Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes

Mohamed El Massad
University of Waterloo
melmassa@uwaterloo.ca

Siddharth Garg
New York University
sg175@nyu.edu

Mahesh V. Tripunitara
University of Waterloo
tripunit@uwaterloo.ca

***Abstract*—Circuit camouflaging is a recently proposed defense mechanism to protect digital integrated circuits (ICs) from reverse engineering attacks by using camouflaged gates, i.e., logic gates whose functionality cannot be precisely determined by the attacker. Recent work appears to establish that an attacker requires time that is exponential in the number of camouflaged gates to reverse engineer a circuit, if the gates that are camouflaged are chosen using a procedure proposed in that work. Consequently, it appears to be the case that even by camouflaging a relatively small number of gates in the circuit, the attacker is forced to undertake several thousands of years of work. In this paper, we refute such claims. With an underlying complexity-theoretic mindset, we show that the same benchmark circuits with the camouflaged gates chosen the same way as prior work, we can decamouflage the circuit in minutes, and not years. As part of constructing our attack, we provide a precise characterization of two problems that the attacker seeks to solve to carry out his attack, and their computational complexity. A composition of solvers for the two problems is our attack procedure. We show that the two problems are co-NP-complete and NP-complete respectively, and our reduction to boolean satisfiability (SAT) and the use of off-the-shelf SAT solvers results in a highly effective attack. We also propose a new notion that we call a discriminating set of input patterns, that soundly captures the attacker's difficulty. Our extensive empirical studies reveal that the discriminating sets of inputs for realistic circuits are surprising small, thereby providing an explanation for the effectiveness of our attack. We provide additional insights by comparing the procedure of choosing gates to be camouflaged proposed in prior work to simply choosing them randomly. After presenting the results from our attack, we provide insights into the fundamental effectiveness of IC camouflaging. Our work serves as a strong caution to designers of ICs that seek security through IC camouflaging.**

I. Introduction

With increasing IC design costs, intellectual property (IP) infringement — ranging from counterfeiting to theft of core technologies and trade secrets — has become a significant concern for semiconductor design companies [23]. One of the major threats arises from reverse engineering an IC, or a targeted module in the IC [7], [10] — fully identifying its functionality by stripping it layer-by-layer (delayering) and extracting the underlying gate-level netlist. This is referred to as IC circuit extraction.

Portions of Siddharth Garg's contribution were made when he was with the Electrical and Computer Engineering Department, University of Waterloo.

NDSS '15, 8-11 February 2015, San Diego, CA, USA
Copyright 2015 Internet Society, ISBN 1-891562-38-X
http://dx.doi.org/10.14722/ndss.2015.23218

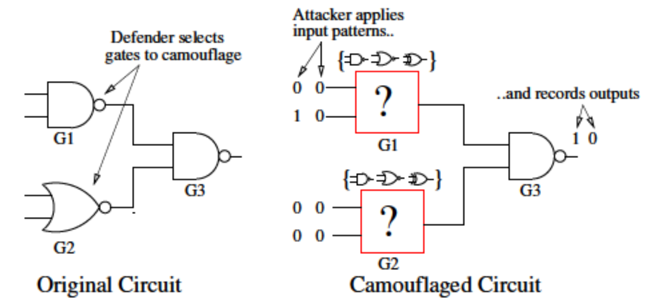


Fig. 1. Overview of IC camouflaging via an example from prior work [20]. The defender (the IC designer) selects gates to camouflage. The attacker does not know the identity of the camouflaged gates, but can apply inputs and record outputs to decamouflage the circuit.

The threat from reverse engineering seems to be particularly significant for relatively small ICs, such as those used in automative and industrial electronics, smart cards and mobile devices, that are typical of the embedded computing domain. Such devices are typically manufactured in relatively mature process technologies that are easier to delayer. Even with more recent advanced 32 nm process technologies, vendors of IC circuit extraction services such as Chipworks [7] advertise their ability to reverse engineer ICs.

