Provably Secure Camouflaging Strategy for IC Protection

Meng Li, Student Member, IEEE, Kaveh Shamsi, Student Member, IEEE, Travis Meade, Student Member, IEEE, Zheng Zhao, Student Member, IEEE, Bei Yu, Member, IEEE, Yier Jin, Member, IEEE, 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 IC 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 counteracting 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. 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—IC Camouflaging, SAT-based Attack, Active Learning, Provably Secure, AND-Tree

I. INTRODUCTION

With the increase of 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]–[5]. By stripping the integrated circuit (IC) layer by layer, gate-level netlist can be extracted and duplicated without the authorization of the IP holder [4], [6]. To protect IC design against RE, IC camouflaging is proposed as a layout-level technique to hide the circuit functionality [7]–[15]. By synthesizing circuits with logic cells that look alike but can have different functionalities (aka 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: fabrication level [7]–[9], [10], cell level [10], [17]–[20] and gate netlist level [10], [21]. Fabrication-level camouflaging mainly focuses on developing fabrication techniques that can hide the circuit structure. In [7], a doping based technique is proposed. By changing the polarity of dopant for the source and drain of MOS transistors, always-on and always-off transistors can be created. In [8], similar effect is realized by changing the type and length of the Lightly-Doped-Drain (LDD) implants. A dummy contact-based method is also proposed to control the connection between two adjacent layers [9]. By creating gaps in the middle of a contact, two layers that appear to connect are actually disconnected. Both doping-based and contact-based methods are shown to be robust against existing RE techniques [7], [9].

Cell-level camouflaging leverages the fabrication techniques to build camouflaging cells that look alike but may have different functionalities. In [10], the proposed cell can function as an XOR, NAND or NOR based on the configuration of the true and dummy contact. In [17], [18], by controlling the doping scheme, a camouflaged lookup table (LUT) is created with more than hundreds of functionalities. While these camouflaging cells can hide the real functionality from physical RE, they usually incur large overhead in terms of power, timing and area compared with regular cells. Gate netlist level camouflaging seeks to develop camouflaging cell insertion algorithm to maximize the resilience of the circuit netlist against RE techniques given predefined overhead constraints. For example, the authors in [10], [21], [22] insert interfered camouflaging cells or camouflaging connections to prevent RE. In [23]–[25], new low-output-corruptibility camouflaging strategies are further proposed to protect the circuit from more advanced RE techniques [26], [27].

Despite the extensive research 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 netlist-level methods usually tend to over-estimate the provided security level and in fact, have been shown vulnerable to the existing SAT-based de-camouflaging attacks as well as removal attacks based on structural and functional information [26]–[29]. 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. The proposed security criterion is defined as the number of input-output patterns that an attacker has to evaluate to decide the functionality of the original circuit and is formally analyzed by showing the equivalence between SAT-based de-camouflaging attack and the active learning scheme [30]–[32]. The proposed security criterion is independent of how the SAT-based attack is formulated or what type of machine used for the attack. The equivalence also enables us to identify two key factors that determine the security of a camouflaged netlist.

To increase the security level of the camouflaged netlist, we propose two camouflaging strategies targeting at the two identified factors. The first camouflaging strategy is a new low-overhead camouflaging cell generation, which is mainly based on the observation that the overhead of a camouflaging cell depends on its actual functionality in the circuit. We create a specific kind of camouflaging cell that incurs negligible overhead for one functionality, which allows for a large amount of insertion into the netlist and thus, provides better security protection. The second camouflaging strategy leverages the AND-tree structure for better security. We analyze the stand-alone AND-tree structure to verify its induced exponential increase of security level and further identify two important properties, denoted as tree decomposability and input bias, both of which are important to guarantee its effectiveness in general circuits. Combining these two strategies together, an IC camouflaging framework is then proposed to further optimize the camouflaged circuit for better protection against removal attacks. Experimental results demonstrate that the functionality of the camouflaged netlist generated by our framework cannot be resolved by existing de-camouflaging techniques and the overhead is negligible.
summarize our contributions as follows:

- 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 camouflaged netlist.
- We propose two novel camouflaging strategies to increase the two identified factors.
- We develop an IC camouflaging framework combining the two strategies to further protect the camouflaged circuits against removal attacks.
- 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 the paper is organized as follows. Section II provides a review on 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. Section IV and Section 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 reverse engineering (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 following two components [10]:

- The camouflaged netlist, which can be acquired from physical RE procedure [2]. The attackers can differentiate between a standard cell and a camouflaging cell, but cannot resolve the functionality of the camouflaging cells.
- A functional circuit which can be acquired from the open market and is treated as a black-box circuit.

Given the functional circuit, the attackers cannot directly probe the internal signals of the black-box circuit. Instead, they can select a sequence of input vectors, import them into the black-box circuit through circuit 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 [10], testing-based attack [10], [13] and SAT-based attack [26]–[28], [34].

Brute force attack proposes to enumerate the possible functionalities for all the camouflaging cells. Then, input vectors are randomly sampled for logic simulation to rule out the false functionalities until the original or equivalent circuit is found. Brute force attack suffers from scalability problem since the attack complexity increases exponentially with respect to the number of camouflaging cells [10]. Testing-based attack targets at one camouflaging gate at a time. For each target gate, input patterns are generated so that the output of all camouflaging gates that interfere with the target gate are known, denoted as justification, and a change at the output of the target gate causes changes at circuit primary outputs, denoted as sensitization. Here, two gates are said to interfere if their outputs are connected to the inputs of same gates, or if the output of one gate is connected to the input of the other [10]. However, when the justification and sensitization conditions cannot be satisfied simultaneously, brute force attack has to be leveraged [10]. As shown in [10], by deliberately inserting gates that interfere with each other, the complexity of testing-based attack is no better than the brute force attack.

