Provably Secure Camouflaging Strategy for IC Protection *

Meng Li[1], Kaveh Shamsi[2], Travis Meade[2], Zheng Zhao[1], Bei Yu[3], Yier Jin[2], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]ECE Department, University of Central Florida, Orlando, FL, USA
[3]CSE Department, The Chinese University of Hong Kong, NT, Hong Kong

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 over-estimation of the security level when countering the 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 novel camouflaging techniques are proposed, the low-overhead camouflaging cell library and the AND-tree structure, 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 by combining these two techniques. Experimental results using the security criterion show that the camouflaged circuits with the proposed framework are of high resilience against the SAT-based attack with negligible performance overhead.

1. INTRODUCTION

As IC design costs increase, intellectual property (IP) protection becomes a significant concern for the semiconductor industry. One of the major threats arises from reverse engineering (RE) [1, 2]. By stripping the integrated circuit (IC) layer by layer, gate-level netlist can be extracted and duplicated without the authorization of the IP holder [2,3]. To protect IC design against RE, IC camouflaging is proposed as a layout-level technique to hide the circuit functionality [4–7]. 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 works on IC camouflaging mainly fall into the following three categories: fabrication level [4–6], cell level [7–9] and netlist level [7, 10]. The fabrication-level camouflaging mainly focuses on developing fabrication techniques that can hide the circuit structure. In [4], 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 [5], 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 [6]. By creating gaps in the middle of a contact, two layers that appear to connect are actually disconnected. Both doping-based and contact-based method are shown to be robust against existing RE techniques [4, 6]. The cell-level camouflaging leverages the fabrication techniques to build camouflaging cells that look alike but may have different functionalities. In [7], the proposed cell can function as an XOR, NAND or NOR gate based on the configuration of the true and dummy contact. In [8, 9], 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. The 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 [7] claimed that by inserting camouflaging cells that can interfere with each other, the camouflaged netlists would require more than 1000 years to resolve. The proposed technique is further improved in [10] with the fanin cone based analysis.

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 [11–13]. 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 circuit. 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. Because the proposed criterion is independent of the way that a SAT-problem is formulated and the software package or computer configuration that the attack is carried on, *the camouflaged circuit that achieves high enough de-camouflaging complexity is guaranteed to be secure against de-camouflaging attack.*

To formally analyze the de-camouflaging complexity, we build the equivalence between SAT-based de-camouflaging attack and the active learning scheme [14–16]. Based on the equivalence, the security criterion can be analytically derived for camouflaged circuits and two key factors that determine the security are revealed, i.e, the number of possible functionalities of the camouflaged netlist and the output hamming distance among different functionalities. To enhance the two factors in circuits, we propose two camouflaging strategies, including a novel low-overhead camouflaging cell generation strategy and an AND-tree structure. We observe the overhead of a camouflaging cell is determined by its actual functionality in the circuit, and thus, create a specific type of camouflaging cell that incurs negligible overhead for one functionality to allow for a large amount of insertion into the netlist and create a huge number of functionalities. The AND-tree structure is proposed to control the output hamming distance for better security. We analyze the standalone AND-tree structure to prove its induced exponential increase of the security level and further identify two important properties of the AND-

*This work is supported in part by CUHK Direct Grant for Research.

ICCAD '16, November 07-10, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2966986.2967065

tree structure that are important to guarantee its effectiveness in general circuits. An IC camouflaging framework is then proposed to combine the two methods together. 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. We summarize our contributions as follows:

- A new security criterion is proposed and formally derived, which enables to capture the trade-off among security, output error probability and hardware cost analytically.
- Two factors that can enforce the security criterion in camouflaged circuits are revealed, with two novel camouflaging strategies proposed to enhance each factor in circuits.
- An IC camouflaging framework is proposed through the combination of these two techniques and is provably secure against existing de-camouflaging attacks.
- The proposed security criterion and IC camouflaging framework are verified with the state-of-the-art de-camouflaging technique with great resilience and negligible overhead.

The rest of the paper is organized as follows. Section 2 provides a review on existing de-camouflaging attacks and the preliminaries on active learning scheme. Section 3 formally builds the equivalence between de-camouflaging and active learning with key security factors revealed. Section 4 and Section 5 describe two camouflaging techniques to enhance security in circuits. Section 6 proposes our overall IC camouflaging framework. Section 7 demonstrates the performance of the proposed framework, followed by conclusion in Section 8.

2. BACKGROUND

In this section, the de-camouflaging attack model and attack techniques are reviewed. We also talk about the active learning scheme, which lay the foundation for our analysis on de-camouflaging complexity in Section 3.

2.1 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 as stated in [7]:

- The attacker has the techniques to reverse engineer an IC to get the camouflaged netlist, including delayering, depackaging, imaging and so on [2], which we refer to as physical RE techniques.
- The attacker can differentiate between a standard cell and a camouflaging cell, but cannot resolve the specific functionality of the camouflaging cell.
- The attacker can acquire a functional circuit as a black box and get the correct outputs for given input vectors.

To recover the original circuit functionality, after reverse engineering the chip with physical techniques, the attacker will query the functional circuit to get the correct input-output patterns to decide the functionality of camouflaged cells. To explore the input-output patterns, three different methods have been proposed, including brute force attack [7], testing-based attack [7, 17] and SAT-based attack [11–13].

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 [7]. 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. However, when the justification and sensitization conditions cannot be satisfied simultaneously, brute force attack has to be leveraged [7]. 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 [11], 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. Currently, the SAT-based attack has been demonstrated to achieve great capability to resolve the camouflaging gates and recover the functionality of the original circuits. No existing camouflaging strategy has demonstrated convincing resilience to the SAT-based attack. In this paper, we will develop formal analysis on the number of discriminating inputs required for the de-camouflaging attack and propose camouflaging strategies that are provably secure against the SAT-based attack.