Recently, Chipworks reverse engineered an Apple Lightening cable and revealed that it includes a TI chip that implements "simple security features" using "5K gates of logic" [8]. In another instance [26], they reverse engineered an IC with "embedded SRAM, ROM and EEPROM, and a hardware encryption algorithm running on-chip" where the "interface to chip was an unknown encrypted protocol." The extracted netlist was then simulated and the "outputs of simulation matched the captured system data." Although the vendor or function of this IC has not been publicly disclosed, such capabilities pose serious security challenges, particularly given the proliferation of embedded computing devices and the advent of the Internet of Things.

To protect against the threat of reverse engineering, a technique called IC camouflaging has recently been proposed and developed [20], [25]. The idea behind camouflaging is the following. In any IC, each logic gate is one of a set of gate-types, for example, XOR, NAND or NOR. A camouflaged gate is one whose gate-type cannot be determined by reverse engineering. That is, from the standpoint of an attacker that carries out reverse engineering, the gate may be any one of the possible gate-types. When inputs are applied, however, a camouflaged gate still performs the function as intended by the designer, for example, XOR, to produce the correct output. Rajendran et al. [20] provide an excellent description as to how a gate can be camouflaged in practice.

IC camouflaging technology has been adopted commercially, for example by Infineon [1], a major semiconductor vendor of embedded computing devices for mobility and security applications. The technology seems particularly attractive in the context of small ICs, e.g., those with only a few 100's or 1000's of gates, which are common in modern embedded and mobile devices. It appears that even for such small ICs, with only a small fraction of the gates camouflaged, the security attained is very high — we discuss this more below and in Section II.

Figure 1 shows an example from prior work [20] of a netlist with three logic gates, two of which are camouflaged. An attacker who wishes to decamouflage a circuit, i.e., determine the Boolean functionality (or identity) of the camouflaged gates, does so by applying input patterns and generating the corresponding output pattern. An input pattern is a bit-string that assigns a boolean value to each input pin of a circuit. The attacker then analyzes these pairs of input-output patterns to determine the identities of camouflaged gates. We call a set of input patterns that is sufficient to decamouflage a circuit a *discriminating set of inputs* for that camouflaged circuit. (See Section III-A for a more precise definition.)

The application of each new input pattern comes at a cost to the attacker. This is because applying an input pattern and observing its output pattern takes some time. The attacker's objective, therefore, is to exercise the camouflaged circuit with as few distinct input patterns as possible. That is, he seeks a discriminating set of minimum size. On the other hand, the defender seeks to ensure that the size of every discriminating set is large.

To do so, a defender camouflages as many gates as she can. However, camouflaged gates use more area, consume more power and are slower than gates that are not camouflaged. That is, camouflaging comes at a cost to the defender. This sets up a cost-security trade-off from the perspective of the defender, i.e., the defender can obtain greater security at the expense of increased circuit area, delay and power. However, ICs in the embedded computing domain are particularly sensitive to cost, particularly the chip footprint (or area) and power consumption.

One of the main contributions of prior work [20] is that this cost-security trade-off can be made to heavily favor the defender by choosing the gates to be camouflaged judiciously. For instance, given an attacker that can exercise a billion inputs a second and a benchmark circuit of more than 2400 gates, there exists a set of only 63 gates (only 2.6% of the gates in the circuit) that if camouflaged, would take the attacker "several thousands of years" to identify [20]. The work proposes techniques to discover such small sets of gates so the cost of camouflaging is low, but the difficulty for the attacker is exponential in the number of camouflaged gates. Their results suggest that if applied carefully, IC camouflaging can be an effective defense mechanism against reverse engineering attacks.

Our work We re-examine the assertions of prior work [20], [25] for realistic benchmark circuits. For those circuits, such work suggests that by camouflaging only a small, appropriately selected, set of gates, "…the attacker is forced to do brute force," which, as we mention above, for an attacker that is able to exercise the circuit with a billion input patterns a second, translates to "several thousands of years" [20].