SAT-based attack is currently the most powerful de-camouflaging attack method. The algorithm starts by treating all the possible circuit functionalities as candidates and collecting them into a set. Then, by iteratively searching the input patterns that can have different outputs for different candidates in the set, denoted as discriminating inputs [26], false functionalities are identified and removed from the set. The process continues until all the functionalities in the set have the same outputs for all input patterns. The most important characteristic of SAT-based attack is that instead of random sampling input patterns from the whole input space, only discriminating input vectors are selected by solving instances of the circuit satisfiability problem. Then, the black box functional circuit is queried to get the corresponding output vector, which are used to rule out the false functionalities.

Since all the attack strategies described above rely on querying the functional circuit, we denote them as query-based attack. Besides the query-based attack, new attack scheme proposed in [29] tries to leverage the structural and functional footprint of the camouflaging strategies to resolve the original circuit functionality. As observed in [29], existing camouflaging strategies [14], [24] that achieve high resilience against SAT-based attack usually suffer from highly biased signal probability for internal circuit nodes, which is not common in real design and thus, can be leveraged to detect camouflaging cells and determine their actual functionality. Therefore, the signal probability skew-based attack can work collaboratively with SAT-based attack. Until now, no camouflaging strategy has systematically demonstrated convincing resilience against both attack schemes. In this paper, we will develop formal analysis for both of the attacks and propose camouflaging strategies that are secure against all the existing attacks.

### B. Active Learning Scheme

In this section, we provide basic definitions concerning active learning. For more detailed description, interested readers can refer to [31].

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 $D$, 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 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 region of uncertainty $R(S)$. By iteratively sampling from $R(S)$ and updating $R(S)$ based on the new sample, $t$ can be learned from $H$. We use the following example to illustrate the concept of active learning.

**Example II.1.** Consider a two-dimensional space, and the target $t$ is a set of points lying inside a fixed rectangular in the plane as shown in Fig. 2. Assuming we already have some samples with their classification, $R(S)$ can then be decided. Consider the three points $s_1$, $s_2$ and $s_3$ in Fig. 2, the label for $s_1$ and $s_3$ 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 [31], if we define error rate \( \varepsilon_{\text{cam}}(h, t) \) for a concept \( h \) with respect to the target \( t \) and the distribution \( \mathcal{D} \) of points \( x \) as
\[
\varepsilon_{\text{cam}}(h, t) = \Pr_{x \sim \mathcal{D}}[h(x) \neq t(x)],
\]
then by adaptively sampling from \( x \in X \), to guarantee \( \varepsilon_{\text{cam}}(h, t) \leq \epsilon \) with sufficient probability, the number of samples \( m \) needed for active learning is
\[
m = \mathcal{O}(\theta d \log(\frac{1}{\epsilon}))
\]
where \( d \) is a measure of the capacity of \( H \). Specially, when \( X \) is boolean domain with \( X = \{0, 1\}^n \) and the concept class contains only boolean function, we have \( d \geq \frac{n}{\log_2|H|} \approx \frac{n}{35} \). Here \(| \theta | \) denotes the cardinality of the set. \( \theta \) is the disagreement coefficient, defined as
\[
\theta = \sup_{h \neq h'} \Pr_{x \sim \mathcal{D}}[\text{DIS}(H_x)]
\]
where \( H_x = \{ h \in H : \varepsilon_{\text{cam}}(h, t) \leq \epsilon \} \), and \( \text{DIS}(H_x) = \{ x : \exists h, h' \in H_x \text{ s.t. } h(x) \neq h'(x) \} \), and \( \Pr_{x \sim \mathcal{D}}[\text{DIS}(H_x)] = \Pr_{x \sim \mathcal{D}}[x \in \text{DIS}(H_x)] \).

III. IC CAMOUFLAGING SECURITY ANALYSIS

Let \( c_0 \) be the original circuit before camouflaging. \( c_0 \) has \( n \) input bits with the input space \( I = \{0, 1\}^n \) and \( I \) output bits with output space \( O \subseteq \{0, 1\} \). Define the indicator function \( e_{c_0} : I \times O \rightarrow \{0, 1\} \) for \( c_0 \), where \( I \times O = \{(i, o) : i \in I, o \in O\} \), as
\[
e_{c_0}(i, o) = \begin{cases} 1, & \text{if } c_0(i) = o, \\ 0, & \text{otherwise}. \end{cases}
\]
where \( e_{c_0} \) indicates whether an output vector \( o \) can be generated by \( c_0 \) given certain input vector \( i \).

During the process of IC camouflaging, \( \tilde{m} \) camouflaging gates are inserted into the original netlist, whose functionalities cannot be resolved by physical RE techniques. Let \( G \) denote the set of all possible functionalities for the camouflaging gate, where \( \forall g \in G, g : \{0, 1\}^n \rightarrow \{0, 1\} \) with \( \tilde{m} \) as the number of the camouflaging gate. Let \( y \) denote \( \tilde{m} \) functions chosen from \( G \), i.e. \( y \in G^{\tilde{m}} \), which assigns each camouflaging gate a function in \( G \) and \( y \) denote the set of all possible \( y \). Depending on \( y \), a set of possible circuit functionalities can be created, denoted as \( C \). Note that \( c_0 \in C \). \forall c \in C \), there exists a corresponding indicator function \( e_c \). Let \( E_c \) denote 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 \( c_0 \in C \) is equivalent to resolving \( e_{c_0} \in E_c \). 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 \( c_0(i) \). Based on \( (i, c_0(i)) \), all \( c \in C \) that are not consistent with \( (i, c_0(i)) \) can be pruned.

**Example III.1.** Consider the camouflaged circuit shown in Fig. 2. We have \( Y = \{\{\text{NAND, NOR}\}, \{\text{AND, OR}\}, \{\text{NAND, OR}\}, \{\text{AND, OR}\}\} \). Assume \{AND, OR\} gives the correct circuit functionality, then, for input pattern \{0001\}, by evaluating the black box functional circuit, the correct output vector should be \{00\}. This indicates \( e_{c_0}(0001, 00) = 1 \). For circuit \( c_y \) with \( y = \{\text{NAND, OR}\} \),
given input pattern \{0001\}, the output becomes \{10\}, therefore \( e_{c_y}(0001, 10) = 0 \). The indicator functions of the other two functionalities both equal to 0 given \{0001\}, \{00\}. Therefore, the input-output pattern \{0001\}, \{00\} can help rule out all the false functionalities of the camouflaged circuits.

