume the attackers have access to the camouflaged netlist, which can be acquired from the physical RE procedure [4], and a black-box functional circuit. Given the functional circuit, the attackers can select a sequence of input vectors, import them into the circuit through the scan chain, query the functional circuit and observe the corresponding outputs. Attackers will infer the correct circuit functionality based on the collected input–output pairs. To explore the input–output patterns, three different methods have been proposed, including brute force attack [8], testing-based attack [8], [31], and SAT-based attack [24]–[26], [32].

Brute force attack proposes to randomly sample input vectors for the logic simulation to rule out the false functionalities but suffers from scalability problem [8]. Testing-based attack generates the input patterns so that the output of all camouflaging gates that interfere with the target gate is known, and a change at the output of the target gate can be observed at circuit primary outputs [8]. However, by inserting gates that interfere with each other deliberately, the complexity of testing-based attack is no better than the brute force attack.

The SAT-based attack is currently the most powerful de-camouflaging attack method. By iteratively searching the input patterns that can differentiate different circuit functionalities, denoted as discriminating inputs [24], false functionalities are identified and ruled out. As shown in [24], the SAT-based attack significantly reduces the number of iterations required for the de-camouflaging procedure.

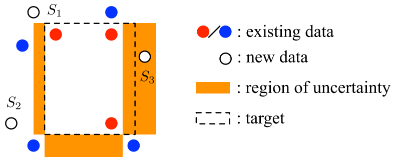


Fig. 1. Example of sampling strategy for active learning.

Since all the attack strategies described above rely on querying the functional circuit, we denote them as query-based attacks. Besides the query-based attacks, new attack scheme proposed in [27] tries to leverage the structural and functional footprint of the camouflaging strategies to resolve the original circuit functionality. The new attack scheme can work collaboratively with the SAT-based attack as well. Until now, no camouflaging strategy has systematically demonstrated convincing resilience against both attack schemes.

B. Active Learning Scheme

In this section, we provide basic definitions concerning active learning. For more detailed description, interested readers can refer to [29].

Considering an arbitrary domain X where a concept h is defined to be a subset of points in the domain, a point $x \in X$ can be classified by its membership in concept h, that is, $h(x) = 1$ if $x \in h$, and $h(x) = 0$ otherwise. A concept class H is a set of concepts. For a target concept $t \in H$, a training sample is a pair $(x, t(x))$ consisting of a point x, which is drawn from X following distribution $\mathcal{D}$, and its classification $t(x)$. A concept h is defined to be consistent with a sample $(x, t(x))$ if $h(x) = t(x)$.

The intuition of active learning is to regard learning as a sequential process, so as to choose samples adaptively. Consider a set S of m samples. The classification of some regions of the domain can be determined, which means all concepts in H that are consistent with S will produce the same classification for the points in these regions. Active learning scheme seeks to avoid sampling new points from these regions, and instead, samples only from the regions that contain points which can have different classifications for different concepts in H, denoted as the region of uncertainty $\mathcal{R}(S)$. By iteratively sampling from $\mathcal{R}(S)$ and updating $\mathcal{R}(S)$ based on the new sample, t can be learned from H. We use the following example to illustrate the concept of active learning.

Example 1: Consider a 2-D space, and the target t is a set of points lying inside a fixed rectangular in the plane as shown in Fig. 1. Assuming we already have some samples with their classification, $\mathcal{R}(S)$ can then be decided. Consider the three points s_1, s_2, and s_3 in Fig. 1, the label for s_1 and s_2 can already be determined based on existing samples. Therefore, only s_3 can help provide further information to decide the target t from the concept class H.

According to [29], if we define error rate $\text{er}_{x\sim\mathcal{D}}(h, t)$ for a concept h with respect to the target t and the distribution

$\mathcal{D}$ of points x as $\text{er}_{x\sim\mathcal{D}}(h,t) = \Pr_{x\sim\mathcal{D}}[h(x) \neq t(x)]$, then by adaptively sampling from $x \in X$, to guarantee $\text{er}_{x\sim\mathcal{D}}(h,t) \leq \epsilon$ with sufficient probability, the number of samples m needed for active learning is

$$m = \mathcal{O}\left(\theta d \log\left(\frac{1}{\epsilon}\right)\right)$$

where d is a measure of the capacity of H. Especially, when X is Boolean domain with $X = \{0,1\}^n$ and the concept class contains only Boolean function, we have $d \geq \left(\log_2|H|/n\right)$ [33]. Here, $|\cdot|$ denotes the cardinality of the set. θ is the disagreement coefficient, defined as

$$\theta = \sup_\epsilon \frac{\Pr_{x\sim\mathcal{D}}[\text{DIS}(H_\epsilon)]}{\epsilon}$$

where $H_\epsilon = \{h \in H : \text{er}_{x\sim\mathcal{D}}(h,t) \leq \epsilon\}$, $\text{DIS}(H_\epsilon) = \left\{x : \exists h, h' \in H_\epsilon \text{ s.t. } h(x) \neq h'(x)\right\}$, and $\Pr_{x\sim\mathcal{D}}[\text{DIS}(H_\epsilon)] = \Pr_{x\sim\mathcal{D}}[x \in \text{DIS}(H_\epsilon)]$.

III. IC Camouflaging Security Analysis

Let c_o be the original circuit before camouflaging. c_o has n input bits with the input space $I \subseteq \{0,1\}^n$ and l output bits with output space $O \subseteq \{0,1\}^l$. Define the indicator function $e_{c_o} : I \times O \to \{0,1\}$ for c_o, where $I \times O = \{(i,o) : i \in I, o \in O\}$, as

$$e_{c_o}(i,o) = \begin{cases} 1, & \text{if } c_o(i) = o \\ 0, & \text{otherwise} \end{cases}$$

where e_{c_o} indicates whether an output vector o can be generated by c_o given certain input vector i.

During the process of IC camouflaging, $\widetilde{m}$ camouflaging gates are inserted into the original netlist, whose functionalities cannot be resolved by physical RE techniques. Let G denotes the set of all possible functionalities for the camouflaging gate, where $\forall g \in G$, $g : \{0,1\}^{\widetilde{n}} \to \{0,1\}$ with $\widetilde{n}$ as the input number of the camouflaging gate. Let y denotes $\widetilde{m}$ functions chosen from G, i.e., $y \in G^{\widetilde{m}}$, which assigns each camouflaging gate a function in G and let Y denotes the set of all possible y. Depending on y, a set of possible circuit functionalities can be created, denoted as C. Note that $c_o \in C$. $\forall c \in C$, there exists a corresponding indicator function e_c. Let E_C denotes the set of indicator functions for all $c \in C$.

Based on the attack model described in Section II, after physical RE, the attacker can acquire the camouflaged netlist but cannot resolve the functionality of the camouflaging cells. Equivalently, the attackers can acquire C and E_C from physical RE. For the attackers, to resolve $c_o \in C$ is equivalent to resolving $e_{c_o} \in E_C$. The attacker can select input pattern $i \in I$, apply to the black-box functional circuit through circuit scan chain and get the correct output $c_o(i)$. Based on $(i, c_o(i))$, all $c \in C$ that are not consistent with $(i, c_o(i))$ can be pruned.

To evaluate the effectiveness of camouflaging and the hardness of de-camouflaging, we define the de-camouflaging complexity as the number of input–output patterns required to rule out the false functionalities and resolve $c_o \in C$, equivalently $e_{c_o} \in E_C$. To evaluate the de-camouflaging complexity, we build the equivalence between the SAT-based de-camouflaging strategy and the active learning scheme as follows.

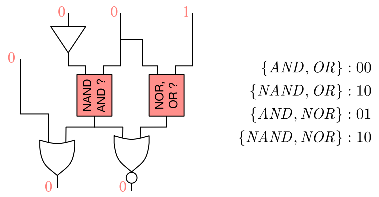
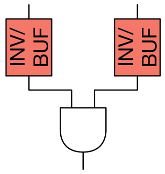


Fig. 2. Example of the camouflaged netlist.



Input	c_0	c_1	c_2	c_3
00	0	0	0	1
01	0	0	1	0
10	0	1	0	0
11	1	0	0	0

Fig. 3. Example of the camouflaged netlist and the truth table for all the possible functionalities.

1) The set of indicator functions of all possible circuit functionalities E_C corresponds to the concept class H.
2) The supply of indicator functions, i.e., $I \times O$, corresponds to the set of points X.
3) The indicator function of the original circuit functionality e_{c_o} corresponds to the target concept t.
4) The input–output relation $((i, c(i)), 1)$ corresponds to the samples $(x, t(x))$.
5) The SAT-based de-camouflaging strategy corresponds to the selective sampling strategy.

Based on the equivalence, the number of input–output patterns required to resolve e_{c_o} with less than ϵ error rate and sufficiently high probability is

$$m(e_{c_o}, E_C) = \mathcal{O}\left(\theta d \log\left(\frac{1}{\epsilon}\right)\right) \quad (1)$$

where $d \geq [(\log_2|E_C|)/n]$ is related to the number of functionalities in E_C. θ is calculated as

$$\theta = \sup_\epsilon \frac{\Pr_{(i,o)\sim I\times O}[(i,o) \in \text{DIS}(E_\epsilon)]}{\epsilon} \quad (2)$$

where $E_\epsilon = \{e_c \in E_C : \text{er}_{(i,o)\sim I\times O}(e_c, e_{c_o}) \leq \epsilon\}$ consists of all the indicator functions that are different from e_{c_o} with probability less than ϵ, and $\text{DIS}(E_\epsilon) = \{(i,o) : \exists e_c, e_{c'} \in E_\epsilon \text{ s.t. } e_c(i,o) \neq e_{c'}(i,o)\}$ consists of all the input–output pairs (i,o) that lead to different outputs of any pair of indicator functions in E_ϵ. We use the following example to illustrate E_ϵ and $\text{DIS}(E_\epsilon)$.