Counter to such results, we have discovered that with the same number of gates chosen in the same manner as that work proposes for camouflaging, an attacker can find a correct completion (i.e., decamouflage the circuit) in only a few minutes, and not thousands of years. We have devised and implemented a new attack procedure for this, which is the focus of our paper.

Underlying our work is a fresh, complexity-theoretic mindset to the problem of IC decamouflaging. This mindset, in turn, suggests the attack procedure that we have designed and implemented. We examine problems that underlie two basic questions: (1) Is a given set of input patterns, I, a discriminating set for a camouflaged circuit C? (2) If so, what is a correct assignment of boolean functionalities (or identities) for each camouflaged gate in C? By iteratively calling a solver for the problem that corresponds to (1), we are able to obtain a discriminating input set. We then call a solver for the problem that corresponds to (2) to decamouflage C. As we discuss in Section III, each call to a solver for problem (1) either returns a new input pattern, or determines that the current set is sufficient to decamouflage the circuit.

Our solvers for the two problems above are based on the observation that the decision version of problem (1) is in co-$\mathbf{NP}$, and that of problem (2) is in $\mathbf{NP}$. $\mathbf{NP}$ is the class of decision problems that can be solved efficiently using a non-deterministic Turing machine; a decision problem is in co-$\mathbf{NP}$ if its complement is in $\mathbf{NP}$ [2]. (We in fact show, in addition, that the two problems are complete for their respective classes — see Section III.) Thus, there exist efficient reductions from the complement of problem (1) and from problem (2) to CNF-SAT, the problem of determining whether a boolean formula in conjunctive normal form is satisfiable. Via the reductions, therefore, we can leverage off-the-shelf SAT solvers such as Minisat [11].

Contributions We make a number of novel contributions that shed new light on the (in)effectiveness of the IC camouflaging techniques from prior work.

- We express the underlying problems an attacker solves to decamouflage a camouflaged circuit precisely and characterize their computational complexity. We establish that the problems are in co-$\mathbf{NP}$ and in $\mathbf{NP}$, respectively. In this context, we introduce the notion of a discriminating set of input patterns that serves as a sound measure of what an attacker must determine to be successful. We also identify that these problems are complete for their respective complexity classes, thereby further validating our attack procedure.

- We present an attack procedure for IC decamouflaging based on repeated calls to an off-the-shelf SAT solver. On the same benchmark circuits as prior work and using the same camouflaging techniques, we establish that circuits can be decamouflaged in minutes, rather than the years that prior work suggests it should take.
- We discuss an extensive empirical analysis that we have conducted. Our results indicate that for realistic benchmark circuits, the number of input patterns the attacker needs to apply to successfully decamouflage circuits is small (e.g., only 32 for a 5597 gate circuit with 79 camouflaged gates) and grows only linearly in the number of camouflaged gates. In addition, our SAT-based attack procedure is able to efficiently determine these inputs. Our empirical results suggest also that simply choosing the gates to be camouflaged randomly can provide as much (or as little) resistance to our attack. Our results allow us to intuit why this is the case.
- Our work leads one to ask the more basic question as to whether IC camouflaging is an effective technique at all. In this context, we address several questions, such as whether we can construct circuits that are indeed difficult for our attack to decamouflage even with only a few camouflaged gates, why the size of discriminating input set is so small for realistic circuits, and attack models under which the size of the smallest discriminating input set seems to be a meaningful measure of security.

In summary, our new IC decamouflaging attack procedure significantly raises the bar on defense mechanisms based on camouflaging. At the minimum, our work serves as a caution by raising fundamental questions that suggest that IC camouflaging may not be as effective as previously thought, at least for the kinds of circuits for which it has been touted, unless large fractions of the gates are camouflaged, which in turn comes with a prohibitive cost.

Organization The remainder of this paper is organized as follows. In Section II we give a background on the IC camouflaging technique. We describe our attack procedure in Section III. We present experimental results in Section IV. We discuss implications of our attack and possible defenses in Section V. In Section VI, we discuss related work. Section VII concludes the paper.