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 \( c_0 \in C \), equivalently \( e_{c_0} \in E_c \). To evaluate the de-camouflaging complexity, we build the equivalence between the SAT-based de-camouflaging strategy and the active learning scheme as follows:

- The set of indicator functions of all possible circuit functionalities \( E_C \) corresponds to the concept class \( H \).
- The supply of indicator functions, i.e. \( I \times O \), corresponds to the set of points \( X \).
- The indicator function of the original circuit functionality \( e_{c_0} \) corresponds to the target concept \( t \).
- The input-output relation \((i, c(i)), 1\) corresponds to the samples \((x, t(x))\).
- 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 \( e_{c_0} \) with less than \( \epsilon \) error rate and sufficiently high probability is
\[
m(e_{c_0}, E_c) = \mathcal{O}(\theta d \log(\frac{1}{\epsilon})),
\]
where \( d \geq \log_2|E_c| \) is related to the number of functionalities in \( E_c \). \( \theta \) is calculated as
\[
\theta = \sup_{(i, o)} \Pr_{(i, o) \sim I \times O}[e_{c(i, o)}(i, o) \in \text{DIS}(E_c)],
\]
where \( E_c = \{ e_c \in E_c : \text{er}_{c(i, o)}(i, o) \in \text{DIS}(E_c) \} \) consists of all the indicator functions that are different from \( e_{c_0} \) with probability less than \( \epsilon \), and \( \text{DIS}(E_c) = \{ (i, o) : \exists e_c, e_c' \in E_c \text{ s.t. } e_c(i, o) \neq e_c'(i, o) \} \) consists of all the input-output pairs \((i, o)\) that leads to different outputs of any pair of indicator functions in \( E_c \). We use the following example to illustrate the \( E_c \) and \( \text{DIS}(E_c) \).

**Example III.2.** Consider the camouflaged circuit and the truth table of all the possible functionalities of the camouflaged circuit shown in Fig. 2. The correct functionality is \( c_0 \) with \( y' = \{\text{BUF, BUF}\} \). 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. \( (10, 0), (11, 1) \). If we assume \( (i, o) \) follows a uniform distribution, then \( \text{er}_{c(i, o)}(i, o) = 4/8 = 1/2 \). Similarly, we have \( \text{er}_{c_{c_1}(i, o)} = 1/2 \). If we set \( \epsilon = 1/2 \), then \( E_{c_1} = \{ e_{c_0}, e_{c_1}, e_{c_2}, e_{c_3} \} \). We can also determine \( \text{DIS}(E_{c_1}) = \{ (00, 0), (00, 1), (01, 0), (01, 1), (10, 0), (10, 1), (11, 1), (11, 1) \} \), and
Always-on

INV/
BUF

Table 1: Overhead characterization of XOR-type camouflaged cell.

<table>
<thead>
<tr>
<th>Cell</th>
<th>BUF</th>
<th>AND2</th>
<th>OR2</th>
<th>AND3</th>
<th>NAND2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
</tr>
<tr>
<td>Area</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
</tr>
<tr>
<td>Power</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
<td>1.0×</td>
</tr>
</tbody>
</table>

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

$$P_k = \Pr_{i \sim I} \left( \bigcup_{k \in [\tilde{n}]} i_k = 0 \right) \cap \left( \bigcap_{k \in [\tilde{n}] - [\tilde{n}']} i_k = 1 \right),$$

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

Meanwhile, for the STF-type camouflaging cell, since it is equivalent to creating a stuck-at fault at several input pins of the cell, these input pins become dummy as their logic values do not impact the output of the cell. This enables us to create dummy wire connections between different nodes that are not connected in the original circuit, which not only hides the original functionality, but also hides 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 cell 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). Although overhead can be introduced in the second step, we argue such overhead can be negligible since only a small subset of gates are modified in the second step. Even if we assume the attackers know the number of cells that are changed in the second step, they cannot determine which cells are changed. Therefore, the attacker still cannot determine the functionality for each cell in the camouflaged circuits. With the increase of circuit size, the total number of possible functionalities of the camouflaged netlist also increases, and thus, results in high resilience towards SAT-based attack as shown in our experimental results.

The effectiveness of the proposed camouflaging cell generation strategy is verified in Section VII. However, there are several drawbacks if we simply use the cell generation strategy to protect the circuits:

- Evaluating $|C|$ or $|E_C|$ accurately is computational intractable, which makes it hard to provide provably secure guarantee.
- The effectiveness is limited by the circuit size since we only replace original cells in the original circuits. Empirically, the proposed strategy works well for large circuits but for small circuits, the security is not sufficient.
- Even for large circuit, it is hard to protect all the circuit outputs [21].

To overcome these problems, in Section V, we will propose another camouflaging technique based on AND-tree structure, which provides provably secure guarantee.

V. AND-TREE CAMOUFLAGING STRATEGY

In this section, we target at increasing $\theta$ as in Equation (1). In [27], 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. 5. 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.

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 XOR-type camouflaging BUF cells. Recall from Section III that $I \subseteq \{0, 1\}^n$ and $Y \subseteq C^n$ represent all the possible combinations of functionalities for the camouflaged cells. For any $i \in I$ and $y \in Y$, the output of the AND-tree structure can be expressed as

$$c_y(i) = g_1(i_1) \land g_2(i_2) \land \cdots \land g_n(i_n),$$

where $i_k$ denotes the $k$th entry of input $i$, and $g_a(\cdot)$ denotes functionality of the $k$th camouflaging cell. $g_a(i_k) = i_k$ if the $k$th cell functions as a buffer, while $g_a(i_k) = \overline{i_k}$ if the $k$th cell functions as an inverter.