2.2 Active Learning Scheme

In this section, we provide basic definitions concerning active learning. For more detailed description, interested readers can refer to [15].

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 may be implicitly 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 $\mathcal{R}(S)$. By iteratively sampling from $\mathcal{R}(S)$ and updating $\mathcal{R}(S)$ with 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 two-dimensional space, and the target t is a set of points lying inside a fixed rectangular in the plane as shown in Figure 1. Assuming we already have some samples with their classification, $\mathcal{R}(S)$ can then be decided accordingly. As in Figure 1, among s_1,s_2 and s_3, only sample s_3 can provide further information to decide t from H.*

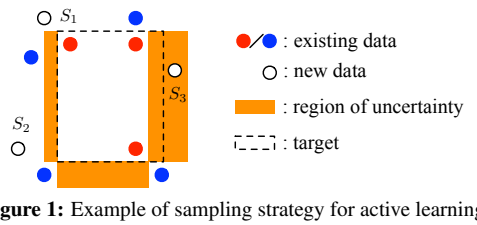


Figure 1: Example of sampling strategy for active learning.

According to [15], if we define error rate $er_{x\sim\mathcal{D}}(h,t)$ for a concept h with respect to the target t and the distribution $\mathcal{D}$ of points x as $er_{x\sim\mathcal{D}}(h,t) = \mathsf{Pr}_{x\sim\mathcal{D}}[h(x) \neq t(x)]$, then, by adaptively sampling from $x \in X$, to guarantee $er_{x\sim\mathcal{D}}(h,t) \leq \epsilon$ with sufficient probability,

on average, the number of samples m needed for active learning is

$$m = O(\theta d \log(\frac{1}{\epsilon})),$$

where d is a measure of the capacity of H. Speically, when X is boolean domain with $X = \{0,1\}^n$ and the concept class contains only boolean function, we have $d \geq \frac{\log_2|H|}{n}$ [18]. Here $|\cdot|$ denotes the cardinality of the set. θ is the disagreement coefficient, defined as

$$\theta = \sup_\epsilon \frac{\mathsf{Pr}_{x\sim\mathcal{D}}[\mathrm{DIS}(H_\epsilon)]}{\epsilon},$$

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

3. IC CAMOUFLAGING SECURITY ANALYSIS

In this section, we formally analyze the proposed security criterion for the camouflaged circuits. By building the equivalence between SAT-based de-camouflaging strategy and the active learning scheme, the de-camouflaging complexity is derived and key factors to enhance the security criterion are revealed as well.

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 $O \subseteq \{0,1\}^l$. 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 denote the set of all possible functionalities for the camouflaging gate, where $\forall g \in G$, $g : \{0,1\}^{\widetilde{n}} \rightarrow \{0,1\}$ with $\widetilde{n}$ as the input number of the camouflaging gate. Let y denote $\widetilde{m}$ functions chosen from G, i.e. $y \in G^{\widetilde{m}}$, which assigns each camouflaging gate a function in G and let Y denote the set of all possible y. Depending on y, a set of possible circuit functionalities can be created, denoted as C. Note that $c_o \in C$.

Based on the attack model described in Section 2, 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 from physical RE. To resolve c_o from C, the attacker is able to select input pattern $i \in I$ and apply to the black box circuit to get the corresponding output $c_o(i)$. The input-output pattern $(i, c_o(i))$ can be used to rule out incorrect functionalities in C. As we have described in Section 2, for the SAT-based attack, it leverages the SAT solver to decide the input-output patterns that can help rule out incorrect assignments.

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 in C and resolve the $c_o \in C$. 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 input space I corresponds to the set of points X;
- The set of possible circuit functionalities C corresponds to the concept class H;
- The original circuit functionality c corresponds to the target concept t;
- The input-output pattern $(i, c(i))$ 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 the functionality of c with less than ϵ output error rate and sufficiently high probability is

$$m(c_o, C) = O(\theta d \log(\frac{1}{\epsilon})), \quad (1)$$

where $d \geq \frac{\log_2|C|}{n}$ is related to the number of functionalities in C and $\theta = \sup_\epsilon \frac{\mathsf{Pr}_{i\sim\mathcal{I}}[\mathrm{DIS}(C_\epsilon)]}{\epsilon}$ is related to the output hamming distance for different functionalities, which is decided by the circuit structure. C_ϵ is adopted from the active learning scheme and corresponds to H_ϵ. It shall be noted that although the derived de-camouflaging complexity is an average case estimation, it is a practical evaluation of the circuit security level. More importantly, it reveals the inherent trade-off among output error rate, hardware cost and de-camouflaging complexity. That is, to achieve large de-camouflaging complexity, more hardware cost is required to increase the number of possible functionalities of the camouflaged circuits and the output error rate needs to be reduced to minimize the hamming distance among different functionalities.

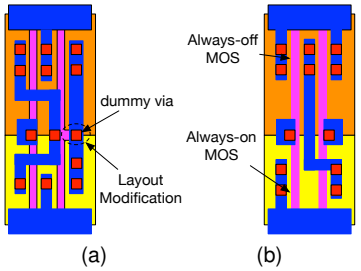


Figure 2: Illustration of two different cell camouflaging strategies: (a) XOR-type and (b) STF-type.