Example 2: Consider the camouflaged circuit and the truth table of all the possible functionalities of the camouflaged circuit shown in Fig. 3. The correct functionality is c_0 with $y^* = \{\text{BUF}, \text{BUF}\}$. Then, for c_0, the indicator function e_{c_0} becomes

$$e_{c_0}(i,o) = \begin{cases} 1, & \text{if } (i,o) \in \{(00,0),(01,0),(10,0),(11,1)\} \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, we can define e_{c_i} for c_i, $1 \leq i \leq 3$. e_{c_1} has different outputs compared with e_{c_0} at four input–output pairs, i.e., $\{(10,0),(10,1),\ (11,0),(11,1)\}$. If we assume (i,o) follows a uniform distribution, then $\text{er}_{(i,o)\sim I\times O}(e_{c_0}, e_{c_1}) = 4/8 = 1/2$. Similarly, we have $\text{er}_{(i,o)\sim I\times O}(e_{c_0}, e_{c_2}) =$

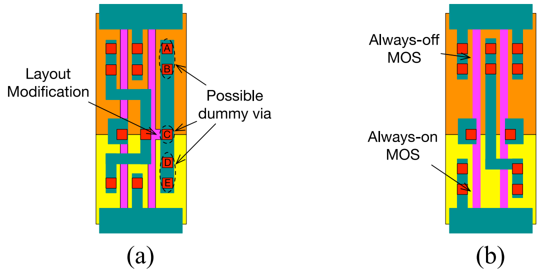


Fig. 4. Examples of two different cell camouflaging strategies: (a) XOR-type and (b) STF-type.

$\mathrm{er}_{(i,o)\sim I\times O}(e_{c_0}, e_{c_2}) = 1/2$. If we set $\epsilon = 1/2$, $E_{1/2} = \{e_{c_0}, e_{c_1}, e_{c_2}, e_{c_3}\}$. We can also determine $\mathrm{DIS}(E_{1/2}) = \{(00, 0), (00, 1), (01, 0), (01, 1), (10, 0), (10, 1), (11, 1), (11, 1)\}$ and $\Pr_{(i,o)\sim I\times O}[(i, o) \in \mathrm{DIS}(E_{1/2})] = 1$. By trying different ϵ, we know θ get the maximum value, i.e., 2, when $\epsilon = 1/2$.

Based on the equivalence of SAT-based attack and active learning, we can identify two key factors that impact the security of different camouflaging strategies, i.e., d and θ, and also use (1) as a quantitative security evaluation metric. It should be noted that (1) refers to a specific scenario defined as probably approximately correct (PAC) learning, which indicates the output of active learning scheme is an approximation of the original functionality with a certain probability. However, the SAT-based attack is an exact learning scheme. Though different, the exact learning problem is at least as difficult as the PAC learning problem, which indicates (1) can still work as a lower bound of the complexity of the SAT-based attack. In the following sections, we will propose camouflaging techniques to increase d and θ to enhance the resilience against SAT-based attack.

IV. Novel Camouflaging Cell Design

In this section, we target at increasing d as in (1). Because the lower bound of d is in proportional to $|C|$, i.e., $|E_C|$, we choose to increase the number of possible functionalities of the camouflaged netlist. $|E_C|$ is related to the number of camouflaging cells inserted into the circuits, the locations of inserted cells, and the number of possible functionalities of different camouflaging cells. Traditional strategies usually target at increasing the possible functionalities for each cell, which suffers from large overhead and provides insufficient protection. We observe that *the overhead of one camouflaging cell is mainly determined by its actual functionality in the circuit.* In this section, we will propose two different camouflaging cell designs, termed as XOR-type cells and stuck-at-fault-type (STF-type) cells, which incur negligible overhead for some specific functionality.

A. XOR-*Type Cell Camouflaging Strategy*

The XOR-type camouflaging strategy leverages the dummy contact technique. For example, as shown in Fig. 4(a), for a BUF cell, we modify the shape of the polysilicon to create an extra overlap between polysilicon and metal layer. Then, we can configure the functionality of the camouflaging cell by determining whether the five contacts are real or dummy. When contact A, B, D, and E are real while contact C is dummy, the cell functions as a BUF. If contact C is real while the rest of the contacts are dummy, the cell functions as an INV. The similar strategy can be applied to other cells, including AND, OR, and so on.

TABLE I
Overhead Characterization of XOR-Type Camouflaged Cell

Cell	BUF		AND2		OR2		AND3	
Func.	BUF	INV	AND2	NAND2	OR2	NOR2	AND3	NAND3
Timing	1.0×	2.0×	1.0×	1.5×	1.0×	1.9×	1.0×	1.8×
Area	1.0×	1.5×	1.0×	1.3×	1.0×	1.3×	1.0×	1.3×
Power	1.0×	1.5×	1.0×	0.9×	1.0×	1.1×	1.0×	1.0×

TABLE II
Overhead Characterization of STF-Type Camouflaged Cell

Cell	OR2		NAND2		AND3		
Func.	OR2	BUF	NAND2	INV	AND3	AND2	BUF
Timing	1.0×	1.4×	1.0×	1.6×	1.0×	1.3×	1.8×
Area	1.0×	1.3×	1.0×	1.5×	1.0×	1.3×	1.7×
Power	1.0×	1.2×	1.0×	1.5×	1.0×	1.1×	1.3×

To evaluate the overhead of the XOR-type camouflaged cells, standard cells from NanGate 45-nm Open Cell Library [34] are modified according to the strategy and scaled to 16-nm technology. Then Calibre xRC [35] is used to extract parasitic information of the cell layouts. We use SPICE simulation to characterize different types of gates, which are based on 16-nm PTM model [36]. As we can see from Table I, when the cell functions as a BUF, the overhead induced by the layout modification is negligible compared with original standard cells.

B. *STF-Type Cell Camouflaging Strategy*

The STF-type camouflaging strategy leverages the doping-based camouflaging technique. The camouflaging cell generated with the STF-type strategy has exactly the same metal and polysilicon layer compared with the existing standard cells in the library. The only difference comes from the type and the shape of the lightly doped-drain, which makes it very difficult to distinguish a regular MOS transistor with the Always-on and Always-off MOS transistor. The STF-type camouflaging strategy fully leverages this flexibility to create camouflaging cells with different functionalities.

For example, as shown in Fig. 4(b), for a NAND2 cell, if we change the doping scheme following [5], we can create Always-on nMOS transistor and Always-off pMOS transistor associated with A. This is equivalent to creating a stuck-at-1 fault at input A and the functionality of the NAND cell becomes an INV for input B. Similar strategy can be applied to all the other cells in the original library. To characterize the STF-type cells, we use the same method as described for the XOR-type camouflaging strategy and the overhead results is listed in Table II.

C. *Discussion*

As described above, both XOR-type and STF-type camouflaging cells proposed above incur negligible overhead for some specific functionalities. It should be noted that they also have different characteristics. For the XOR-type camouflaging cell, when the attacker misinterprets the type of the

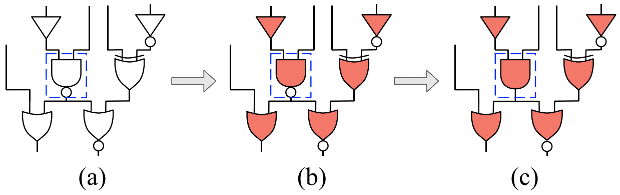


Fig. 5. Two-step IC camouflaging with XOR-type and STF-type cells: (a) original circuit netlist; (b) change all standard cells into camouflaging cells that appear to be same and have same functionalities; and (c) random select cells and replace them with cells that appear to be different but work with the same functionalities.

contact, the probability of logic error at the output of the cell is always 1. For the STF-type cell, a misinterpretation of the doping scheme may not always lead to incorrect logic value at the output of the gate. For example, consider an AND gate with $\widetilde{n}$ inputs, denoted as $i_1, i_2, \ldots, i_{\widetilde{n}}$, and first $\widetilde{n}'$ inputs are dummy. Then, the probability of logic error at the output of the cell can be calculated as

$$P_e = \Pr_{i \sim I}\left[\left(\bigcup_{k\in[\widetilde{n}']} i_k = 0\right) \cap \left(\bigcap_{k\in[\widetilde{n}]\setminus[\widetilde{n}']} i_k = 1\right)\right]$$

where $[\widetilde{n}] = \{1, 2, \ldots, \widetilde{n}\}$.

Meanwhile, for the STF-type camouflaging cell, because it has one or more input pins that do not impact the output of the cell, it enables to create dummy connections between different nodes to hide the circuit structure.

To leverage the XOR-type and STF-type cells to camouflage the original circuits, we propose a two-step strategy. In the first step, we replace all the standard cells with the camouflaging cells, e.g., NAND cell to an STF-type NAND cell in Fig. 5(b). For these camouflaging cells, they are set to work as the cells that they appear to be, e.g., an STF-type NAND cell works as a real NAND gate, and therefore, the replacement incurs negligible overhead based on our characterization results above. Then, in the second step, we randomly choose a small subset of gates in the netlist, and replace them with new camouflaging cells that appear differently but indeed work with the same functionality as the original cells, e.g., a NAND cell is replaced by an XOR-type AND cell in Fig. 5(c). The overall introduced overhead is negligible since only a small subset of gates are modified in the second step. As the attackers cannot determine the functionality for each cell in the camouflaged circuits, the total number of possible functionalities can be extremely large.

The effectiveness of the proposed camouflaging cell generation strategy is verified in Section VII. However, as evaluating $|C|$ or $|E_C|$ accurately is computationally intractable, it is hard to provide a provably secure guarantee. Meanwhile, the effectiveness of the proposed method is limited by the circuit size since we only replace original cells in the circuits.