Let $y^* \in Y$ denote 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 VI.** 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 VI, we will first demonstrate that when the logic value 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 $c_y \circ e_{c_y}$ is $1/2^{n-1}$ compared with $e_{c_y}$. Meanwhile, we will show that $DIS(E_C) = I \times O$. Then, based on the definition of $\theta$ in Equation (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 $c_y(i^y, 1) \neq c_y^*(i^y, 1)$ and $c_y(i^y, 0) \neq c_y^*(i^y, 0)$. 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 we can write $e_{c_y} \circ e_{c_y} = e_{c_y}$. Meanwhile, $DIS(E_C) = I \times O$. Then, based on the definition of $\theta$ in Equation (2), we will prove that $\theta = 2^{n-1}$.

$$\theta = 2^{n-1} = \Pr_{i \sim I}[i = i^y \lor i = i^{y^*}] = \Pr_{i \sim I}[i = i^y] + \Pr_{i \sim I}[i = i^{y^*}]. \quad (3)$$
Note when \( y = y^* \), \( \epsilon_{(i,o)\sim I\times\phi(e_{c_y},e_{c_y})} = 0 \). Therefore, 
\[
E_c = \{ e_{c_y} \in E_C : \epsilon_{(i,o)\sim I\times\phi(e_{c_y},e_{c_y})} \leq \epsilon \} = \{ e_{c_y} \in E_C : \Pr_{i\sim I}[i = i^y] + \Pr_{i\sim I}[i = i^{\overline{y}}] = 2 \Pr_{i\sim I}[i = i^y] \leq \epsilon \} \cup \{ e_{c_y^*} \} 
\]
(4)
Because \( \forall e_{c_y} \in E_c \), where \( y \neq y^* \), is different from \( e_{c_y^*} \), at exactly four points, we have 
\[
\text{DIS}(E_c) = \{(i,o) \in I \times \phi : \Pr_{i\sim I}[i = i^y] \leq \epsilon - \Pr_{i\sim I}[i = i^{\overline{y}}], \\
\quad \quad o \in \{0,1\} \} \cup \{(i^{\overline{y}},1),(i^y,0)\}. 
\]
(5)
For tree inputs, if the logic values follow independent Bernoulli distribution with probability of 0.5, \( \forall i \in I \), we have 
\[
\Pr_{i\sim I}[i = i^y] = \Pr_{i\sim I}[i = i^{\overline{y}}] = \frac{1}{2}, 
\]
and \( \forall e_{c_y} \in E_C \) with \( y \neq y^* \), 
\[
\epsilon_{(i,o)\sim I\times\phi(e_{c_y},e_{c_y})} = \frac{2n-1}{2n-1}. 
\]
Therefore, by setting \( \epsilon = \frac{1}{2} \), we have \( E_c = E_C \) and \( \text{DIS}(E_c) = I \times \phi \). According to the definition of \( \theta \), 
\[
\theta = \frac{\Pr_{(i,o)\sim I\times\phi}[i,o] \in \text{DIS}(E_c)}{\epsilon} = 2^{n-1}. 
\]
Hence proved.

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

**Theorem V.2.** 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, 
\[
m(e_{c_y},E_C) = O(2^n). 
\]
**Proof.** Based on Lemma [V.1] we have \( \theta = 2^{n-1} \) for an \( n \)-input AND-tree. Meanwhile, because \( |E_C| = 2^n \), we have \( d \geq \log_2 |E_C|/n = 1 \). Therefore, \( m(e_{c_y},E_C) = O(2^n) \). Hence proved.

From Theorem V.2 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 a 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 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:

- The input pins to the AND-tree structure may not be primary inputs to the circuit. As shown in Fig. 7, since the fanin cone of different input pins can be overlapped, the requirement on independence may not be satisfied. Meanwhile, depending on the logic gates in the fanin cone, the signal probability for each tree input may also deviate from 0.5.
- There are usually more than one primary outputs in the circuit and more than one paths from some internal nodes of an AND-tree to the primary outputs. Consider the AND-tree structure in Fig. 7(b). The internal node 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.

Therefore, to determine the security of an AND-tree structure in general circuits, we need to characterize the two factors, which we define as input bias and tree decomposability.

![Fig. 7: Two situations that can impact the security of AND-tree structure](image)

1) Input Bias Evaluation

Input bias is proposed to characterize the distance between actual joint distribution for logic values at tree input pins and the ideal independent Bernoulli distribution.

As shown in Fig. 7(a), the logic value of input pins is determined by the primary inputs and logic gates in the fanin cones, which makes the original assumption on independent Bernoulli distribution for input pins invalid. We denote this as input bias since the actual input distribution deviates from the ideal distribution. Input bias mainly impact \( E_c \) and \( \text{DIS}(E_c) \). According to Equation (5), to decide \( \text{DIS}(E_c) \), 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 Equation (5), \( \forall y \neq y^* \), we have 
\[
\epsilon_{(i,o)\sim I\times\phi(e_{c_y},e_{c_y})} = \Pr_{i\sim I}[i = i^y] + \Pr_{i\sim I}[i = i^{\overline{y}}] \geq \Pr_{i\sim I}[i = i^y]. 
\]
To get a non-empty \( E_c \), we must choose \( \epsilon \geq \Pr_{i\sim I}[i = i^y] \). Because we always have \( \Pr_{(i,o)\sim I\times\phi}[i,o] \in \text{DIS}(E_c) \leq 1 \), then 
\[
\theta = \frac{\Pr_{(i,o)\sim I\times\phi}[i,o] \in \text{DIS}(E_c)}{\epsilon} \leq \frac{1}{\Pr_{i\sim I}[i = i^y]}. 
\]
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 \( \theta \). If the upper bound is smaller than the pre-defined 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 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.
Both of the conditions are important. For example, the AND-tree structure in Fig. (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 sub-tree structure rooted at $Node_1$, and then de-camouflage the remaining part of the tree, which is also an AND-tree structure, but with fewer input pins. The number of input vectors needed to de-camouflage the decomposable AND-tree is thus limited the sum of the input vectors needed to de-camouflage the two subtrees. Due to tree decomposability, the size of the two subtrees are much smaller than the original AND-tree, which indicates much smaller de-camouflaging complexity.

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 record of all the paths from $u$ to the primary outputs. If the tree root exists in each path, then, the tree is non-decomposable.