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

	BUF		AND2		OR2		AND3	
Function	BUF	INV	AND2	NAND2	OR2	NOR2	AND3	NAND3
Timing	1.0x	2.0x	1.0x	1.5x	1.0x	1.9x	1.0x	1.8x
Area	1.0x	1.5x	1.0x	1.3x	1.0x	1.3x	1.0x	1.3x
Power	1.0x	1.5x	1.0x	0.9x	1.0x	1.1x	1.0x	1.0x

4. NOVEL CAMOUFLAGING CELL DESIGN

In this section, we target at increasing d as in Eq. (1). Because the lower bound of d is in proportional to $|C|$, we choose to increase the possible functionalities of the camouflaged netlist to increase d. To accomplish this, traditional methods usually target at increasing the possible functionalities for the camouflaged cell. However, this usually causes large overhead in terms of power, area and timing, which largely limits the number of camouflaging cell that can be inserted into the netlist, and thus, limit the total number of possible functionalities. We propose two camouflaging cell design strategies, termed as XOR-type strategy and stuck-at-fault-type (STF-type) strategy based on the observation that for one camouflaging cell, its overhead is determined by its functionality in the netlist. By creating camouflaging cells with negligible overhead for one specific functionality, we are able to insert large amount of camouflaging gates, most of which functions with negligible overhead, into the original netlist.

4.1 XOR-type Cell Camouflaging Strategy

The XOR-type camouflaging strategy leverages the dummy contacts. As shown in Figure 2 (a), for a BUF cell, we modify the shape of the polysilicon to create extra overlap between polysilicon and metal layer. Then, a contact is inserted to connect the two layers. Depending on whether the contact is real or dummy, the cell functions either as an INV or a BUF.

To evaluate the overhead of the XOR-type camouflaged cells, standard cells from Nangate 45nm Open Cell Library [19] are modified according to the strategy and scaled to 16nm technology. Then, Calibre xRC [20] is used to extract parasitic information of the cell layouts. We use SPICE simulation to characterize different types of gates, which is based on 16nm PTM model [21]. As we can see from Table 1, when the contact is dummy, the overhead induced by the layout modification is negligible compared with original standard cells. However, when the contact is real, large overhead can be observed for timing, area and power. Note that for the XOR-type cell camouflaging strategy, when the attacker misinterprets the contact, the probability of logic error at the output of the cell is 1.

4.2 STF-type Cell Camouflaging Strategy

The STF-type camouflaging strategy leverages the doping-based technique. The camouflaging cell generated with the STF-type camouflaging 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 LDD, 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 Figure 2 (b), for a two-input NAND cell, we can use Always-on doping scheme for the NMOS transistor and Always-off doping scheme for the PMOS transistor associated with input 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.

Table 2: Overhead characterization of STF-type camouflaged cell.

	AND2		OR2		NAND2		NOR2	
Function	AND2	BUF	OR2	BUF	NAND2	INV	NOR2	INV
Timing	1.0x	1.4x	1.0x	1.4x	1.0x	1.6x	1.0x	1.6x
Area	1.0x	1.3x	1.0x	1.3x	1.0x	1.5x	1.0x	1.5x
Power	1.0x	1.2x	1.0x	1.2x	1.0x	1.5x	1.0x	1.5x

It is obvious that the STF-type camouflaging strategy does not impact the timing or performance when it functions normally since the layout is not modified. However, when Always-on or Always-off scheme is used, the overhead needs to be characterized. We use similar method as described for the XOR-type camouflaging strategy and the overhead results is listed in Table 2. Note that unlike the XOR-type, a mis-interpretation of the doping scheme may not always lead to incorrect logic value at the output of the gate. 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

$$\mathsf{Pr}_{\text{error}} = \underset{i\sim I}{\mathsf{Pr}}\left[\left(\bigcup_{k\in[\widetilde{n}']} i_k = 0\right) \cap \left(\bigcap_{k\in[\widetilde{n}]\setminus[\widetilde{n}']} i_k = 1\right)\right]$$

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

Meanwhile, compared with XOR-type camouflaging strategy, the generated dummy inputs enable us to create dummy wire connections between different nodes that are not connected originally in the circuit, which further hides the circuit functionality.

The proposed camouflaging cell design strategy enables us to create a huge amount of possible functionalities for the camouflaged circuit because 1) combination of two techniques greatly increase the number of possible configurations for each camouflaging cell, and 2) negligible overhead enables much more insertions of camouflaging cells into original netlist. We verify the effectiveness of the proposed method in Section 7. However, simply leveraging the camouflaging cell generation strategy is not secure enough since 1) evaluating the size of C can be computationally intractable and thus, it is hard to provide provably secure guarantee; 2) the camouflaging strategy is usually more effective for large circuits while for small circuits, the circuit functionality can still be completely or partially resolved. To overcome the listed problems, we will propose another techniques based on AND-tree structure, which will provide us provably secure guarantee with negligible overhead.

5. AND-TREE CAMOUFLAGING STRATEGY

In this section, we target at increasing θ as in Eq. (1). θ is related to the output hamming distance for different functionalities and is mainly decided by the circuit structure. In [13], AND-tree structure is noticed to achieve good resilience to SAT-based de-camouflaging attack when the input pins are camouflaged as shown in Figure 3. 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.

5.1 Security Analysis of AND-Tree Structure

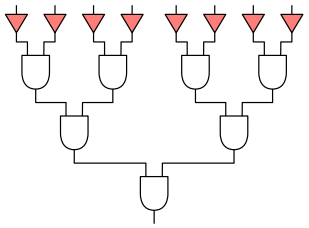


Figure 3: Example of a camouflaged and-tree structure.

Consider the AND-tree structure with n input pins shown in Figure 3 where all the input pins are camouflaged with the XOR-type camouflaging BUF. Recall from Section 3 that $I \subseteq \{0,1\}^n$ and $Y \subseteq G^n$ represents all the possible combination of functions for the camouflaging cells. For any input $i \in I$ and for any $y \in Y$, the output of the AND-tree structure can be expressed as

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

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

Let $y^* \in Y$ denote the correct configuration for all the camouflaging cells. Then, depending on the value of y, there are 2^n different circuit functionalities. For any $y \in Y$, there exists exactly one input $i \in I$ such that $f_y(i) = 1$, denoted as i^y. Therefore, we have $\mathsf{Pr}_{i\sim I}[f_y(i) = 1] = \mathsf{Pr}_{i\sim I}[i = i^y]$. Further, any false configuration y is different compared with y^* for exactly two input vectors.