V. AND-Tree Camouflaging Strategy

In this section, we target at increasing θ as in (1). In [25], the AND-tree structure is noticed to achieve good resilience to SAT-based de-camouflaging attack when the input pins are camouflaged as shown in Fig. 6. In this section, we provide formal analysis for the AND-tree structure and further identify two important characteristics of the AND-tree structure, denoted as input bias and tree decomposability, to characterize its effectiveness in general circuits.

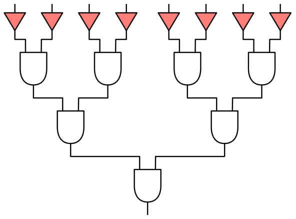


Fig. 6. Example of a camouflaged AND-tree structure.

A. Security Analysis of the AND-Tree Structure

Consider the AND-tree structure with n input pins shown in Fig. 6, where all the input pins are camouflaged with the XOR-type camouflaging BUF cell. Recall from Section III that $I \subseteq \{0, 1\}^n$ and $Y \subseteq G^n$ represent all the possible combinations of functionalities for the camouflaging cells. For any $i \in I$ and $y \in Y$, the output of the AND-tree structure can be expressed as

$$c_y(i) = g_1(i_1) \wedge g_2(i_2) \wedge \ldots \wedge g_n(i_n)$$

where i_k denotes the kth entry of input i and $g_k(\cdot)$ denotes functionality of the kth camouflaging cell. $g_k(i_k) = i_k$ if the kth cell functions as a BUF, while $g_k(i_k) = \overline{i_k}$ if the kth cell functions as an INV.

Let $y^* \in Y$ denotes the correct configuration for all the camouflaging cells. Then, depending on y, there are 2^n different circuit functionalities, i.e., $|C| = |E_C| = 2^n$. For any $y \in Y$, there exists exactly one input $i \in I$ such that $c_y(i) = 1$, denoted as i^y. Therefore, we have $\Pr_{i\sim I}[c_y(i) = 1] = \Pr_{i\sim I}[i = i^y]$. Now, we have the following lemma for the camouflaged AND-tree structure.

Lemma 1: For an n-bit AND-tree structure with all tree inputs camouflaged with XOR-type camouflaging BUF cells, if the logic values for tree inputs follow identical independent Bernoulli distribution with probability of 0.5, then, we have $\theta = 2^{n-1}$.

To prove Lemma 1, we will first demonstrate that when the logic values for all the tree inputs follow identical independent Bernoulli distribution with probability of 0.5, for any $y \neq y^*$, the error rate of the indicator function e_{c_y} is $1/2^{n-1}$ compared with $e_{c_{y^*}}$. Meanwhile, we will show that $\mathrm{DIS}(E_\epsilon) = I \times O$. Then, based on the definition of θ in (2), we will prove that $\theta = 2^{n-1}$.

Proof: For any $y \neq y^*$, c_y is different compared with c_{y^*} for exactly two input vectors, i.e., i^y and i^{y^*}. For i^y, because $c_y(i^y) = 1$ while $c_{y^*}(i^y) = 0$, we have $e_{c_y}(i^y, 1) \neq e_{c_{y^*}}(i^y, 1)$ and $e_{c_y}(i^y, 0) \neq e_{c_{y^*}}(i^y, 0)$. Therefore, e_{c_y} has different outputs compared with $e_{c_{y^*}}$ at exactly four points, i.e., $\{(i^y, 1), (i^y, 0), (i^{y^*}, 1), (i^{y^*}, 0)\}$. This indicates $\forall e_{c_y} \in E_C$ with $y \neq y^*$

$$\begin{aligned}
&\mathrm{er}_{(i,o)\sim I\times O}\left(e_{c_y}, e_{c_{y^*}}\right)\\
&= \Pr_{(i,o)\sim I\times O}\left[e_{c_y}(i,o) \neq e_{c_{y^*}}(i,o)\right]\\
&= \Pr_{(i,o)\sim I\times O}\left[(i,o) \in \left\{(i^y,0),(i^y,1),\left(i^{y^*},0\right),\left(i^{y^*},1\right)\right\}\right]\\
&= \Pr_{i\sim I}\left[i = i^y \vee i = i^{y^*}\right] = \Pr_{i\sim I}\left[i = i^y\right] + \Pr_{i\sim I}\left[i = i^{y^*}\right]. \qquad (3)
\end{aligned}$$

Note when $y = y^*$, $\text{er}_{(i,o)\sim I\times O}(e_{c_y}, e_{c_{y^*}}) = 0$. Therefore,

$$\begin{aligned} E_\epsilon &= \left\{e_{c_y} \in E_C : \text{er}_{(i,o)\sim I\times O}\left(e_{c_y}, e_{c_{y^*}}\right) \le \epsilon\right\} \\ &= \left\{e_{c_y} \in E_C : \Pr_{i\sim I}\left[i = i^y\right] + \Pr_{i\sim I}\left[i = i^{y^*}\right] \le \epsilon\right\} \cup \left\{e_{c_{y^*}}\right\} \\ &= \left\{e_{c_y} \in E_C : \Pr_{i\sim I}\left[i = i^y\right] \le \epsilon - \Pr_{i\sim I}\left[i = i^{y^*}\right]\right\} \cup \left\{e_{c_{y^*}}\right\}. \end{aligned} \tag{4}$$

Because $\forall e_{c_y} \in E_\epsilon$, where $y \ne y^*$, is different from $e_{c_{y^*}}$ at exactly four points, we have

$$\begin{aligned} \text{DIS}(E_\epsilon) = \Bigg\{ &(i,o) \in I \times O : \Pr_{i\sim I}\left[i = i^y\right] \le \epsilon - \Pr_{i\sim I}\left[i = i^{y^*}\right] \\ &o \in \{0,1\}\Bigg\} \cup \left\{\left(i^{y^*}, 1\right), \left(i^{y^*}, 0\right)\right\}. \end{aligned} \tag{5}$$

For tree inputs, if the logic values follow independent Bernoulli distribution with probability of 0.5, $\forall i^y \in I$, we have

$$\Pr_{(i,o)\sim I\times O}\left[(i,o) \in \left\{\left(i^y, 0\right), \left(i^y, 1\right)\right\}\right] = \Pr_{i\sim I}\left[i = i^y\right] = \frac{1}{2^n}$$

and $\forall e_{c_y} \in E_C$ with $y \ne y^*$

$$\text{er}_{(i,o)\sim I\times O}\left(e_{c_y}, e_{c_{y^*}}\right) = \frac{1}{2^{n-1}}.$$

Therefore, by setting $\epsilon = 1/2^{n-1}$, we have $E_\epsilon = E_C$ and $\text{DIS}(E_\epsilon) = I \times O$. According to the definition of θ

$$\theta = \frac{\Pr_{(i,o)\sim I\times O}[(i,o) \in \text{DIS}(E_\epsilon)]}{\epsilon} = 2^{n-1}.$$

Hence, proved. ■

Base on Lemma 1, we now have the following theorem concerning the security of a camouflaged AND-tree structure.

Theorem 1: For an n-input AND-tree structure with all tree inputs camouflaged with XOR-type camouflaging BUF cells, if the logic values for tree inputs follow identical independent Bernoulli distribution with probability of 0.5, then

$$m\left(e_{c_{y^*}}, E_C\right) = \mathcal{O}\left(2^n\right).$$

Proof: Based on Lemma 1, we have $\theta = 2^{n-1}$ for an n-input AND-tree. Meanwhile, because $|E_C| = 2^n$, we have $d \ge \log_2 |E_C|/n = 1$. Therefore, $m(e_{c_{y^*}}, E_C) = \mathcal{O}(2^n)$. Hence, proved. ■

From Theorem 1, under the assumption that the logic values for tree inputs follow identical independent Bernoulli distribution with probability of 0.5, we can formally prove the security of an n-input AND-tree by showing that the de-camouflaging complexity of an SAT-based attack scales exponentially with the increase of tree input size.

B. AND-Tree Structure in General Circuits

According to the analysis above, a stand-alone AND-tree structure can lead to an exponential increase of de-camouflaging complexity. However, this may not be true for an AND-tree structure in general circuits due to the following reasons.

1) As shown in Fig. 7(a), different tree inputs may not be independent since their fanin cones can overlap. Meanwhile, the signal probability for each input can also deviate from 0.5.

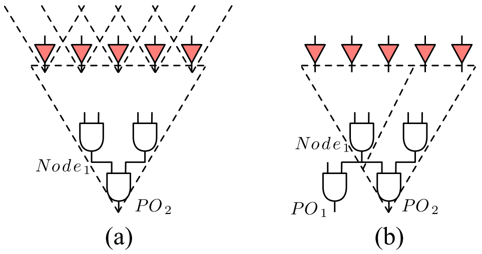


Fig. 7. Two situations that can impact the security of AND-tree structure: (a) overlapped fanin cone for input pins leads to correlation and (b) extra path to primary outputs from internal node makes it possible to decompose the tree.

2) There are usually more than one primary outputs and more than one paths from some internal nodes of an AND-tree to the primary outputs. For example in Fig. 7(b), Node_1 can bypass the root of the tree PO_2 and get observed at the primary output PO_1. This can also reduce the de-camouflaging complexity of the AND-tree structure.

The two factors are defined as input bias and tree decomposability.

1) Input Bias Evaluation: Input bias is proposed to characterize the distance between the actual joint signal distribution for input pins and the ideal independent Bernoulli distribution. It mainly impact E_ϵ and $\text{DIS}(E_\epsilon)$. According to (5), to decide $\text{DIS}(E_\epsilon)$, we need to calculate the probability of each input vector, which, however, is intractable for large circuits. To capture the impact of input bias, we instead consider the following approximate approach.