II. Background: IC Camouflaging

We begin by discussing the IC camouflaging defense mechanisms proposed by prior work [20]. We discuss only the details that are needed to comprehend our work; we refer the reader to that work for more details. The attack setting that work addresses is:

- An attacker first de-layers an IC to reveal an underlying camouflaged circuit, for example, the one in Figure 1. He knows the set of Boolean functions a camouflaged gate can implement — this set is easy to determine. For example, the camouflaged gates in Figure 1 implement one of the following three Boolean functions: {XOR, NAND, NOR}.
- An attacker purchases a second copy of the target IC, applies a (selected) set of input patterns to the circuit, records the corresponding output patterns, and determines the identities of the camouflaged gates. The attacker uses a specific technique (described next) to select which inputs to apply to then determine the identities of the camouflaged gates.

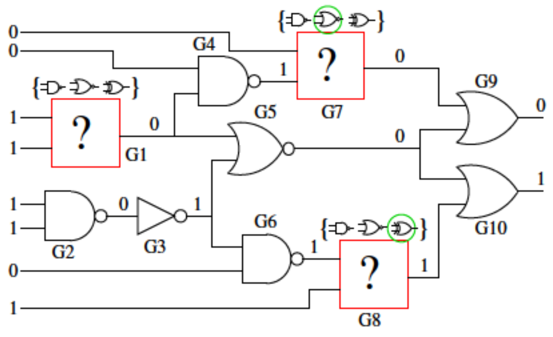


Fig. 2. Camouflaged circuit with three camouflaged gates. To determine the identity of gate G7, the attacker justifies gate G1 (sets both its inputs to logic 1) and sensitizes gate G7 by setting the input of Gate G9 to logic 0.

The technique, in prior work [20], that the attacker uses to decamouflage the circuit is as follows. The attacker targets one camouflaged gate at a time. For each target gate, he selects an input pattern such that:

- The outputs of all camouflaged gates that *interfere* with the target gate are known regardless of their identities. This is referred to as *justification*. Two gates are said to interfere with one another if both their outputs are directly or via some other gates connected to an input of a gate, or if the output of one is directly or via some other gates connected to an input of the other. In Figure 1 for example, the camouflaged gates G1 and G2 interfere with one another. In Figure 2 on the other hand, the camouflaged gates G7 and G8 do not interfere with one another.
- A change in the output of the target gate causes a change in one of the outputs of the circuit. This is referred to as *sensitization*.

The two conditions above allow the identity of the target gate to be revealed. For example, in Figure 2, assume that gate G7 is targeted first. Gate G1 interferes with G7. Thus both its inputs are set to logic 1, justifying its output to logic 0 regardless of whether its identity is NAND, NOR or XOR. Next, note that the output of gate G7 (the target gate) propagates to one of the circuit outputs (the output of gate G9). Hence the sensitization condition is met. Now the attacker can set the inputs of G7 to desired values and observe its output, thus determining its identity. Gates G8 and G1 can then be decamouflaged in a similar manner.

For the camouflaged circuit in Figure 1, however, the justification and sensitization conditions cannot be met simultaneously. Say the attacker targets G2 first. As G1 interferes with G2, the attacker tries to justify Gate G1 by setting both its inputs to logic 1. This sets the input of G3 to logic 0 and its output to logic 0 regardless of the output of G2. G2 therefore cannot be sensitized if G1 is justified, and vice versa. Gates for which the justification and sensitization conditions cannot be simultaneously met are referred to as *non-resolvable* gates.

Brute Force Attack In their work, Rajendran et al. [20] assert that to decamouflage a set of camouflaged gates that are non-resolvable and interfering (like G1 and G2 in Figure 1), the attacker has no choice but to resort to a brute force attack. In a brute force attack, the attacker does the following.