Moreover, for any $y \neq y^*$, the output error rate becomes

$$\begin{aligned}
&\underset{i\sim I}{er}(y, y^*)\\
=&\underset{i\sim I}{\mathsf{Pr}}[f_y(i) \neq f_{y^*}(i)]\\
=&\underset{i\sim I}{\mathsf{Pr}}[f_y(i) = 1 \wedge f_{y^*}(i) = 0] + \underset{i\sim I}{\mathsf{Pr}}[f_y(i) = 0 \wedge f_{y^*}(i) = 1]\\
=&\underset{i\sim I}{\mathsf{Pr}}[f_y(i) = 1] + \underset{i\sim I}{\mathsf{Pr}}[f_{y^*}(i) = 1]. \qquad (2)
\end{aligned}$$

Note that $\underset{i\sim I}{er}(y, y^*) = 0$ when $y = y^*$. The Eq. (2) implies that,

$$\begin{aligned}
C_\epsilon &= \left\{f_y \in C : \underset{i\sim I}{\mathsf{Pr}}[f_y(i) = 1] + \underset{i\sim I}{\mathsf{Pr}}[f_{y^*}(i) = 1] \leq \epsilon\right\} \cup \{f_{y^*}\}\\
&= \left\{f_y \in C : \underset{i\sim I}{\mathsf{Pr}}[i = i^y] \leq \epsilon - \underset{i\sim I}{\mathsf{Pr}}[i = i^{y^*}]\right\} \cup \{f_{y^*}\} \qquad (3)
\end{aligned}$$

Claim 1. *If $f_y \in C_\epsilon$, then $i^y \in \mathrm{DIS}(C_\epsilon)$.*

PROOF. $\forall f_y \in C_\epsilon$ and $y \neq y^*$, we always have $f_y(i^y) = 1$ and $f_{y^*}(i^y) = 0$. Meanwhile, for $f_{y^*} \in C_\epsilon$, we have $f_{y^*}(i^{y^*}) = 1$ and $f_y(i^{y^*}) = 0\ \forall y \neq y^*$. Therefore, $\forall f_y \in C_\epsilon, i^y \in \mathrm{DIS}(C_\epsilon)$. □

Because $\mathsf{Pr}_{i\sim I}[f_y(i) = 1] = \mathsf{Pr}_{i\sim I}[i = i^y]$, the Eq. (3) implies

$$\mathrm{DIS}(C_\epsilon) = \left\{i^y \in I : \underset{i\sim I}{\mathsf{Pr}}[i = i^y] \leq \epsilon - \underset{i\sim I}{\mathsf{Pr}}[i = i^{y^*}]\right\} \cup \{i^{y^*}\}. \quad (4)$$

For primary inputs, we can assume that their logic values follow independent Bernoulli distribution with probability 0.5. Then, for any $y \in Y$, we have

$$\mathsf{Pr}_{i\sim I}[i = i^y] = \frac{1}{2^n}.$$

Therefore, if we set $\epsilon = \frac{1}{2^{n-1}}$, then, we have $C_\epsilon = C$ and based on Claim 1, $\mathrm{DIS}(C_\epsilon) = \{f_y \in C : \underset{i\sim I}{\mathsf{Pr}}[i = i^y] \leq \frac{1}{2^n}\} = I$. According to the definition of θ, we have

$$\theta = \frac{\mathsf{Pr}_{i\sim I}[\mathrm{DIS}(C_\epsilon)]}{\epsilon} = 2^{n-1}.$$

Therefore, to decide y^*, the de-camouflaging complexity increases exponentially with the circuit size, which makes it intractable for the SAT-based method to resolve the functionality of the large AND-tree structure.

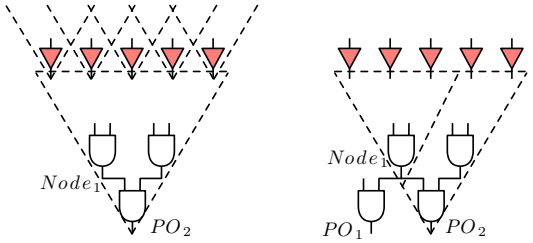


Figure 4: Two situations that can impact the security of AND-tree structure: (a) overlapped fanin cone for input pins leads to correlation; (b) extra path to primary outputs from internal node makes it possible to decompose the tree.

5.2 AND-Tree Structure in General Circuits

While a stand-alone AND-tree structure can lead to exponential increase of de-camouflaging complexity, in general circuits, the following points should be considered:

- The input pins to the AND-tree structure may not be primary inputs to the circuit. This can result in bias in the input distribution.
- There are usually more than one primary outputs in the circuit and more than one paths from each internal node to the primary outputs. This can reduce the effective size of the tree.

The two situations are illustrated in Figure 4.

As shown in Figure 4 (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 probability for different logic values are not uniform. Input bias mainly impact C_ϵ and $\mathrm{DIS}(C_\epsilon)$. According to Eq. (4), to decide $\mathrm{DIS}(C_\epsilon)$, we need to calculate the probability of each input vector, which, however, becomes intractable for large circuits. We instead propose the following methods to decide the impact of input bias:

- Because $\epsilon \geq \mathsf{Pr}_{i\in I}[f_{y^*}(i)]$ and $\mathsf{Pr}_{i\in I}[i \in \mathrm{DIS}(C_\epsilon)] \leq 1$, the de-camouflaging complexity is upper bounded by $1/\mathsf{Pr}_{i\in I}[f_{y^*}(i)]$. Therefore, we can first calculate $\mathsf{Pr}_{i\in I}[f_{y^*}(i)]$ with the method shown in [22]. If the upper bound is smaller than the predefined de-camouflaging complexity, the requirement is not satisfied.
- When the upper bound is large enough, we further adopt normalized Kullback-Leibler (KL) divergence [23], which is widely used to evaluate the distance between the two distributions. Normalized KL divergence for two discrete probability distribution, P and Q, is calculated as

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

  In our case, since Q is uniform, $KL(P|Q) = (n - E_p)/n$, where E_p is the total entropy of distribution P. The underlying assumption is that large distance between the input distribution and the ideal uniform distribution implies smaller θ. Note that the larger the KL divergence, the closer $KL(P|Q)$ approaches to 1.