According to (3), $\forall y \ne y^*$, we have

$$\begin{aligned} \text{er}_{(i,o)\sim I\times O}\left(e_{c_y}, e_{c_{y^*}}\right) &= \Pr_{i\sim I}\left[i = i^y\right] + \Pr_{i\sim I}\left[i = i^{y^*}\right] \\ &\ge \Pr_{i\sim I}\left[i = i^{y^*}\right]. \end{aligned}$$

To get a nonempty E_ϵ, we must choose $\epsilon \ge \Pr_{i\sim I}[i = i^{y^*}]$. Because we always have $\Pr_{(i,o)\sim I\times O}[(i,o) \in \text{DIS}(E_\epsilon)] \le 1$, then

$$\theta = \frac{\Pr_{(i,o)\sim I\times O}[(i,o) \in \text{DIS}(E_\epsilon)]}{\epsilon} \le \frac{1}{\Pr_{i\sim I}\left[i = i^{y^*}\right]}.$$

Therefore, to evaluate the impact of input bias, we can first calculate $\Pr_{i\sim I}[i = i^{y^*}]$ to get the upper bound of θ. If the upper bound is smaller than the predefined requirement, then, we conclude the AND-tree in the circuit is not enough to guarantee the security.

To evaluate $\Pr_{i\sim I}[i = i^{y^*}]$, we consider the procedure as shown in Fig. 8. We first extract the fanin cone for all the tree input pins as in Fig. 8(b). Then, as in Fig. 8(c), we form the circuit that connects each tree input pin to its desired logic value. The output of the formed circuit equals to 1 if and only if the logic values for all the tree inputs equal to y^*. Therefore, if we force the output equals to 1 and solve the value for the circuit primary inputs, we can get $\Pr_{i\sim I}[i = i^{y^*}]$. We formulate the problem as an SAT problem and show the pseudocode in Algorithm 1. FORMSATPROB grabs the fanin

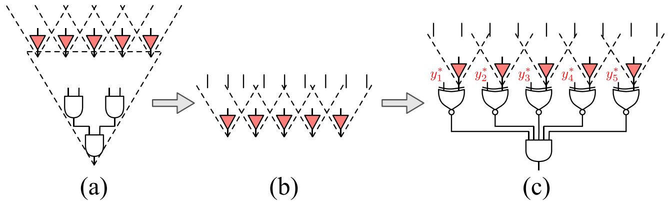


Fig. 8. Exact calculation of $\Pr_{i \sim I}[f_{y^*}(i) = 1]$: (a) original tree structure; (b) fanin cone extracted for all the tree input pins; and (c) form the circuit that connects all tree input pins to the desired logic values, i.e., $\{y_1^*, y_2^*, y_3^*, y_4^*, y_5^*\}$. By forcing the output of the formed circuit to 1, we can solve the logic values for the circuit primary inputs iteratively as in Algorithm 1.

Algorithm 1 Algorithm of Calculating $\Pr_{i \sim I}[i = i^{y^*}]$

```
1: $F \leftarrow \text{FORMSATPROB}(G, i, y, y^*)$;
2: $Cnt \leftarrow 0$;
3: **while** $F$ is satisfiable **do**
4:     $i_t \leftarrow \text{SATSOLVE}(F)$;
5:     $Cnt \leftarrow Cnt + 1$;
6:     **if** $\frac{Cnt}{2^{|i|}} \geq T_h$ **then**
7:         **Return** $\frac{Cnt}{2^{|i|}}$;
8:     $F \leftarrow F \wedge (i \neq i_t)$;
9: **Return** $\frac{Cnt}{2^{|i|}}$;
```

cone for the tree input pins and forms the SAT equation as in Fig. 8(c) (line 1). *Cnt* is used to count the total number of input vectors that satisfy the SAT equation, which in turn can be used to calculate $\Pr_{i \sim I}[i = i^{y^*}]$. We initialize *Cnt* to 0 and iteratively solve the SAT problem to search for the input vectors until the SAT problem is not satisfiable or *Cnt* is larger than a pre-defined threshold.

Given $\Pr_{i \sim I}[i = i^{y^*}]$, we can determine the upper bound of the de-camouflaging complexity for the tree structure. When the upper bound is large enough, to further evaluate the tree structure, we adopt the normalized Kullback–Leibler (KL) divergence to calculate the distance between the actual distribution for logic values of tree input pins compared with the ideal distribution. Normalized KL divergence for two discrete probability distribution, P and Q, is calculated as

$$\text{KL}(P|Q) = \frac{1}{n} \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{6}$$

In our case, since Q is uniform, $\text{KL}(P|Q) = (n - H_p)/n$, where H_p is the total entropy of distribution P. Note that the larger the KL divergence is, the closer $\text{KL}(P|Q)$ approaches to 1 and the worse the security of the AND-tree is.

2) Tree Decomposability Characterization: To characterize the impact of multiple paths to primary outputs, we propose the concept on tree decomposability.

Definition 1 (Decomposable Tree): An AND-tree structure is decomposable if: 1) there exists a path from the internal node of the tree to the primary output that can bypass the root of the tree and 2) change of the logic value of the internal node can be observed at the primary output through the path.

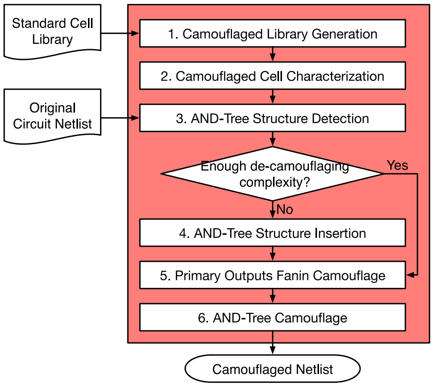

Standard Cell Library
Original Circuit Netlist
1. Camouflaged Library Generation
2. Camouflaged Cell Characterization
3. AND-Tree Structure Detection
Enough de-camouflaging complexity?
Yes
No
4. AND-Tree Structure Insertion
5. Primary Outputs Fanin Camouflage
6. AND-Tree Camouflage
Camouflaged Netlist

Fig. 9. Proposed IC camouflaging flow.

Algorithm 2 Determine Whether an AND-Tree Is Decomposable

```
// $G$ is the original circuit and $G_t$ is the AND-tree
// $r_t$ is the root of the tree
$U \leftarrow \text{TOPOLOGICALSORT}(r, G_t)$;
**for** $u \in U$ **do**
    **if** $u$.fanout $> 1$ **then**
        $\{p_1, \ldots, p_m\} \leftarrow \text{DFS}(u, G)$;
        **if** $\exists i$, s.t. $r \notin p_i$ **then**;
            **return** True;
**return** False;
```

Both of the conditions are important. For example, the AND-tree structure in Fig. 7(b) is decomposable because the internal node Node_1 can bypass the root of the tree PO_2 and get observed at the output PO_1. Tree decomposability is undesired because it enables the attacker to first de-camouflage the subtree structure rooted at Node_1, and then de-camouflage the remaining part of the tree. The number of input vectors needed to de-camouflage the decomposable AND-tree is thus limited by the sum of the input vectors needed to de-camouflage the two subtrees. Because the sizes of the two subtrees are much smaller than the original AND-tree, the de-camouflaging complexity becomes much smaller.

To determine whether an AND-tree is decomposable, we propose the algorithm shown in Algorithm 2. We traverse the tree structure in a reverse topological order starting from the root. For each internal node u of the tree, if it has more than 1 successors, then, we do a depth-first search starting from u and keep a record of all the paths from u to the primary outputs. If the tree root exists in each path, then, the tree is nondecomposable.

VI. Provably Secure IC Camouflaging

In this section, we will leverage the proposed camouflaging cell generation method and the AND-tree structure to provide provably secure camouflaging strategy. The overall flow of the proposed IC camouflaging framework is illustrated in Fig. 9. The first step is the camouflaging cell library generation with the proposed techniques described in Section IV with an accurate characterization of timing, power, and area overhead for each cell. In the third step, existing AND-tree structure is detected for the original netlist. If the predefined de-camouflaging complexity is not satisfied, new AND-tree structure needs to be inserted as in the fourth step. In the fifth

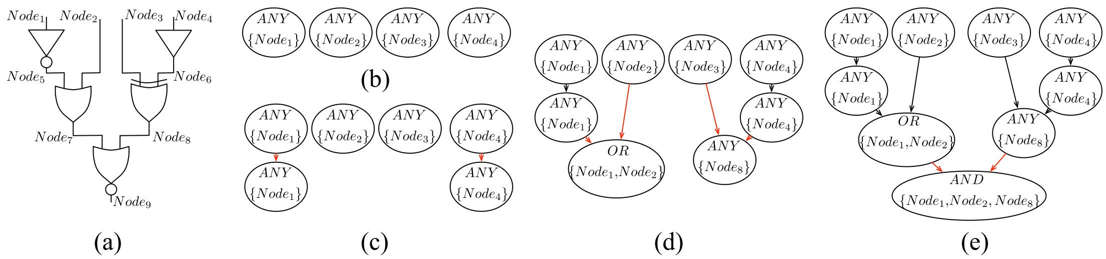


Fig. 10. Detect tree structure in topological order: (a) circuit netlist and (b)–(e) start from primary inputs to calculate tree structure in topological order.

Algorithm 3 Algorithm of AND-Tree Detection

```
1:  Let {AND, ANY, OR} denote a set of tree types.
2:  $U \leftarrow$ TopologicalSort($G$);
3:  for $u \in U$ do
4:      if $u$ is primary input then
5:          $u$.treetype ← ANY;
6:          $u$.treeinput ← $u$;
7:      else
8:          if $u$.gatetype ∈ {BUF, INV} then
9:              $u$.treetype ← $u$.fanin.treetype;
10:             $u$.treeinput ← $u$.fanin.treeinput;
11:         else if $u$.gatetype ∈ {AND, NAND, OR, NOR} then
12:             if $u$.gatetype ∈ {AND, NAND} then
13:                 $u$.treetype ← AND;
14:             else if $u$.gatetype ∈ {OR, NOR} then
15:                 $u$.treetype ← OR;
16:             for $v \in u$.fanin do
17:                 if $v$.treetype = $u$.treetype and Size($v$.fanout) = 1 then
18:                     $u$.treeinput.Add($v$.treeinput);
19:                 else
20:                     $u$.treeinput.Add($v$);
21:         else
22:             $u$.treetype ← ANY;
23:             $u$.treeinput ← $u$;
24:         if $u$.gatetype ∈ {INV, NOR, NAND} then
25:             $u$.treetype ← Invert($u$.treetype);
26: return $U$.
```

step, we leverage the inserted AND-tree structure to protect all the primary outputs. We further camouflage the AND-tree structure to enhance the resilience against tree removal attack in the sixth step.