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. Then, accurate characterization is performed to determine the timing, power and area overhead for each cell in the camouflaging cell library. In the third step, existing AND-tree structure is detected for the original netlist. If the pre-defined de-camouflaging complexity is not satisfied, new AND-tree structure needs to be inserted as in the fourth step. Otherwise, we can simply camouflaged the AND-tree structure to enforce the resilience to structural attacks. In the fifth step, we leverage the inserted AND-tree structure to protect all the primary outputs to ensure that at least one large AND-tree exists in the fanin cone of each primary output. We further camouflaged the AND-tree structure to enhance the resilience against tree removal attack in the sixth step. After the sixth step, a camouflaged netlist will be generated.

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 pseudo code of the algorithm we propose 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 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

### Algorithm 1 Algorithm of Calculating $Pr_{i \to y}[i = i^*]$.

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

Given $Pr_{i \to y}[i = i^*]$, 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 calculate the distance between the actual distribution for logic values of tree input pins compared with the ideal distribution. Our intuition is that while the ideal distribution provides the best security, the closer the actual distribution is compared to the ideal distribution, the better security the tree structure can provide. To evaluate the distance between different distributions, we adopt the normalized Kullback-Leibler (KL) divergence $\sum P(i) \log \frac{P(i)}{Q(i)}$. (6)

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

In summary, the strategy to evaluate the input bias of existing tree structures in the original circuits becomes:

- First, evaluate $Pr_{i \to y}[i = i^*]$ following Algorithm 1 to determine the upper bound of $\theta$.
- Second, when the upper bound is large enough, do random sampling for circuit primary inputs and then, derive the logic value for the tree input pins, based on which, $KL(P, Q)$ can be calculated following Equation 6. If $KL(P, Q)$ is smaller than a pre-defined threshold, then, we consider the provided security of the tree structure to be large enough.

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.

### VI. PROVABLY SECURE IC CAMOUFLAGING

In this section, we will leverage the proposed camouflaging cell camouflaging 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. Then, accurate characterization is performed to determine the timing, power and area overhead for each cell in the camouflaging cell library. In the third step, existing AND-tree structure is detected for the original netlist. If the pre-defined de-camouflaging complexity is not satisfied, new AND-tree structure needs to be inserted as in the fourth step. Otherwise, we can simply camouflaged the AND-tree structure to enforce the resilience to structural attacks. In the fifth step, we leverage the inserted AND-tree structure to protect all the primary outputs to ensure that at least one large AND-tree exists in the fanin cone of each primary output. We further camouflaged the AND-tree structure to enhance the resilience against tree removal attack in the sixth step. After the sixth step, a camouflaged netlist will be generated.

### 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 pseudo code of the algorithm we propose 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 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

Both of the conditions are important. For example, the AND-tree structure in Fig. (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 sub-tree structure rooted at $Node_1$, and then de-camouflage the remaining part of the tree, which is also an AND-tree structure, but with fewer input pins. The number of input vectors needed to de-camouflage the decomposable AND-tree is thus limited the sum of the input vectors needed to de-camouflage the two subtrees. Due to tree decomposability, the size of the two subtrees are much smaller than the original AND-tree, which indicates much smaller de-camouflaging complexity.

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 record of all the paths from $u$ to the primary outputs. If the tree root exists in each path, then, the tree is non-decomposable.
the circuit graph. Depending on the type of the gate, there are following possibilities (lines 7–31):
- 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.
- 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 possible situations depending on the predecessors’ tree types and the tree type of the node. When the tree types are the same, larger tree structure can be formed (lines 18–19). In this case, function ADD() is called to add the predecessor’s tree structure to the node. When the tree types are different, only the predecessor itself can be added to the tree (lines 20–22).
- If the gate is NAND or NOR, we can treat it as an AND or OR connected with INV and follow the procedure above.
- For other type of gates, including XOR, XNOR, MUX and so on, no tree structure can be formed and the node itself is added to the tree (lines 24–27).

We now use an example to illustrate the Algorithm 3:

Example VI.1. Consider the circuit shown in Fig. 10(a). As in Fig. 10(b), for primary input Node1, the tree type is ANY and the input of the tree is \{Node1\}. For internal node Node5, since it is connected with Node3 through a BUF, the tree type for Node5 is also ANY and the inputs is also \{Node1\}. For Node6, 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. \{Node6\}. Consider Node7, since it is connected with an AND gate, the tree type has to be OR. For the two inputs, i.e. Node5 and Node6, since the tree types for both nodes are ANY, they can be combined to form large tree structure. Therefore, the input pins for the tree rooted at Node7 becomes \{Node5, Node6\}. Similarly, we can determine the tree type and tree inputs for Node9. Because Node9 is connected with an NOR gate, the tree type becomes AND.

After the calculation of the tree structure rooted at each node, we can examine whether the pre-defined de-camouflaging complexity is satisfied as described in Section V. If the requirement is not satisfied, new tree structures need to be inserted. In our paper, we do not consider combining existing trees in original netlists with the inserted new trees. Instead, we insert a new tree structure that is able to provide sufficient security by itself.

B. Stochastic Greedy AND-Tree Insertion
The insertion of the AND-tree structure needs to satisfy the following requirements:
- The functionality of the original circuit is not changed.
- The overhead induced by the insertion should be minimized.