- Iterates through all possible assignments of identities to camouflaged gates, i.e., all possible decamouflaging solutions. This is exponential in the number of camouflaged gates.
- In each iteration, applies randomly generated input patterns to determine if the current candidate solution (identities for the camouflaged gates) is the correct solution, and exits when he finds such a solution.

IC Camouflaging Technique Based on the attacker's characterization above, Rajendran et al. [20] propose to select gates for camouflaging such that all selected gates are non-resolvable and interfere with one another. In other words, they seek to maximize the size of the largest clique of non-resolvable, interfering camouflaged gates. In the rest of this paper, we refer to this technique to select camouflaged gates as the "*largest-clique*" selection.

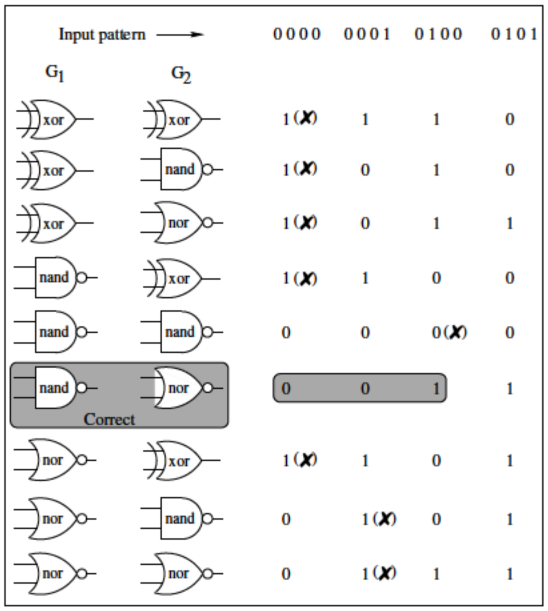
They then claim that selecting gates using the largest-clique technique raises the bar on the attacker significantly; it forces him to resort to the brute force attack that is exponential in the size of the clique. For each candidate solution (identities for all camouflaged gates), the attack must apply at least one input pattern. Thus, by camouflaging only 40 gates out of the 2400 in a circuit, i.e., 1.7% of the gates, where those 40 gates are non-resolvable and interfering, even if the attacker is able to apply 1 billion input patterns per second, it will still take him more than 1,000 years to decamouflage the circuit.

A. Counter-example to Brute Force Attack

We now present a counter-example to the assertion from prior work that an attacker requires brute-force if the camouflaged gates are non-resolvable and interfere with one another. That is, if the gates that are camouflaged are chosen in such a manner, the attacker is forced to iterate through every possible candidate solution, and try at least one input pattern for each. Our counter-example is for the circuit in Figure 1.

We establish that even though both camouflaged gates in the circuit in Figure 1 are non-resolvable and interfering, the circuit can be decamouflaged using far fewer input patterns than required by the brute force attack.

As gates G1 and G2 in Figure 1 are non-resolvable and interfering, the brute force attack requires, in the worst case, at least $3^2 = 9$ input patterns. This is because each gate can have one of three possible identities, and there are two gates. Figure 3 shows the outputs corresponding to 4 input patterns



G_1	G_2	0000	0001	0100	0101
xor	xor	1 (✗)	1	1	0
xor	nand	1 (✗)	0	1	0
xor	nor	1 (✗)	0	1	1
nand	xor	1 (✗)	1	0	0
nand	nand	0	0	0 (✗)	0
nand (Correct)	nor	0	0	1	1
nor	xor	1 (✗)	1	0	1
nor	nand	0	1 (✗)	0	1
nor	nor	0	1 (✗)	1	1

Input pattern
Correct

Fig. 3. Outputs for all 9 possible completions for the netlist in Figure 1 for 4 input patterns. These input patterns comprise a discriminating set. The correct completion is G1 = NAND, G2 = NOR, which is shown in a shaded box. Also shown in a shaded box is a set of outputs for 3 input patterns that comprise a discriminating set for the correct completion. The "X" marks show the input pattern that rules out an incorrect completion.

for all possible solutions (identities of gates G1 and G2). We make several interesting observations.