A. AND-Tree Detection in Original Netlist

AND-tree represents a set of circuit structures. We denote all the circuit structures that generate 1 as output for only one input vector as AND-tree and those that generate 0 as output for only one input vector as OR-tree. The pseudocode of the algorithm to detect the tree structure is shown in Algorithm 3. We start from the primary inputs of the circuit and sort all the circuit nodes in a topological order (line 2). For each node, we keep a record of the tree rooted at this node by recording the input pins of the tree. For primary inputs, the type of tree rooted at the node can be treated as either AND-type or OR-type (lines 4–6). For the internal nodes, to determine the input pins of the tree structure, we consider the gate type of the node and its predecessors in the circuit graph. Depending on the type of the gate, there are different possibilities (lines 7–31). First, if the gate is INV or BUF, the node will have the same tree as its input (lines 8–10). For INV, function Invert() is called to change the tree type from AND-type to OR-type or vice versa. Second, if the gate is AND or OR, the tree type rooted at the node can first be determined (lines 12–16). Then, to determine the input pins, there are two situations depending on the tree types of the predecessors and the node. When the tree types are the same, larger tree structure can be formed (lines 18 and 19). Otherwise, only the predecessors can be added to the tree (lines 20–22).

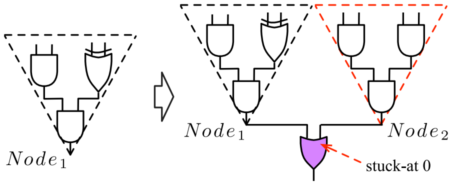


Fig. 11. Insert AND-type tree structure to the circuit (Node_2 is stuck-at-0 to guarantee the functional correctness).

Example 3: Consider the circuit shown in Fig. 10(a). As in Fig. 10(b), for primary input Node_1, the tree type is ANY and the input of the tree is $\{\text{Node}_1\}$. For internal node Node_5, since it is connected with Node_1 through a BUF, the tree type for Node_5 is also ANY and the inputs is also $\{\text{Node}_1\}$. For Node_8, since it is connected with an XOR gate, the tree type becomes ANY and the inputs to the tree is the node itself, i.e., $\{\text{Node}_8\}$. Consider Node_7, since it is connected with an OR gate, the tree type has to be OR. For the two inputs, i.e., Node_5 and Node_2, since the tree types for both nodes are ANY, they can be combined to form a large tree structure. Therefore, the input pins for the tree rooted at Node_7 becomes $\{\text{Node}_1, \text{Node}_2\}$. Similarly, we can determine the tree type and tree inputs for Node_9. Because Node_9 is connected with a NOR gate, the tree type becomes AND.

After the calculation of the tree structure rooted at each node, we can examine whether the predefined de-camouflaging complexity is satisfied as described in Section V-B. If the requirement is not satisfied, new tree structures need to be inserted.

B. Stochastic Greedy AND-Tree Insertion

When inserting the AND-tree into the circuit, we need to guarantee the functionality of the original circuit is not changed. We leverage the STF-type camouflaging cells as described in Section IV. Consider an example circuit as shown in Fig. 11. To insert an AND-tree structure at Node_1, we first insert an OR gate to Node_1 with the other input Node_2

Algorithm 4 Algorithm of Stochastic Greedy AND-Tree Insertion

```
$U_{PO} \leftarrow POs$;
$U \leftarrow$ TopologicalSort($G$);
RemoveCriticalNode($U$);
while $U_{PO} \neq \emptyset$ do
    for $u \in U$ do
        $u.\text{score} \leftarrow$ ComputeIS($G$);
    $U_{IS} \leftarrow$ FindTopK($U$);
    $u_c \leftarrow$ RandomSelectCand($U_{IS}$);
    $G \leftarrow$ AndTreeInsert($u_c, G$);
    $U_{PO} \leftarrow$ RemoveCoveredPO($U_{PO}, u_c$);
return $U$;
```

as dummy pin. Then, an AND-tree structure is created with Node$_2$ being the root. The input pins of the AND-tree structure are connected to the primary inputs and camouflaged with XOR-type cells.

To detect the stuck-at 0 fault at Node$_2$, we again follow the same analysis as in Section V. The logic value of Node$_2$, which is 0 in reality, can be expressed as

$$c_y(i) = g_{n+1}(g_1(i_1) \wedge g_2(i_2) \wedge \ldots \wedge g_n(i_n)).$$

Note that $g_{n+1}(i) = 0$ indicates there is a stuck-at-0 fault at Node$_2$, and $g_{n+1}(i) = i$ otherwise. Among all the possible configurations, there are 2^n correct configurations with g_{n+1} interpreted as stuck-at-0 and 2^n incorrect configurations with $g_{n+1}(i) = i$. For any false configuration y, c_y outputs 1 for exactly one input vector, denoted as i^y, and thus, is different from c_{y*} for exactly one input vector. For the corresponding indicator function, e_{c_y} is different from $e_{c_{y*}}$ at exactly two points, i.e., $\{(i^y, 1), (i^y, 0)\}$. Therefore, $\forall y$ with $c_y \neq c_{y*}$, we have

$$\begin{aligned} &\text{er}_{(i,o)\sim I\times O}\Big(e_{c_y}, e_{c_{y*}}\Big) \\ &= \Pr_{(i,o)\sim I\times O}\Big[e_{c_y}(i,o) \neq e_{c_{y*}}(i,o)\Big] \\ &= \Pr_{(i,o)\sim I\times O}\Big[(i,o) \in \big\{(i^y, 0), (i^y, 1)\big\}\Big] \\ &= \Pr_{i\sim I}\big[i = i^y\big]. \end{aligned} \tag{7}$$

Since we connect the input pins of the inserted tree structure with circuit primary inputs, we can assume no input bias for tree inputs, which indicates

$$\Pr_{i\sim I}\big[i = i^y\big] = \frac{1}{2^n}.$$

Similar to proof in Section V-A, if we set $\epsilon = 1/2^n$, then, we have $E_\epsilon = E_C$ and $\text{DIS}(E_\epsilon) = I \times O$. In this case, $\theta = 2^n$. Therefore, $m(e_{c_{y*}}, E_C) = \mathcal{O}(2^n)$.

Therefore, the required number of input vectors to decamouflage the circuit increases exponentially to the size of the inserted AND-tree structure. The insertion of OR-tree follows the same procedure except that we need to use an AND gate with stuck-at-1 fault at the dummy input, which is the root of the OR-tree structure.

To insert the tree structure into the circuit, we propose a stochastic greedy tree insertion algorithm as shown in Algorithm 4, which tries to minimize the performance overhead and guarantee the functionalities for all primary outputs are protected. We first add all the primary outputs to protect in a set U_{PO}. Then, we traverse the circuit graph in a topological order and calculate an insertion score (IS) for each internal nodes. IS is defined to consider the node's switching probability SA, observe probability P_{ob} and the number of primary outputs N_O that have not been camouflaged in its fanout cone, which is calculated as

$$\text{IS} = \frac{\alpha \times \text{SA} - \beta \times P_{\text{ob}}}{N_O}. \tag{8}$$

where α and β are coefficients defined to balance SA and P_{ob}. Note that we get rid of the circuit nodes along critical paths in the process to minimize the impact on performance. We find k nodes with the smallest scores from U and randomly select one node as the candidate for tree insertion. All the primary outputs in the fanout cone of the candidate node are removed from U_{PO} and the procedure is continued until all the primary outputs are protected.

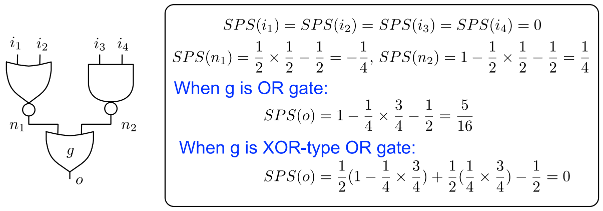

i_1
i_2
i_3
i_4
n_1
n_2
g
o
$SPS(i_1) = SPS(i_2) = SPS(i_3) = SPS(i_4) = 0$
$SPS(n_1) = \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} = -\frac{1}{4}$, $SPS(n_2) = 1 - \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} = \frac{1}{4}$
When g is OR gate:
$SPS(o) = 1 - \frac{1}{4} \times \frac{3}{4} - \frac{1}{2} = \frac{5}{16}$
When g is XOR-type OR gate:
$SPS(o) = \frac{1}{2}(1 - \frac{1}{4} \times \frac{3}{4}) + \frac{1}{2}(\frac{1}{4} \times \frac{3}{4}) - \frac{1}{2} = 0$

Fig. 12. SPS-based functional attack: when g is OR gate, $SPS(o) = \frac{5}{16}$ and when g is XOR-type camouflaging cell, whether g works as an OR gate or a NOR gate are equally possible for the attacker [27], and thus $SPS(o) = 0$.