Algorithm 3 Algorithm of And-Tree Detection

1. Let \{AND, ANY, OR\} denote a set of tree types.
2. \( U \leftarrow \text{TOPOLOGICALSORT}(G) \);
3. for \( u \in U \) do
4. if \( u \) is primary input then
5. \( u.\text{treetype} \leftarrow \text{ANY} \);
6. \( u.\text{treeinput} \leftarrow u \);
7. else
8. if \( u.\text{gatetype} \in \{\text{BUF}, \text{INV}\} \) then
9. \( u.\text{treetype} \leftarrow u.\text{fanin_1}.\text{treetype} \);
10. \( u.\text{treeinput} \leftarrow u.\text{fanin_1}.\text{treeinput} \);
11. else if \( u.\text{gatetype} \in \{\text{AND}, \text{NAND}, \text{OR}, \text{NOR}\} \) then
12. \( u.\text{treetype} \leftarrow \text{AND} \);
13. else if \( u.\text{gatetype} \in \{\text{OR}, \text{NOR}\} \) then
14. \( u.\text{treetype} \leftarrow \text{OR} \);
15. for \( v \in u.\text{fanin} \) do
16. if \( v.\text{treetype} = \text{u.\text{treetype}} \) and \( \text{SIZE}(v.\text{fanout}) = 1 \) then
17. \( u.\text{treeinput}.\text{ADD}(v.\text{treeinput}) \);
18. else
19. \( u.\text{treeinput}.\text{ADD}(v) \);
20. else
21. \( u.\text{treeinput} \leftarrow \text{ANY} \);
22. \( u.\text{treeinput} \leftarrow u \);
23. if \( u.\text{gatetype} \in \{\text{INV}, \text{NOR}, \text{NAND}\} \) then
24. \( u.\text{treetype} \leftarrow \text{INVERT}(u.\text{treetype}) \);
25. return \( U \).

A false interpretation of the AND-tree functionality leads to erroneous outputs.

To satisfy the first requirement, 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 Node1, we first insert an OR gate to Node0 with the other input Node2 as dummy pin. Then, an AND-tree structure is created with Node2 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 Node2, we again follow the same analysis as in Section V. The logic value of Node2, which is 0 in reality, can be expressed as \( c_y(i) = g_{n+1}(g_1(i_1) \land g_2(i_2) \land \ldots \land g_n(i_n)) \).

Note that \( g_{n+1} (i) = 0 \) indicates a stuck-at-0 fault at Node2, 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 \( c_y^r \) 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
\[
\begin{align*}
\epsilon_T(i, o) &= 1 \times O(e_{c_y^*}, e_{c_y^*}) \\
&= \Pr_{i \sim \xi} \left[ e_{c_y}(i, o) \neq e_{c_y^*}(i, o) \right] \\
&= \Pr_{i \sim \xi} \left[ (i, o) \in \{ (i^y, 0), (i^y, 1) \} \right] \\
&= \Pr_{i \sim \xi} [i = i^y].
\end{align*}
\]

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

Similar to proof in Section \textbf{V-A}, if we set \( \epsilon = \frac{1}{2^n} \), then, we have \( E_e = E_C \) and \( \text{DIS}(E_e) = T \times O \). 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 de-camouflage 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.

**Algorithm 4** Algorithm of Stochastic Greedy AND-Tree Insertion

1. \( U_{PO} \leftarrow \text{POs} \); 
2. \( U \leftarrow \text{TOPOLOGICALSORT}(G) \); 
3. \( \text{REMOVECRITICALNODE}(U) \); 
4. \textbf{while} \( U_{PO} \neq \emptyset \) \textbf{do} 
5. \( \text{for } u \in U \textbf{ do} \) 
6. \( \text{u.score} \leftarrow \text{COMPUTEIS}(G) \); 
7. \( U_{IS} \leftarrow \text{FINDTOPK}(U) \); 
8. \( u_c \leftarrow \text{RANDOMSELECTCAND}(U_{IS}) \); 
9. \( G \leftarrow \text{ANDTREEMERGE}(u_c, G) \); 
10. \( U_{PO} \leftarrow \text{REMOVECOVEREDPO}(U_{PO}, u_c) \); 
11. \textbf{return} \( U \); 

To determine the location for the insertion of the tree structure, 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 that we hope to protect in a set \( U_{PO} \). Then, to decide the candidate circuit node for tree insertion, we traverse the circuit graph in a topological order and calculate an insertion score (IS) for each internal node. 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
\[
IS = \frac{\alpha \times SA - \beta \times P_{ob}}{N_o}.
\]

By defining IS following Equation (8), we look for the circuit node with lowest average cost to camouflage one primary output. Here, cost is defined considering introduced power overhead \( SA \) and error observability \( P_{ob} \). \( \alpha \) and \( \beta \) are coefficients defined to balance \( SA \) and \( P_{ob} \). By increasing \( \alpha \), we can reduce the introduced power overhead, while by increasing \( \beta \), we prefer circuit nodes which leads to better error probability at primary outputs. Note that before we calculate the score, all the circuit nodes along timing critical paths are removed first to guarantee negligible impact on performance. Then, we find \( k \) nodes with 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.

The insertion of AND-tree structure helps camouflage the functionality of original netlist. While the timing overhead can be small since the nodes along critical paths are not changed, the introduced power and area overhead cannot be avoided. However, the size of the inserted tree only depends on the required security level and is independent of the size of original netlist. Meanwhile, while the induced area and power overhead increases linearly as the tree size, the de-camouflaging complexity increases exponentially. Therefore, to ensure certain de-camouflaging complexity, the overhead is acceptable. For relatively large circuit, the overhead is even negligible.

More importantly, the de-camouflaging complexity, which is defined as the number of input-output patterns required for de-camouflaging, is independent of the way that a SAT-problem is formulated and the software package or computer configuration that the attack is carried on. Therefore, the proposed camouflaging framework is provably secure provided that the requirement on the size of non-decomposable tree and input bias is satisfied.

**C. AND-Tree Camouflaging Against Removal Attack**

By inserting AND-tree into original circuit netlists, the de-camouflaging complexity can be increased exponentially, which ensures good resilience against SAT-based attack. However, because large AND-tree is a unique structure that does not usually exist in general circuit netlist, it is possible for the attacker to identify and remove it. In [29], the authors propose to identify the AND-tree by calculating the signal probability skew (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 attacker assumes same probability for each functionality, and calculate the signal probability as the average value. For a signal with large uncertainty, its SPS tends to approach 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 non-decomposable 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. To protect the inserted AND-tree from such removal attack, 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 functionality, as in Fig. 13(b). Then, we replace the NAND, NOR and INV cells with XOR-type camouflaging cells that look differently 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 [29]. Therefore, functional attacks by SPS are rendered useless.