To evaluate the KL divergence, sampling method can be applied to ensure accurate estimation with smaller number of samples in the input space [23]. Note that the proposed methods do not guarantee accurate estimation of de-camouflaging complexity. However, as we will show in our experimental results, the proposed methods can help evaluate the de-camouflaging complexity of the AND-tree structure. We leave in-depth research towards this direction as our future work.

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 the internal node of the tree can bypass the root of the tree and be observed at the primary output.*

For example, consider the AND-tree structure in Figure 4 (b). The internal node $Node_1$ can bypass the root of the tree PO_2 and get observed at the primary output PO_1. Therefore, an attacker is able to first de-camouflage the sub-tree structure rooted at $Node_1$. After determining the configuration of the input pins in the fanin cone of D, the attacker can 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 an AND-tree is determined by the larger one of the two sub-trees. Then, we have the following lemma on the de-camouflaging complexity of AND-tree structure.

Lemma 1. *The number of input vectors needed to de-camouflage an AND-tree structure is determined by the size of the largest non-decomposable sub-tree.*

6. OVERALL IC CAMOUFLAGING FRAMEWORK

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 Figure 5. The first step is the camouflaging cell library generation with the proposed techniques described in Section 4. Then, accurate characterization is performed to decide 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 predefined de-camouflaging complexity is not satisfied, new AND-tree structure needs to be inserted as in the fourth step. Otherwise, we simply need to camouflage the input pins with XOR-type camouflaging cells. In the sixth step, we further camouflage the fanin cone of each primary output to ensure that at least one large AND-tree structure exists in its fanin cone and that the attacker cannot resolve the functionality for each primary output. After the sixth step, a camouflaged netlist will be generated.

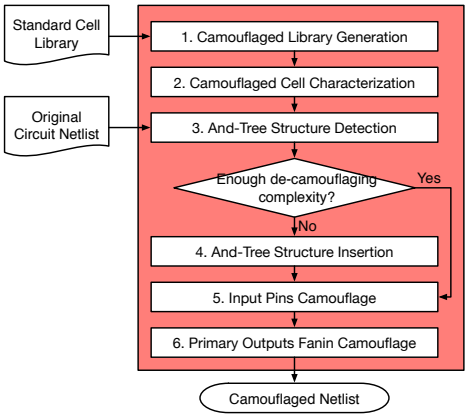


Figure 5: The proposed IC camouflaging flow.

6.1 AND-Tree Detection in Original Netlist

The AND-tree structure represents a set of circuit structures that output 0 (or 1) for only one input vector and output 1 (or 0) for all other input vectors. 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. In this section, we will describe the algorithm used for tree structure detection.

To detect the tree structure, we start from the primary inputs of the circuit and traverse the circuit graph in a topological order. For each node, we keep record of the tree rooted at the node by recording the input pins of the tree. To decide the input pins, we consider the gate type of the node and its predecessors in the circuit graph. We use the example in Figure 6 to illustrate the algorithm, the pseudo code of which is shown in Algorithm 1. For primary inputs, e.g. $Node_I$, the type of the tree rooted at the node can be treated as either AND-type or OR-type. For the internal nodes, depending on the type of the gate, there are following possibilities:

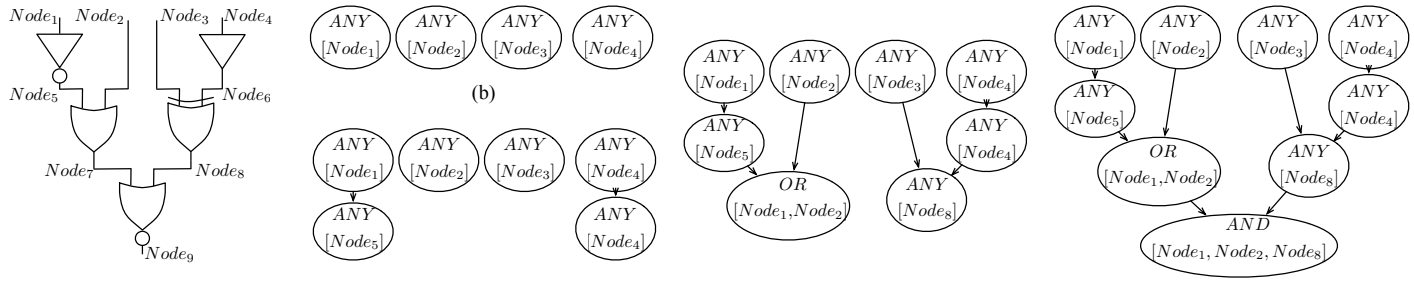


Figure 6: Detect tree structure in topological order: (a) circuit netlist; (b) (c) (d) (e) start from primary inputs to calculate tree structure in topological order.

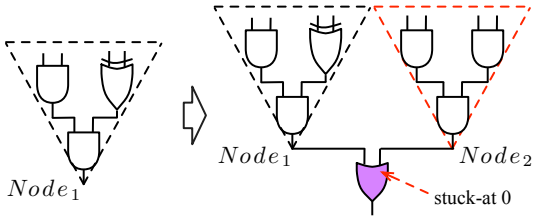


Figure 7: Insert AND-type tree structure to the circuit.