C. AND-Tree Camouflaging Against Removal Attack

Though AND-tree structure provides good resilience against SAT-based attack, because it has obvious structural and functional footprint, it is possible for the attacker to identify and remove it. Yasin *et al.* [27] proposed to identify the AND-tree by calculating the SPS for each circuit node. SPS of a signal s is defined as $\Pr[s = 1] - 0.5$. In Fig. 12, we use an example to illustrate the attack process. Starting from primary inputs, the attacker traverses the circuit netlist topologically. For a standard cell, the signal probability can be easily calculated while for a camouflaging cell, because its actual functionality in the circuit is unknown the authors calculate the signal probability as the average value for different functionalities. For a signal with large uncertainty, its SPS approaches to 0. For the root of an AND-tree, its SPS approaches to -0.5 exponentially with respect to the size of the AND-tree. This makes it possible for the attacker to identify the inserted tree structure by SPS. Besides the SPS-based attack, because a nondecomposable AND-tree is an isolated structure that does not have many connections with the original circuit, the attackers can also leverage this structural footprint to detect the inserted AND-tree structure. Therefore, we propose to camouflage the structure both functionally and structurally.

We use the example in Fig. 13 to illustrate our AND-tree camouflaging strategy. Consider an 8-input AND-tree in Fig. 13(a). We first replace the standard cells in the AND-tree with camouflaging cells that look the same and share the same

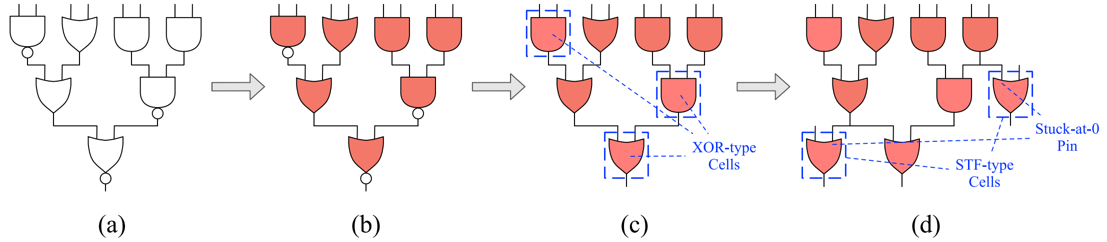


Fig. 13. AND-tree protection: (a) original AND-tree structure; (b) and (c) functional camouflaging to reduce SPS for the root node; and (d) structural camouflaging to add dummy connections from AND-tree internal nodes.

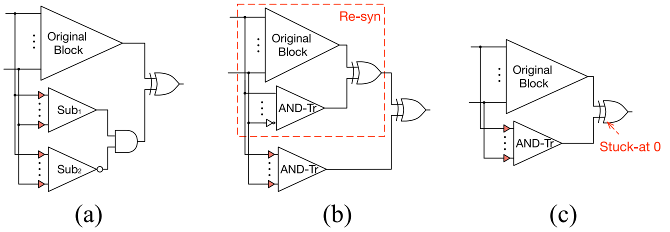


Fig. 14. Comparison on AND-tree insertion strategy: (a) Anti-SAT [22]; (b) CamoPerturb [21]; and (c) our insertion strategy.

functionality, as in Fig. 13(b). Then, we replace the NAND, NOR, and INV cells with XOR-type camouflaging cells that look different but share the same functionality, as in Fig. 13(c). Because the attacker cannot determine whether the output of each cell in the AND-tree is negated or not, e.g., whether an AND cell works as an AND or a NAND cell in the circuit, the SPS for each node in the AND-tree is always kept as 0 according to [27].

To prevent removal attack based on the structural information, we leverage the STF-type camouflaging cell to connect the internal nodes of the AND-tree to other gates as in Fig. 13(d). For an $\tilde{n}$-input AND-tree, there are in total $\tilde{n}-1$ gates in the tree following the structure in Fig. 13(a). To ensure the size of the largest AND-tree detected by the attacker to be less than $\tilde{n}'$, we can always insert $O((\tilde{n}-1)/(\tilde{n}'-1))$ STF-type camouflaging cells to create dummy connections to the internal nodes as in Fig. 13(d). Meanwhile, to prevent the attackers from identifying the inputs to the AND-tree, we can insert extra XOR-type BUF cells to other primary inputs. Note that the inserted AND-tree is nondecomposable from the defense perspective since the logic value of AND-tree internal nodes cannot be sensitized through the dummy connections. However, for the attackers, because he cannot determine the connections are dummy based on the structural attack following Algorithm 3, the largest nondecomposable tree that can be detected by the structural attack is reduced significantly. Therefore, the resilience to SAT-based attack is not impacted, while the vulnerability to removal attacks is mitigated significantly.

D. Comparison Between State-of-the-Art Techniques

The idea to leverage AND-tree structures has also been explored by Anti-SAT [22] and CamoPerturb [21]. Now, we compare the three strategies in terms of security, overhead, and impact on the original circuit, i.e., whether resynthesis is required.

TABLE III
COMPARISON WITH CAMOPERTURB [21] AND ANTI-SAT [22]

Strategy	De-cam Complexity			Overhead	Resyn
	Worst	Avg	Best		
Ours	$2^{\tilde{n}}$	$2^{\tilde{n}}$	$2^{\tilde{n}}$	$\tilde{n}$-bit tree	No
CamoPerturb	1	$2^{\tilde{n}-1}$	$2^{\tilde{n}}$	$\tilde{n}$-bit tree+ Resyn	Yes
Anti-SAT	$2^{\tilde{n}}$	$2^{\tilde{n}}$	$2^{\tilde{n}}$	2 $\tilde{n}$-bit tree	No

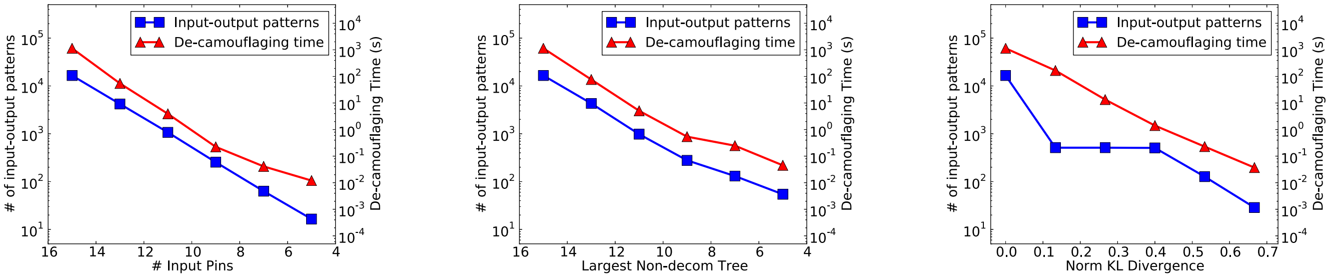
Assume an AND-tree with $\tilde{n}$-bit inputs is to be inserted into the circuit. As shown in Fig. 14(b), for the Anti-SAT strategy, two subtrees of the same size, denoted as Sub_1 and Sub_2, are inserted into the circuit. An INV is inserted at the output of Sub_2. XOR-type BUF cells are inserted at the input pins of the two AND-trees. For the circuit to function correctly, the XOR-type BUF cells of same input signals in the Sub_1 and Sub_2 need to have the same functionality. To de-camouflage the circuit, the attacker always needs to query $2^{\tilde{n}}$ input vectors [22].

For CamoPerturb, to insert an $\tilde{n}$-bit AND-tree, a specific input vector i^* is first selected. Then, original circuit is resynthesized by flipping the output value corresponding to i^*. An AND-tree is then inserted to correct the flipped output just for i^*. Based on [21], to de-camouflage the circuit, all input vectors are discriminating inputs and for each $i \neq i^*$, at most one false functionality can be pruned. However, i^* can rule out all the false functionalities. Therefore, on average, $2^{\tilde{n}-1}$ input vectors need to be measured. While the best case and worst case de-camouflaging complexity become $2^{\tilde{n}}$ and 1. In our strategy, the de-camouflaging complexity is always $2^{\tilde{n}}$ because, for any input vector, exactly one false functionality can be ruled out.

The three strategies mainly introduce area and power overhead while the impact on timing can be negligible by avoiding any modification of circuit critical paths. Compared with our method, Anti-SAT suffers from almost double area and power overhead because one AND-tree and one NAND-tree of the same size is inserted. For CamoPerturb, besides the overhead introduced by the inserted AND-tree, extra overhead can be introduced in the process of resynthesis. We summarize the comparison in Table III. As we can see, our strategy for AND-tree insertion provides the best security guarantee without the requirement for resynthesis.

VII. EXPERIMENTAL RESULTS

In this section, we report on our experiments to demonstrate the effectiveness of the proposed IC camouflaging strategy. The camouflaging algorithm is implemented in C++. The



(a)

(b)

(c)

Fig. 15. Effectiveness of tree structure and impact of tree decomposability and input bias: (a) de-camouflaging complexity and time for ideal AND-tree structure; (b) change of de-camouflaging complexity and time with the size of the largest nondecomposable tree; and (c) change of de-camouflaging complexity and time with the input bias.

TABLE IV
VERIFY THE PROPOSED CAMOUFLAGING CELL GENERATION STRATEGY BY DE-CAMOUFLAGING TIME, ATTACK ON INDIVIDUAL POS AND INTRODUCED OVERHEAD

bench	# PI	# PO	# gate	time	partial	area (%)	power (%)
c432	36	7	203	1.758	7/7	2.5	1.9
c880	60	23	466	1.2×10^4	23/23	1.1	0.85
c1355	41	32	619	N/A	29/32	0.86	0.73
c1908	33	25	938	N/A	0/25	0.58	0.71
c2670	233	64	1490	N/A	60/64	0.37	0.41
c3540	50	22	1741	N/A	9/22	0.27	0.13
c5315	178	123	2608	N/A	116/123	0.17	0.11
i4	192	6	536	1.9×10^3	6/6	1.2	0.94
apex2	39	3	652	N/A	1/3	0.77	0.62
ex5	8	63	1126	6.9×10^2	63/63	0.43	0.32
i9	88	63	1186	2.1×10^4	63/63	0.45	0.22
i7	199	67	1581	1.5×10^2	67/67	0.37	0.40
k2	46	45	1906	N/A	24/45	0.21	0.17
dalu	75	16	2373	N/A	3/16	0.21	0.18

SAT-based de-camouflaging algorithm is adopted from [25] and the SPS-based removal attack is implemented following [27]. We run all the experiments on an eight-core 3.40-GHz Linux server with 32-GB RAM. The benchmarks are chosen from ISCAS and MCNC benchmarks [37], [38]. For the de-camouflaging algorithm, we set the runtime limit to 1.5×10^5 s.