- Each decamouflaging solution results in a unique sequence of outputs. Therefore, 4 input patterns are sufficient to determine the correct decamouflaging, whereas the brute force attack suggests that 9 input patterns are required. These 4 input patterns are a discriminating set of inputs.
- Assume, as we show in the figure, that the correct decamouflaging solution is G1 = NAND and G2 = NOR. Then, only 3 input patterns are required, as the corresponding outputs are unique. The figure shows the outputs of the 3 input patterns in a shaded box. It also shows, using an "X" mark which possibilities of gate identities are eliminated by each input pattern, assuming the input patterns are applied in left-to-right order as they appear in the table. Note that the order in which the input patterns are applied is inconsequential to the fact that they comprise a discriminating set.
- Assuming the true identities are G1 = NAND and G2 = NOR, we see that a single input pattern (e.g., '0000') eliminates five out of the nine candidate solutions, i.e., more than half of the solution space is eliminated using a single input pattern. Applying the next two input patterns eliminates the other two options and reveals the correct solution.
- None of the input patterns in Figure 3 requires either G1 or G2 to be justified. Recall from our discussions in the previous section, that for this circuit, justification happens when the inputs of a camouflaged gate are set to 1.

Based on this counter-example, we conclude that the assertion in prior work that an attacker has to resort to a brute force attack if all the camouflaged gates are picked using the largest-clique technique is too conservative.

In the next section we propose a new IC decamouflaging attack based on a complexity-theoretic characterization of the IC decamouflaging problem. We show that by using our attack procedure, the attacker can decamouflage circuits within minutes that would otherwise take an impractically long time using brute force.

III. IC Decamouflaging: Our Attack

We now describe our attack. We first adopt some terminology for ease of exposition. We refer to the original circuit before camouflaging as $\mathcal{C}$. We emphasize that $\mathcal{C}$ is available to the attacker as a black-box only. That is, the only thing an attacker can do with $\mathcal{C}$ is to apply input patterns, and observe the corresponding output patterns. We refer to the camouflaged version of $\mathcal{C}$ as C. The circuit $\mathcal{C}$ has n input bits, m output bits and k camouflaged gates. For instance, in Figure 1, the circuit has $n = 4$ input bits, $m = 1$ output bits and $k = 2$ camouflaged gates.

Let L be the set of all possible gate types. For example, for the circuit in Figure 1, $L = \{\text{XOR}, \text{NAND}, \text{NOR}\}$. Let X be a function X: $[1, k] \longrightarrow L$. That is, if we arbitrarily assign the indices $1, \dots, k$ to the k camouflaged gates, the function X maps each camouflaged gate to one of the allowed gate types. We call X a completion of C, and denote the completed circuit as C_X. Note that C_X does not necessarily have the same functionality as $\mathcal{C}$. In Definition 1 below, we define a correct completion. But before that, we need to introduce some more terminology.

An input pattern i is an n bit boolean vector that assigns a boolean value to each input pin. The set of all possible input patterns is denoted as $\mathcal{I} = \{0, 1\}^n$. Any subset $I \subseteq \mathcal{I}$ is referred to as a set of input patterns. $\mathcal{C}(i)$ represents the m bit output of the black-box circuit for input pattern i. Correspondingly, $C_X(i)$ represents the m bit output of the camouflaged circuit C completed with X.

A correct completion of the camouflaged circuit C is now defined as follows.

Definition 1. ***[Correct Completion[1]]*** *A completion X is referred to as a correct completion if and only if:*

$$\forall i \in \mathcal{I}, \quad C_X(i) = \mathcal{C}(i)$$

That is, a correct completion is an assignment of gates to all the camouflaged gates such that the resultant circuit produces the same output for every input, as the black-box camouflaged circuit. The goal of the attacker is to find a correct completion. Note that our definition above accounts for the possibility that there can be more than one correct completion. If an attacker is able to arrive at any one correct completion, then he has successfully accomplished his goal which is to reverse engineer the (Boolean functionality of the) camouflaged circuit.