- If the gate is INV or BUF, e.g. $Node_5$, the node will have the same tree as its input. 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, e.g. $Node_7$ and $Node_9$, the tree type has to be the same as the gate type. When the predecessor's tree type is the same as the tree type of the node, larger tree structure can be formed. In this case, function ADD() is called to add the predecessor's tree structure to the node. Otherwise, only the predecessor itself can be added to the tree.
- 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, e.g. $Node_8$, no tree structure can be formed and the node itself is added to the tree.

The algorithm is shown in Algorithm 1. Note that the constraints on the output number of the predecessor is to ensure that no extra paths exist so that the tree is non-decomposable.

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 5.2. If the requirement is not satisfied, new tree structures need to be inserted.

6.2 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, including timing, power and area.
- A false interpretation of the AND-tree structure will lead to an error at circuit primary outputs.

To satisfy the first requirement, we leverage the STF-type camouflaging cells as described in Section 4. Consider an example circuit as shown in Figure 7. To insert an AND-tree structure at $Node_1$, we first insert an OR gate to $Node_1$ with the other input $Node_2$ as dummy pin. Then, an AND-tree structure is created with $Node_2$ being the root. The input pins of the AND-tree structure is connected to the primary inputs and camouflaged with XOR-type cells. Note that the added tree-structure can always be treated as non-decomposable with no input bias.

To detect the stuck-at 0 fault at $Node_2$, we again follow the same analysis as in Section 5. The logic value of $Node_2$, which is 0 in reality, can be expressed as

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

Algorithm 1 Algorithm of And-Tree Detection

```
1: procedure AndTreeDetection(G)
2:     // G is the original circuit netlist.
3:     Let {AND, ANY, OR} denote a set of tree types.
4:     U ← TopologicalSort(G)
5:     for u ∈ U do
6:         if u is primary input then
7:             u.treetype ← ANY
8:             u.treeinput ← u
9:         else
10:            if u.gatetype ∈ {BUF, INV} then
11:                u.treetype ← u.fanin.treetype
12:                u.treeinput ← u.fanin.treeinput
13:            else if u.gatetype ∈ {AND, NAND, OR, NOR} then
14:                if u.gatetype ∈ {AND, NAND} then
15:                    u.treetype ← AND
16:                else if u.gatetype ∈ {OR, NOR} then
17:                    u.treetype ← OR
18:                end if
19:                for v ∈ u.fanin do
20:                    if v.treetype = u.treetype and
       Size(v.fanout) = 1 then
21:                        u.treeinput.Add(v.treeinput)
22:                    else
23:                        u.treeinput.Add(v)
24:                    end if
25:                end for
26:            else
27:                u.treetype ← ANY
28:                u.treeinput ← u
29:            end if
30:            if u.gatetype ∈ {INV, NOR, NAND} then
31:                u.treetype ← Invert(u.treetype)
32:            end if
33:        end if
34:    end forreturn U.
35: end procedure
```

Note that $g_{n+1}(i) = 0$ when there is stuck-at-0 fault at $Node_2$, and $g_{n+1}(i) = i$ otherwise. Among all the possible configuration y, there are 2^n correct configuration with g_{n+1} interpreted as stuck-at-0 and 2^n incorrect configurations with $g_{n+1}(i) = i$. Meanwhile, any false configuration y outputs 1 for exactly one input vector and thus, is different from y^* for one input vector. Therefore, for any $y \neq y^*$, we have

$$\begin{aligned} \underset{i\sim I}{\mathrm{er}}(y, y^*) &= \underset{i\sim I}{\mathsf{Pr}}[f_y(i) \neq f_{y^*}(i)] \\ &= \underset{i\sim I}{\mathsf{Pr}}[f_y(i) = 1 \wedge f_{y^*}(i) = 0] \\ &= \underset{i\sim I}{\mathsf{Pr}}[i = i^y]. \end{aligned} \tag{5}$$

Since the inserted tree has no input bias, we have

$$\mathsf{Pr}_{i\sim I}[i = i^y] = \frac{1}{2^n}.$$

If we set $\epsilon = \frac{1}{2^n}$, then, we have $C_\epsilon = C$ and $\text{DIS}(C_\epsilon) = I$. In this case, $\theta = 2^n$ and the overall security level increases exponentially with respect to the size of the 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.

The selection of the node to insert the tree structure is also crucial to satisfy the second and third requirements. To ensure the smallest overhead and largest impact of false interpretation, besides getting rid of all the nodes on the critical paths, we propose a greedy method to select the node for tree insertion. We define an insertion score (IS) for an internal node and select the nodes with lowest scores. IS considers the node's switching probability SA, observe probability P_{ob} and number of primary outputs N_O that have not been camouflaged in its fanout cone. IS is calculated as

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

Equivalently, we look for node with lowest average cost to camouflage one primary output each time. Then, to camouflage the inserted AND-tree, we just need to insert XOR-type camouflaging cells at the input pins of the tree structure. Meanwhile, to ensure the fanin cone of all the outputs are camouflaged, new insertion node can be selected iteratively based on IS and connected to the inserted AND-tree with the STF-type cell until there is at least one tree structure in the fanin cone of each primary output.

6.3 Performance Analysis

The insertion of the AND-tree structure can lead to timing, power and area overhead. 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, it should be noted that 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, as we will show, the overhead is acceptable. For relatively large circuit, the overhead is even negligible. More importantly, as we have pointed, provided that the requirements on the non-decomposability and input bias of the inserted AND-tree are satisfied, the proposed camouflaging framework is provably secure independent of how a SAT-problem is formulated and what software/computer configuration is used.