A. Verification of Camouflaging Cell Generation Strategy

We first demonstrate the security achieved by using camouflaging cell generation strategy. As described in Section IV-C, we first replace all the standard cells with camouflaging cells and then, randomly change ten cells with camouflaging cells that appear to be different but work with the same functionality. In Table IV, we show the introduced overhead, de-camouflaging complexity and time. N/A indicates that the camouflaged netlist cannot be resolved within 1.5×10^5 s. As we can see, the area overhead is on average 0.68% and the power overhead is on average 0.55%, both of which are very small even for small benchmark circuits. Meanwhile, for large circuits, with the camouflaging cell generation strategy, the de-camouflaging algorithm cannot be finished within the predefined time. As we have pointed out in Section IV-C, the SAT-based algorithm can still de-camouflage some small benchmarks with less than 1600 gates. Also, for the circuits that cannot be fully de-camouflaged, we can still run de-camouflaging attacks for each primary outputs separately and partially de-camouflage the design as shown in the partial column in Table IV. The experimental results demonstrate both the effectiveness and the limitation of the camouflaging cell generation strategy.

TABLE V
EXISTING TREE STRUCTURE IN BENCHMARK CIRCUITS

bench		D-tree	ND-tree	norm KL div.
ISCAS	c1355	7	3	0.7
	c1908	14	12	0.339
	c2670	34	34	0.640
	c3540	10	9	0.671
	c5315	10	9	0.093
MCNC	i4	7	7	0.732
	ex5	6	6	0.589
	i7	4	3	0.612
	k2	175	39	0.968

B. Evaluation of AND-Tree-Based Camouflaging Strategy

To evaluate the security of the AND-tree-based camouflaging strategy, we start from stand-alone tree structures. As we can see in Fig. 15(a), both the de-camouflaging time and complexity increase exponentially as we expect. To examine the impact of tree decomposability, we fix the size of an AND-tree, i.e., 15 input pins, and change the size of the largest nondecomposable subtree in the 15-input AND-tree. The size of other nondecomposable subtrees is limited to be smaller than 3. We show the change of the de-camouflaging time and complexity in Fig. 15(b). As we have discussed in Section V-B2, the de-camouflaging complexity of the 15-input tree is limited by the sum of the de-camouflaging complexity of each subtree. As in Fig. 15(b), the de-camouflaging complexity indeed reduces exponentially with the size of the largest nondecomposable tree. We also verify the impact of input bias. We add extra circuits to the fanin cone of the tree input pins and gradually changes the input number of the extra circuits to change the KL divergence of the input distribution compared to the uniform distribution. As we show in Fig. 15(c), with the decrease of the input number of the circuits in the fanin cone, i.e., the increase of the normalized KL divergence, both the de-camouflaging time and complexity decreases.

To further examine the AND-tree structure, we consider the tree structure in the original netlist. We detect the existing AND-tree structure following Algorithm 3. In Table V, we list the input size of the largest decomposable tree detected in

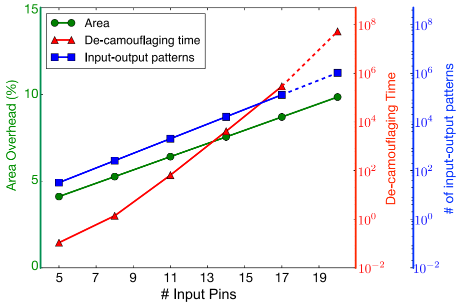


Fig. 16. Tradeoff between overhead and de-camouflaging complexity (dotted lines indicate extrapolation).

TABLE VI
INTRODUCED OVERHEAD OF THE TREE-BASED CAMOUFLAGING STRATEGY WHEN 64-INPUT AND-TREE IS INSERTED

bench	gate	overhead			security	
		area	power	timing	de-cam time	partial
i4	536	32.5	19.4	0.5	N/A	0/6
i9	1186	11.5	6.2	0.1	N/A	0/63
c2670	1490	11.8	6.0	0.1	N/A	0/64
i7	1581	9.7	4.7	0.2	N/A	0/67
dalu	2373	5.5	4.3	0.0	N/A	0/16
c5315	2608	5.9	2.8	0.0	N/A	0/123
c7552	3719	4.8	2.4	0.0	N/A	0/107
des	6729	1.9	1.2	0.0	N/A	0/245

the original netlist, i.e., D-tree, and the largest detected nondecomposable tree in the original netlist, i.e., ND-tree. For most of the circuits, the existing nondecomposable tree structure is very small. For benchmark c2670 and k2, large tree structure exists. The calculation of normalized KL divergence indicates that high bias exists for the input pins of the tree structure in k2 since the value is very close to 1. We camouflage the input pins for tree structures in both benchmarks and use the SAT-based method to de-camouflage. For c2670, original circuit functionality cannot be resolved within the predefined time threshold, while for k2, the de-camouflaging algorithm finishes within 8.5 s and 70 iterations. The results demonstrate the importance to consider both tree decomposability and input bias to evaluate the impact of the AND-tree structure in circuit netlist.

Then, we insert tree structure into the benchmark circuits following Algorithm 4. We set $\alpha = \beta = 1$ for IS evaluation. We show the tradeoff between the area overhead and the de-camouflaging time and complexity in Fig. 16 for benchmark c880. As we can see, the area overhead increases linearly with respect to the size of the inserted tree while the de-camouflaging time and complexity increases exponentially. We then insert 64-input AND-tree structure to benchmark circuits. We leverage the SAT-based attack to de-camouflage the camouflaged circuits. We also extract the subcircuits for each primary output and try to resolve the circuit separately. We show the results in Table VI. As we can see, the SAT-based attack cannot de-camouflage any primary output of each benchmark circuits. We also report the introduced overhead, including power, area, and timing in Table VI. As shown in Table VI, the main overhead comes from area and power while the impact on timing is negligible. Meanwhile, for large circuit, e.g., des, the area and power overhead is less than 2%.

TABLE VII
AREA AND POWER OVERHEAD COMPARISON WITH ANTI-SAT AND CAMOPERTURB

bench	Anti-SAT [22]		CamoPerturb [21]		Ours	
	area	power	area	power	area	power
i4	62.2	38.9	49.4	41.6	32.5	19.4
i9	23.4	12.3	29.1	29.6	11.5	6.2
c2670	24.9	12.0	25.8	19.0	15.2	6.0
i7	19.6	9.4	26.3	22.1	9.7	4.7
dalu	11.1	8.6	11.2	9.65	5.5	4.3
c5315	12.6	5.6	16.2	13.3	7.9	2.8
c7552	10.3	4.8	11.9	8.65	6.9	2.4
des	3.67	2.4	11.1	15.7	1.9	1.2

We then compare the proposed tree insertion strategy with Anti-SAT [22] and CamoPerturb proposed in [21]. We use all the three methods to insert 64-bit AND-tree into the benchmark circuits and compare the introduced power and area overhead in Table VII. Since analytical comparison on the de-camouflaging complexity is provided in Section VI-D, we do not run SAT-based attack for the three strategies. As in Table VII, our method achieves similar overhead compared with Anti-SAT. CamoPerturb suffers from larger power and area overhead compared with Anti-SAT and our strategy since large overhead is introduced in the resynthesis process.

C. Impact of Structural and Functional Camouflaging

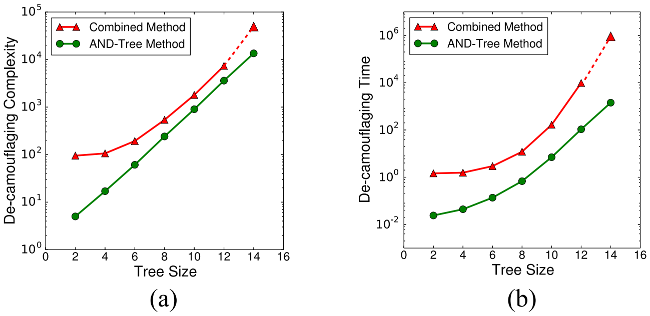
We now verify the effectiveness of the structural and functional camouflaging for the AND-tree-based camouflaging strategy and demonstrate the introduced overhead. We insert AND-tree with 64 input bins. We consider SPS-based methods proposed in [27] as functional attack and tree detection algorithm following Algorithm 2 as structural attack, which represents the state-of-the-art removal attack strategies. As shown in Table VIII, after structural and functional camouflaging, the area and power overhead increases on average by 5.1% and 0.3%. However, for large benchmarks, i.e., des, the total area and power overhead after structural and functional camouflaging is less than 3%. After structural and functional camouflaging, SPS for the each internal node of AND-tree becomes 0.0, and the size of the largest nondecomposable AND-tree that can be detected following Algorithm 3 (i.e., "detected AND-tree" in Table VIII) is 4. Therefore, structural and functional camouflaging can protect the inserted tree structure against the state-of-the-art removal attack strategies. Meanwhile, as we have discussed in Section VI-C, because the logic value of internal nodes of the AND-tree cannot be observed from the dummy connections, the overall resilience to SAT-attack is not reduced. As shown in Table VIII, for the circuit netlists after structural and functional camouflaging, SAT-attack cannot be finished within the predefined time threshold.

D. Effectiveness of Combination of Two Camouflaging Strategies

Finally, we demonstrate the effectiveness of combining the two camouflaging strategies, i.e., camouflaging cell generation strategy and AND-tree-based camouflaging strategy. To combine the two camouflaging strategies, we first insert an AND-tree structure into the original netlist following Algorithm 4.