[1]We refer to a completion that is not correct as an incorrect completion.

A. Discriminating Set of Input Patterns

We now characterize the notion of a discriminating set of input patterns, and the computational complexity of deciding whether a given set of input patterns is discriminating. Before we do so in Definition 3 below, we define what we call the Set of Candidate Completions for a set of inputs I.

Definition 2. ***[Set of Candidate Completions]*** *The set of candidate completions, $\mathcal{P}(I)$, for a set of input patterns I comprises those completions that have the same output as the black-box circuit for all inputs in I. That is,*

$$\mathcal{P}(I) = \{X \mid \forall i \in I, C_X(i) = \mathcal{C}(i)\}.$$

Given I, a member of the set of candidate completions for it, $X \in \mathcal{P}(I)$, necessarily agrees with the black-box circuit only on the inputs in I. As a correct completion must agree with the black-box camouflaged circuit on all inputs, $\mathcal{P}(I)$ certainly contains all correct completions, and perhaps some incorrect completions. And this is the case for every $I \subseteq \mathcal{I}$. We express this via the following lemma, which is in turn used to prove Theorem 1 below.

Lemma 1. *Given a camouflaged circuit C, any $I \subseteq \mathcal{I}$, and the set of candidate completions, $\mathcal{P}(I)$ for it. $\mathcal{P}(I)$ contains all correct completions of C.*

Proof: A correct completion agrees with the camouflaged circuit on all inputs. Therefore, it agrees with the camouflaged circuit on every subset I of all inputs. ■

In Figure 3, for instance, when I consists of only one input pattern '0000,' the set $\mathcal{P}(I)$ consists of 4 candidate completions. These include the correct completion (G1=NAND, G2=NOR) but also three other incorrect completions (G1=NAND, G2=NAND; G1=NOR, G2=NAND; and G1=NOR, G2=NOR). However, when I consists of all 4 input patterns indicated in Figure 3, $\mathcal{P}(I)$ consists of only one completion, which is the correct completion. Such a set of input patterns that distinguishes the correct completion(s) from all incorrect completions is referred to as a discriminating set of inputs patterns, or simply a discriminating set. We define it as follows.

Definition 3. ***[Discriminating Set]*** *A set of input patterns $I \subseteq \mathcal{I}$ is discriminating for a camouflaged circuit C if*

$$\forall X_1, X_2 \in \mathcal{P}(I) \text{ and } \forall i \in \mathcal{I}, \qquad C_{X_1}(i) = C_{X_2}(i) \quad (1)$$

Intuition The intuition behind our characterization of a discriminating set I is the following. Suppose we have two completions X_1, X_2 that are both in the set of candidate completions $\mathcal{P}(I)$. Then, we deem I to be a discriminating set if the fact that C_{X_1} agrees with C_{X_2} on all inputs in I implies that C_{X_1} and C_{X_2} agree on all possible inputs.

We can now establish that given $\mathcal{P}(I)$ for I that is discriminating, every member of $\mathcal{P}(I)$ must be a correct completion. This is exactly the value of the notion of a discriminating set — it distinguishes a correct completion from an incorrect one.

Theorem 1. *Given $I \subseteq \mathcal{I}$ that is a discriminating set, suppose $\mathcal{P}(I) = \{X_1, \dots, X_n\}$. Then, all of $X_1, \dots, X_n$ are correct completions of C.*

Proof: Assume otherwise, for the purpose of contradiction. Then at least one of $X_1, \ldots, X_n$ is not a correct completion. That is, for some $X \in \mathcal{P}(I)$, there exists an input in the set of all inputs, $i \in \mathcal{I}$, such that $C_X(i) \neq \mathcal{C}(i)$. But as I is discriminating, we know that $C_{X_1}, \ldots, C_{X_n}$ agree on all inputs, i.e., all $i \in \mathcal{I}$. Therefore, none of $X_1, \ldots, X_n$ is a correct completion. But this contradicts Lemma 1. ■

Of course, the set of all inputs, $\mathcal{I}$, is discriminating. But the question that is most relevant to us is whether there is some $I \subset \mathcal{I}$ that is discriminating for a given camouflaged circuit. In particular, an attacker seeks an I that is small because this allows him to reverse engineer the camouflaged circuit quickly — empirically, we find that our attack procedure indeeds succeeds in doing so.