It shall be noted that as the increase of the size of the inserted tree structure, the error rate between the correct configuration y^* and any incorrect configuration y decreases exponentially. An attacker may detect the AND-tree structure in the camouflaged netlist and randomly assign a configuration for it. We argue that this attack is not realistic from the following two aspects: 1) For some critical bits in the system, even low error rate can be unacceptable, e.g., the bits that control the "valid" signals for some security models or the bits that control the state transitions between system privilege mode and user mode; 2) the AND-tree can be camouflaged with the proposed camouflaging cells, which makes it functions as an AND-tree but appears differently and thus, further makes it impossible for the attackers to detect and resolve the inserted structure.

7. 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 [13] and tested on an eight-core 3.40 GHz Linux server with 32 GB RAM. The area overhead is calculated in terms of gate count while the timing and power overhead are evaluated with Synopsys Primetime and Primetime-PX [24]. The benchmarks are chosen from ISCAS and MCNC benchmarks [25, 26]. For the de-camouflaging algorithm, we set the runtime limit to 1.5×10^5 seconds.

We first demonstrate the effectiveness of the proposed camouflaging cell generation strategy. Because the generated camouflaging cells have negligible overhead for some specific function, we are able to insert a large amount of camouflaged cells into the netlist. Table 3 shows the de-camouflaging complexity and the time required for the SAT-based algorithm to recover the functionality of the original netlist. N/A indicates that the camouflaged netlist cannot be resolved within 1.5×10^5 seconds. As we can see, the SAT-based algorithm can still de-camouflage some small circuit benchmarks with number of gates less than 1600. Even for those large circuits that cannot be de-camouflaged completely, resolving the functionality of some primary outputs is still possible.

Table 3: Effectiveness of the proposed camouflaged cell generation strategy.

bench		# input	# output	# gate	time	# iter	Partial
ISCAS	c432	36	7	203	1.758	80	Yes
	c880	60	23	466	1.2×10^4	148	Yes
	c1908	33	25	938	N/A	N/A	No
	c2670	233	64	1490	N/A	N/A	Yes
	c3540	50	22	1741	N/A	N/A	Yes
	c5315	178	123	2608	N/A	N/A	No
MCNC	i4	192	6	536	1.9×10^3	743	Yes
	apex2	39	3	652	N/A	N/A	Yes
	ex5	8	63	1126	6.9×10^2	139	Yes
	i9	88	63	1186	2.1×10^4	81	Yes
	i7	199	67	1581	1.5×10^2	225	Yes
	k2	46	45	1906	N/A	N/A	Yes

Table 4: Existing tree structure in benchmark circuits.

bench		decom tree	non-decom tree	norm KL div
ISCAS	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

We then evaluate the effectiveness of the tree structure. We start from simple stand-alone tree structures. As in Figure 8 (a), we show the increase in the de-camouflaging complexity and time with respect to the tree size. As we can see, both the de-camouflaging time and complexity increase exponentially as we expect. To examine the impact of tree decomposability, for an AND-tree with 15 input pins, we change the size of the largest non-decomposable tree size and show the change of de-camouflaging time and complexity in Figure 8 (b). The results prove the lemma that the de-camouflaging complexity is determined by largest non-decomposable tree. The impact of input bias is also examined by changing the normalized KL divergence by adding extra circuits to the fanin cone of input pins. As we show in Figure 8 (c), as the increase of normalized KL divergence, both the de-camouflaging time and input-output patterns decreases.

Next, we camouflage the benchmark circuits with the AND-tree based method. Specifically, we pay attention to the camouflaging of small circuit since for large circuits, the proposed camouflaging cell generation strategy can already provide good resilience against the SAT-based attack. We first examine the existing non-decomposable tree structure in the original netlist and calculate the normalized KL divergence for the tree structure. We list the statistics of the largest tree structure for different benchmarks in Table 4. For most of the circuits, the existing 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 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 SAT-based method to de-camouflage. For $c2670$, original circuit functionality cannot be resolved while for $k2$, the de-camouflaging algorithm finishes within 8.5 seconds and 70 iterations. The results demonstrate the effectiveness of the proposed methods to evaluate the input bias.

Then, we insert tree structure into the benchmark circuits. We show the trade-off between the area overhead and the de-camouflaging time and complexity in Figure 9 (a) for benchmark $c880$. As we can see, the area overhead increases linearly to the size of inserted tree while

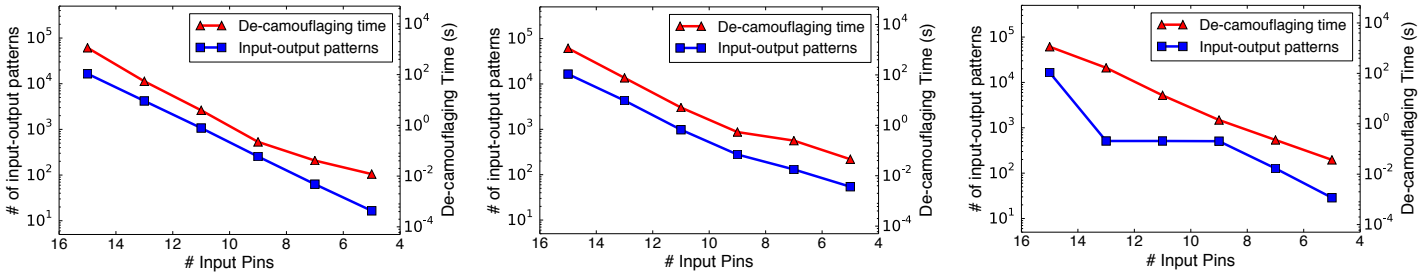
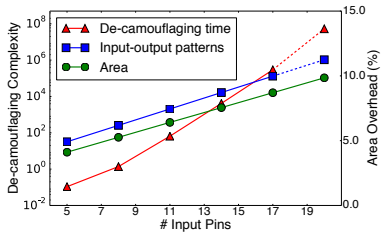


Figure 8: Effectiveness of tree structure and impact of tree decomposability and input bias.