TABLE VIII
VERIFICATION OF OVERHEAD AND EFFECTIVENESS OF STRUCTURAL AND FUNCTIONAL CAMOUFLAGING

bench	ICCAD 2016 [12]					Ours				
	area (%)	power (%)	SPS	detected ND-tree	SAT-attack	area (%)	power (%)	SPS	detected ND-tree	SAT-attack
i4	32.5	19.4	0.5	64	N/A	46.8	20.3	0.0	4	N/A
i9	11.5	6.2	0.5	64	N/A	16.4	6.5	0.0	4	N/A
c2670	15.2	6.0	0.5	64	N/A	19.1	6.3	0.0	4	N/A
i7	9.7	4.7	0.5	64	N/A	13.7	4.9	0.0	4	N/A
dalu	5.5	4.3	0.5	64	N/A	7.8	4.5	0.0	4	N/A
c5315	7.9	2.8	0.5	64	N/A	9.8	2.9	0.0	4	N/A
c7552	6.9	2.4	0.5	64	N/A	8.3	2.5	0.0	4	N/A
des	1.9	1.2	0.5	64	N/A	2.6	1.3	0.0	4	N/A



(a)

(b)

Fig. 17. Verification of the effectiveness of combining the proposed two camouflaging techniques on benchmark c880: comparison on (a) de-camouflaging complexity and (b) de-camouflaging time (dotted line indicate extrapolation).

Then, we leverage the XOR-type and STF-type cells to further camouflage the circuit netlists following the strategy described in Section IV-C. We examine the effectiveness of the combined strategy by comparing the de-camouflaging complexity and time with the situation when only AND-tree-based strategy is used. We run the experiments on benchmark c880. As shown in Fig. 17, by combining the two camouflaging strategies, both the de-camouflaging complexity and time are further increased, which indicates better security level.

VIII. CONCLUSION

In this paper, we have proposed a quantitative security criterion for de-camouflaging complexity measurements. The security criterion was formally analyzed based on the equivalence between the de-camouflaging strategy and the active learning scheme. Meanwhile, two camouflaging techniques were proposed: 1) the low-overhead camouflaging cell library and 2) the AND-tree structure, following the security criterion. A provably secure camouflaging framework was then developed to combine the two techniques, which achieves exponentially increasing security levels at the cost of linearly increasing overhead. The experimental results using the security criterion demonstrated that the camouflaged circuits with the proposed framework achieve high resilience against the SAT-based attack with an only negligible performance overhead.

REFERENCES

[1] Y. Jin, "Introduction to hardware security," *Electronics*, vol. 4, no. 4, pp. 763–784, 2015.

[2] P. Subramanyan *et al.*, "Reverse engineering digital circuits using structural and functional analyses," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 63–80, Mar. 2014.

[3] S. E. Quadir *et al.*, "A survey on chip to system reverse engineering," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, pp. 1–6, 2016.

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

[5] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware Trojans: Extended version," *J. Cryptograph. Eng.*, vol. 4, no. 1, pp. 19–31, 2014.

[6] L.-W. Chow, W. M. Clark, Jr., and J. P. Baukus, "Covert transformation of transistor properties as a circuit protection method," U.S. Patent 7 217 977, May 15, 2007.

[7] L.-W. Chow, J. P. Baukus, and W. M. Clark, Jr., "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," U.S. Patent 7 294 935, Nov. 13, 2007.

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

[9] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *Proc. IEEE/ACM Design Autom. Conf.*, 2014, pp. 1–153.

[10] A. Chen, X. S. Hu, Y. Jin, M. Niemier, and X. Yin, "Using emerging technologies for hardware security beyond PUFs," in *Proc. Design Autom. Test Europe*, Mar. 2016, pp. 1544–1549.

[11] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid STT-CMOS designs for reverse-engineering prevention," in *Proc. IEEE/ACM Design Autom. Conf.*, 2016, pp. 1–88.

[12] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. Int. Conf. Comput.-Aided Design*, 2016, pp. 1–28.

[13] K. Shamsi *et al.*, "Circuit obfuscation and oracle-guided attacks: Who can prevail?" in *Proc. IEEE Great Lakes Symp. VLSI*, 2017, pp. 357–362.

[14] 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 Security*, vol. 12, no. 1, pp. 64–77, Jan. 2017.

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

[16] A. Iyengar and S. Ghosh, "Threshold voltage-defined switches for programmable gates," in *Proc. GOMACTech*, 2016, pp. 1–2.

[17] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai, "A secure camouflaged threshold voltage defined logic family," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, 2016, pp. 229–235.

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

[19] Y.-W. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Proc. IEEE Latin–Amer. Test Symp.*, 2015, pp. 1–6.

[20] K. Shamsi *et al.*, "Cyclic obfuscation for creating SAT-unresolvable circuits," in *Proc. IEEE Great Lakes Symp. VLSI*, 2017, pp. 173–178.

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

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

[23] 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*, 2016, pp. 236–241.

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

[25] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, 2015, pp. 137–143.

[26] 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.

[27] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of Anti-SAT," in *Proc. Asia South Pac. Design Autom. Conf.*, 2017, pp. 342–347.

[28] S. Dasgupta and J. Langford, "A tutorial on active learning," in *Proc. Int. Conf. Mach. Learn.*, 2009.

[29] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *J. Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.

[30] S. Hanneke, "A bound on the label complexity of agnostic active learning," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 353–360.

[31] J. Rajendran, O. Sinanoglu, and R. Karri, "VLSI testing based security metric for IC camouflaging," in *Proc. IEEE Int. Test Conf.*, 2013, pp. 1–4.

[32] K. Shamsi *et al.*, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, 2017, pp. 95–100.

[33] R. Wiener, "An algorithm for learning Boolean functions for dynamic power reduction," Ph.D. dissertation, Dept. Comput. Sci., Univ. Haifa, Haifa, Israel, 2007.

[34] (2008). *NanGate FreePDK45 Generic Open Cell Library*. [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib

[35] *Calibre Verification User's Manual*, Mentor Graph., Wilsonville, OR, USA, 2008.

[36] (2008). *Predictive Technology Model Version 2.1*. [Online]. Available: http://ptm.asu.edu

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

[38] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," MCNC, New York, NY, USA, Rep., 1991.

Meng Li (S'15) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Texas at Austin (UT Austin), Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include hardware-oriented security, reliability, power grid simulation acceleration, and deep learning.

Mr. Li was a recipient of the Best Paper Award in HOST 2017 and the Graduate Fellowship from UT Austin in 2013.

Kaveh Shamsi (S'15) received the B.S. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with the University of Florida, Gainesville, FL, USA, under the supervision of Dr. Y. Jin.

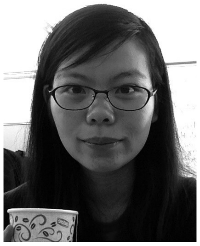
His current research interests include analog and digital circuit design and formal methods to advance hardware security.

Mr. Shamsi was a recipient of the HOST 2017 Best Paper Award.

Travis Meade (S'17) received the B.S. degree in mathematics from the University of Central Florida, Orlando, FL, USA, in 2012, where he is currently pursuing the Ph.D. degree in computer science under the supervision of Dr. S. Zhang and Dr. Y. Jin.

His current research interest includes computer algorithms of annotating gate-level netlists.

Mr. Meade was a recipient of the Best Paper Award in ASP-DAC 2016 and HOST 2017.



Zheng Zhao received the B.S. degree in automation from Tongji University, Shanghai, China, in 2012, and the M.S. degree in electrical and computer engineering from Shanghai Jiao Tong University, Shanghai, in 2015. She is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

Her current research interests include hardware security, optical computing/interconnect, and logic synthesis.

Bei Yu (S'11–M'14) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Dr. Yu was a recipient of the four Best Paper Awards at International Symposium on Physical Design in 2017, the SPIE Advanced Lithography Conference in 2016, the International Conference on Computer Aided Design in 2013, and the Asia and South Pacific Design Automation Conference in 2012, three additional Best Paper Award nominations at DAC/ICCAD/ASPDAC, and three ICCAD Contest Awards in 2015, 2013, and 2012. He has served on the editorial boards of *Integration, the VLSI Journal* and *IET Cyber-Physical Systems: Theory and Applications*.

Yier Jin (M'13) received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA, in 2012.

He is currently an Assistant Professor with the EECS Department, University of Central Florida, Orlando, FL, USA. His current research interests include trusted embedded systems, trusted hardware intellectual property (IP) cores, hardware-software co-protection on computer systems, security analysis on Internet of Things (IoT) and wearable devices with particular emphasis on information integrity and privacy protection in the IoT era. He proposed various approaches in the above area of hardware security, including the hardware Trojan detection methodology relying on local side-channel information, the post-deployment hardware trust assessment framework, and the proof-carrying hardware IP protection scheme.

Dr. Jin was a recipient of the DoE Early CAREER Award in 2016 and the Best Paper Award DAC'15, ASP-DAC'16, and HOST'17.

David Z. Pan (S'97–M'00–SM'06–F'14) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, CA, USA.

He is currently an Engineering Foundation Professor with the Department of Electrical and Computer Engineering, University of Texas at Austin (UT Austin), Austin, TX, USA. He has published over 300 refereed journal/conference papers and eight U.S. patents. His current research interests include cross-layer IC design for manufacturing, reliability, security, machine learning in EDA, design/CAD for analog/mixed signal designs, and emerging technologies.

Dr. Pan was a recipient of the many awards, including the SRC Technical Excellence Award, the 16 Best Paper Awards, the DAC Top 10 Author Award in Fifth Decade, the ASP-DAC Frequently Cited Author Award, the Communications of ACM Research Highlights, the ACM/SIGDA Outstanding New Faculty Award, the NSF CAREER Award, the IBM Faculty Award (4 times), the UCLA Engineering Distinguished Young Alumnus Award, and the UT Austin RAISE Faculty Excellence Award. He has served in many journal editorial boards and conference committees, including various leadership roles. He is a fellow of SPIE.