To prevent removal attack based on 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 a $n$-input AND-tree, there are in total $n-1$ 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 $n'$, we can always insert $O((n-1)/(n'-1))$ 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 by inserting dummy connections with STF-type cells, the original circuit functionality is maintained. Meanwhile, the inserted AND-tree is still non-decomposable from the defense perspective since the logic value of AND-tree internal nodes cannot be sensitized through the dummy connections. However, from the attackers' point of view, because he cannot determine the connections are dummy based on structural attack following Algorithm 3 the largest non-decomposable tree that can be detected by structural attack is reduced significantly. At the same time, because structural and functional camouflaging focuses on internal nodes of the tree structure, the input bias is not impacted as well. 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

Until now, we have described our IC camouflaging strategy that detects, inserts and camouflages AND-tree structure to provide guaranteed security towards SAT-based attack. Similar idea to leverage AND-tree structures has also been explored by Anti-SAT [24] and CamoPerturb [23]. In this section, we compare the three strategies in terms of provided security, overhead and their impact on the original circuit netlist, i.e. whether re-synthesis is required.

Assume an AND-tree with $n$-bit inputs is to be inserted into the circuit. According to Anti-SAT strategy, in fact, two subtrees, denoted as $Sub_1$ and $Sub_2$, are inserted into the circuit. $Sub_1$ and $Sub_2$ are AND-trees of the same size with $n$ input pins, and share the same input signals. An inverter 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$^{n}$ input vectors [24].

For CamoPerturb strategy, to insert a $n$-bit AND-tree, a specific input vector $i^*$ is first selected. Then, original circuit is re-synthesized by flipping the output value corresponding to $i^*$. An AND-tree is then inserted to correct the flipped output just for $i^*$. XOR-type BUF cells are inserted into the input pins of the AND-tree and their actual functionality in the circuit is determined by $i^*$. Based on [23], 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, it should be noted that $i^*$ can rule out all the false functionalities. Because $i^*$ is unknown to the attackers, on average, $2^{n-1}$ input vectors need to be measured. In the best case, the attacker has to measure $2^n$ input vectors to de-camouflage the AND-tree. However, in the worst case, only 1 input vector, i.e. $i^*$ is required for the attacker. It should be noted that for our AND-tree insertion strategy, as shown in TABLE III the de-camouflaging complexity is also $2^n$. This is because with our strategy, the circuit output is never impacted by the AND-tree output due to the stuck-at-0 input pin at the XOR gate. Therefore, 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 re-synthesis. As we show in Section VII-B large overhead can be introduced in the process of re-synthesis depending on $i^*$. We summarize the comparison in TABLE III as we can see, our strategy for AND-tree insertion provides the best security guarantee. Meanwhile, because no re-synthesis is required for our methods, it introduces less modification to the original design and thus is easier for final design closure.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>De-cam Complexity</th>
<th>Overhead</th>
<th>Resyn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Worst</td>
<td>Avg</td>
<td>Best</td>
</tr>
<tr>
<td>Ours</td>
<td>$2^n$</td>
<td>$2^{n-1}$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>CamoPerturb</td>
<td>1</td>
<td>$2^{n-1}$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>Anti-SAT</td>
<td>$2^n$</td>
<td>$2^n$</td>
<td>$2^n$</td>
</tr>
</tbody>
</table>

### 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 SAT-based de-camouflaging algorithm is adopted from [27] and the SPS-based removal attack is implemented following [29]. 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 [40], [41]. For the de-camouflaging algorithm, we set the runtime limit to $1.5 \times 10^5$ seconds.

**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 10 cells with camouflaging cells that appear to be different but work with same functionality. We show the introduced overhead, de-camouflaging complexity and the time required for the SAT-based algorithm to resolve the original circuit functionality in TABLE IV. N/A indicates that the camouflaged netlist cannot be resolved within $1.5 \times 10^5$ seconds. As we can see, the area overhead, which is calculated as the sum of the area of each cell, 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, simply with the camouflaging cell generation strategy, the de-camouflaging algorithm cannot be finished within the pre-defined time. However, 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 the effectiveness of the camouflaging cell generation strategy for large circuit, and also shows the necessity to have other SAT-resilient protection strategies.

**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. We show the increase of the de-camouflaging complexity and time with respect to the tree size in Fig. 13(a). As we can see, 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 non-decomposable subtree in the 15-input AND-tree. The size of other non-decomposable 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. When the size of the largest non-decomposable tree is much larger than the other subtrees, the de-camouflaging complexity of the 15-input tree is mainly determined by its largest non-decomposable subtree. As in Fig. 15(c) the de-camouflaging complexity indeed reduces exponentially with the size of the largest non-decomposable 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 uniform distribution. As we show in Fig. 15(c) with the decrease of the input number of the added 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 2 in TABLE V we list the input size of the largest decomposable tree detected in the original netlist, i.e. D-tree, and the largest detected non-decomposable tree in the original netlist, i.e. ND-tree. For most of the circuits, the existing non-decomposable tree structure is very small. For benchmark cis670 and k2, large tree structure exists. The calculation of normalized KL divergence indicates that high bias exists for the input pins of tree structure in k2 since the value is very close to 1. We camouflaged the input pins for tree structures in both benchmarks and use SAT-based method to de-camouflage. For cis670, original circuit functionality cannot be resolved within the predefined time threshold, while for k2, the de-camouflaging algorithm finishes within 8.5 seconds 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 3. We set $\alpha = \beta = 1$ for IS evaluation. We show the trade-off between the area overhead and the de-camouflaging time and complexity in Fig. 15(d). As we can see, the area overhead increases linearly with the increase of the de-camouflaging time and complexity.
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 non-decomposable tree; (c) change of de-camouflaging complexity and time with the input bias.

Fig. 16: Trade-off 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.