the de-camouflaging time and complexity increases exponentially. We also insert AND-tree structure to other benchmarks and report the area, power and delay overhead. The size of the inserted tree is 20. As we can see, the main overhead comes from area and power while the impact on timing is negligible. Meanwhile, for large circuit, the area and power overhead is negligible as indicated by the last two benchmarks.



bench	# gate	area	power	timing
c432	203	16.7	14.1	0.30
c499	275	5.83	4.32	0.00
c880	466	9.85	10.8	0.06
i4	536	12.0	8.73	0.00
i7	1581	5.41	4.02	0.15
ex5	1126	4.15	3.73	0.11
ex1010	5086	0.75	1.06	0.00
des	6974	0.64	0.23	0.00
sparc_exu	27368	0.22	0.05	0.00

Figure 9: Overhead of tree based camouflaging strategy: (a) trade-off between overhead and de-camouflaging complexity (dotted lines indicate extrapolation); (b) overhead for different benchmarks.

Finally, we demonstrate the effectiveness of combining the two camouflaging techniques together. Because of the proposed camouflaging cell generation strategy, the attacker cannot decide the functionality of each gate. Combined with the AND-tree structure, better security level are provided. We examine the effectiveness by comparing the de-camouflaging complexity and time through the comparisons with the situation when only AND-tree is inserted. As shown in Figure 10, the combination of two camouflaging methods can lead to higher de-camouflaging complexity and time.

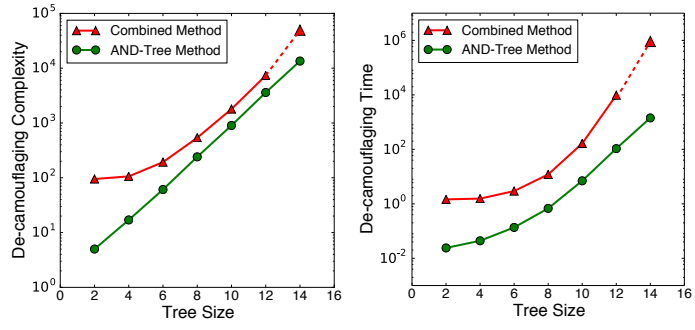


Figure 10: Effectiveness of combining the proposed two camouflaging techniques (dotted line indicate extrapolation).

8. CONCLUSION

In this paper, we propose a quantitative security criterion for de-camouflaging complexity measurements. The security criterion is formally analyzed based on the equivalence between the de-camouflaging strategy and the active learning scheme. Meanwhile, two camouflaging techniques are proposed, the low-overhead camouflaging cell library and the AND-tree structure. A provably secure camouflaging framework is then developed by combining the two techniques to achieve exponentially increasing security levels at the cost of linearly increasing overhead. Experimental results using the security criterion demonstrates that the camouflaged circuits with the proposed framework achieve high resilience against the SAT-based attack with negligible performance overhead.

9. REFERENCES

[1] Yier Jin. Introduction to hardware security. *Electronics*, 2015.

[2] Randy Torrance and Dick James. The state-of-the-art in semiconductor reverse engineering. In *DAC*, 2011.

[3] Chipwork. http://ww.chipworks.com/.

[4] Georg T Becker, Francesco Regazzoni, Christof Paar, and Wayne P Burleson. Stealthy dopant-level hardware trojans: extended version. *Journal of Cryptographic Engineering*, 4(1):19–31, 2014.

[5] Lap-Wai Chow, William M Clark Jr, and James P Baukus. Covert transformation of transistor properties as a circuit protection method, May 15 2007. US Patent 7,217,977.

[6] Lap-Wai Chow, James P Baukus, and William 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, November 13 2007. US Patent 7,294,935.

[7] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *CCS*, 2013.

[8] Shweta Malik, Georg T Becker, Christof Paar, and Wayne P Burleson. Development of a layout-level hardware obfuscation tool. In *ISVLSI*, 2015.

[9] Anirudh Iyengar and Swaroop Ghosh. Threshold voltage-defined switches for programmable gates. *arXiv preprint arXiv:1512.01581*, 2015.

[10] Yu-Wei Lee and Nur A Touba. Improving logic obfuscation via logic cone analysis. In *LATS*, 2015.

[11] Mohamed El Massad, Siddharth Garg, and Mahesh V. Tripunitara. Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes. In *NDSS*, 2015.

[12] Duo Liu, Cunxi Yu, Xiangyu Zhang, and Daniel E Holcomb. Oracle-guided incremental sat solving to reverse engineer camouflaged logic circuits. In *DATE*, 2016.

[13] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *HOST*, 2015.

[14] Sanjoy Dasgupta and John Langford. A tutorial on active learning. In *ICML*, 2009.

[15] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 1994.

[16] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007.

[17] Jeyavijayan Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Vlsi testing based security metric for ic camouflaging. In *ITC*, 2013.

[18] Roni Wiener. *An Algorithm for Learning Boolean Functions for Dynamic Power Reduction*. PhD thesis, University Of Haifa, 2007.

[19] NanGate FreePDK45 Generic Open Cell Library. http://www.si2.org/openeda.si2.org/projects/nangatelib, 2008.

[20] Mentor Graphics. Calibre verification user's manual, 2008.

[21] Predictive Technology Model ver. 2.1. http://ptm.asu.edu, 2008.

[22] Evgueni Goldberg, Mukul Prasad, and Robert Brayton. Using sat for combinational equivalence checking. In *DATE*, 2001.

[23] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1968.

[24] Synopsys PrimeTime. http://www.synopsys.com.

[25] Franc Brglez, David Bryan, and Krzysztof Koźmiński. Combinational profiles of sequential benchmark circuits. In *ISCAS*, 1989.

[26] S. Yang. Logic synthesis and optimization benchmarks user guide: Version 3.0. Technical report, MCNC Technical Report, 1991.