<table>
<thead>
<tr>
<th>bench</th>
<th>area</th>
<th>power</th>
<th>timing</th>
<th>de-cam time</th>
<th>partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>i4</td>
<td>536</td>
<td>32.5</td>
<td>19.4</td>
<td>0.5</td>
<td>N/A</td>
</tr>
<tr>
<td>i9</td>
<td>1186</td>
<td>11.5</td>
<td>6.2</td>
<td>0.1</td>
<td>N/A</td>
</tr>
<tr>
<td>c2670</td>
<td>1490</td>
<td>11.8</td>
<td>6.0</td>
<td>0.1</td>
<td>N/A</td>
</tr>
<tr>
<td>i7</td>
<td>1581</td>
<td>9.7</td>
<td>4.7</td>
<td>0.2</td>
<td>N/A</td>
</tr>
<tr>
<td>dalu</td>
<td>2373</td>
<td>5.5</td>
<td>4.3</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>c5315</td>
<td>2608</td>
<td>5.9</td>
<td>2.8</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>c7552</td>
<td>3719</td>
<td>4.8</td>
<td>2.4</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>des</td>
<td>6729</td>
<td>1.9</td>
<td>1.2</td>
<td>0.0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

We then compare the proposed tree insertion strategy with Anti-SAT [24] and CamoPerturb proposed in [23]. 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 VII-D, we do not run SAT-based attack for the three strategies. As shown 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 re-synthesis 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 [29] 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 are 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 non-decomposable 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 VII-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 pre-defined 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. 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: the low-overhead camouflaging cell library and 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. Experimental results using the security criterion demonstrated that the camouflaged circuits with the proposed framework achieve high resilience against the SAT-based attack with only negligible performance overhead.

REFERENCES


Meng Li received his 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, University of Texas at Austin, under the supervision of Prof. David Z. Pan. His research interests include hardware-oriented security, reliability, power grid simulation acceleration and deep learning. He received best paper award in HOST 2017 and 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 a Ph.D. degree in Computer Engineering at the University of Florida Florida, Gainesville, USA under the supervision of Dr. Yier Jin. His research focuses on using analog and digital circuit design, and formal methods to advance hardware security. He is the recipient of the HOST 2017 best paper award.

Travis Meade received his B.S. degree in Mathematics from University of Central Florida, Orlando, Florida, United States of America in 2012. He is currently pursuing a Ph.D. degree in Computer Science at University of Central Florida under the supervision of Dr. Shaojie Zhang and Dr. Yier Jin. His research focuses on using computer algorithms of annotating gate-level netlists. He received best paper award in ASP-DAC 2016 and HOST 2017.

Zheng Zhao received his B.S. degree in Automation from Tongji University, Shanghai, China in 2012 and M.S degree in Electrical and Computer Engineering from Shanghai Jiao Tong University, Shanghai, China in 2015. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering, University of Texas at Austin, under the supervision of Prof. David Z. Pan. Her research interests include hardware security, optical computing/interconnect and logic synthesis.

Bei Yu (S’11–M’14) received his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin in 2014. He is currently an Assistant Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has served in the editorial board of Integration, the VLSI Journal and JET Cyber-Physical Systems: Theory & Applications. He received four Best Paper Awards at International Symposium on Physical Design 2017, SPIE Advanced Lithography Conference 2016, International Conference on Computer Aided Design 2013, and Asia and South Pacific Design Automation Conference 2012, plus three additional Best Paper Award nominations at DAC/ICCAD/ASPDAC and three ICCAD contest awards in 2015, 2013 and 2012.

Yier Jin is currently an assistant professor in the EECS Department at the University of Central Florida. He received his Ph.D degree in Electrical Engineering in 2012 from Yale University after he got his B.S. and M.S. degrees in Electrical Engineering from Zhejiang University, China, in 2005 and 2007, respectively. His research focuses on the areas of trusted embedded systems, trusted hardware Intellectual property (IP) cores and hardware-software co-protection on computer systems. He proposed various approaches in the 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. He is also interested in the security analysis on Internet of Things (IoT) and wearable devices with particular emphasis on information integrity and privacy protection in the IoT era. He is awarded the DoE Early CAREER Award in 2016 and is the best paper award recipient of DAC’15, ASP-DAC’16, and HOST’17.

David Z. Pan (S’97-M’00-SM’06-F’14) received his B.S. degree from Peking University, and his M.S. and Ph.D. degrees from University of California, Los Angeles (UCLA). From 2000 to 2003, he was a Research Staff Member with IBM T. J. Watson Research Center. He is currently the Engineering Foundation Professor at the Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include cross-layer nanometer IC design for manufacturability, reliability, physical security, physical design, analog design automation, and CAD for emerging technologies such as 3D-IC and nanophotonics. He has published over 280 technical papers, and is the holder of 8 U.S. patents. He has graduated 21 PhD students who are now holding key academic and industry positions.

He has served as a Senior Associate Editor for ACM Transactions on Design Automation of Electronic Systems (TODAES), an Associate Editor for IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on Very Large Scale Integration Systems (TVLSI), IEEE Transactions on Circuits and Systems PART I (TCAS-I), IEEE Transactions on Circuits and Systems PART II (TCAS-II), IEEE Design & Test, Science China Information Sciences, Journal of Computer Science and Technology, IEEE CAS Society Newsletter, etc. He has served in the Executive and Program Committees of many major conferences, including DAC, ICCAD, ASPDAC, and ISPD. He is the ASPDAC 2017 Program Chair, ICCAD 2018 Program Chair, DAC 2014 Tutorial Chair, and ISPD 2008 General Chair.

He has received a number of awards for his research contributions, including the SRC 2013 Technical Excellence Award, DAC Top 10 Author in Fifth Decade, DAC Prolific Author Award, ASP-DAC Frequently Cited Author Award, 14 Best Paper Awards at premier venues (HOST 2017, SPIE 2016, ISPD 2014, ICCAD 2013, ASPDAC 2012, ISPD 2011, IBM Research 2010 Pat Goldberg Memorial Best Paper Award, ASPDAC 2010, DATE 2009, ICICDT 2009, SRC Techcon in 1998, 2007, 2012 and 2015) plus 11 additional Best Paper Award nominations at DAC/ICCAD/ASPDAC/ISPD, Communications of the ACM Research Highlights (2014), ACM/SIGDA Outstanding New Faculty Award (2005), NSF CAREER Award (2007), SRC Inventor Recognition Award three times, IBM Faculty Award four times, UCLa Engineering Distinguished Young Alumni Award (2009), UT Austin RAISE Faculty Excellence Award (2014), and many international CAD contest awards, among others. He is a Fellow of IEEE and SPIE.