Attack methodology Our attack methodology is to first identify such a discriminating set of inputs, and then use it to correctly complete the camouflaged circuit. Towards this, we first characterize the problem of determining if a given set of input patterns, I, is discriminating.

Definition 4. *We define DISC-SET-DEC to be the following decision problem. Given the following three inputs: (i) a camouflaged circuit C, (ii) I, a set of input patterns, and (iii) the set of outputs obtained from applying input patterns in I to the black-box circuit, i.e., $\mathcal{C}(I) = \{\mathcal{C}(i_1), \ldots, \mathcal{C}(i_n)\}$, where $I = \{i_1, \ldots, i_n\}$. Is I a discriminating set for C?*

Theorem 2. *DISC-SET-DEC is in co-$\mathbf{NP}$.*

Proof: We prove the above by showing that the complement of DISC-SET-DEC, which we call NOT-DISC-SET-DEC, is in $\mathbf{NP}$. NOT-DISC-SET-DEC is the problem, given the same inputs, of determining whether I is *not* a discriminating set of input patterns. A problem is in $\mathbf{NP}$ if it has an efficiently sized proof (a certificate) for every true instance of the problem that can be verified efficiently [2]. "Efficient," in this context, means polynomial in the size of the inputs. For NOT-DISC-SET-DEC, such a certificate consists of two distinct completions X_1 and X_2, and a new input pattern $i' \notin I$ such that the following two conditions hold. First,

$$C_{X_1}(i) = C_{X_2}(i) = \mathcal{C}(i) \quad \forall i \in I,$$

Second,

$$C_{X_1}(i') \neq C_{X_2}(i').$$

X_1 and X_2 are both in the set of candidate completions, $\mathcal{P}(I)$, but do not agree with each other on the input i'. The existence of such a certificate establishes that I is not a discriminating set, because Equation (1) in Definition 3 is not satisfied. Such a certificate is linear in the size of the input, because each of X_1, X_2, i' can be encoded with size at worst the size of C. Verifying the certificate can also be done in time polynomial in the input. All we do is check: (1) that each of X_1, X_2 is a completion of C, which is linear-time in the size of C, (2) that $C_{X_1}(i) = C_{X_2}(i)$ for all $i \in I$, which can be done in time linear in the size of C for each i, for a total time of $O(|C| \cdot |I|)$, and (3) that $C_{X_1}(i) \neq C_{X_2}(i)$, which can be done in time linear in the size of C. ■

A consequence of Theorem 2 is that NOT-DISC-SET-DEC can be efficiently reduced to a problem that is complete for the complexity class $\mathbf{NP}$, such as CNF-SAT. A SAT solver can then be used to generate a certificate for NOT-DISC-SET-DEC, which also serves as a counter-example for the orginal problem DISC-SET-DEC. As we discuss below, the certificate is useful in constructing a discriminating input set I for C.

Theorem 3. *DISC-SET-DEC is co-$\mathbf{NP}$-complete.*

The proof is in the appendix, and establishes that NOT-DISC-SET-DEC is $\mathbf{NP}$-complete. Then, by definition, its complement DISC-SET-DEC is co-$\mathbf{NP}$-complete [2]. Theorem 3 is not necessary for there to exist an efficient reduction to SAT; Theorem 2 alone suffices. However, it does suggest that seeking an efficient algorithm for DISC-SET-DEC would be naive given the customary assumption that $\mathbf{P} \neq \mathbf{NP}$. Even though a SAT solver cannot fully address the intractability that is inherent in DISC-SET-DEC, it has been observed that such solvers can be surprisingly effective for large classes of input instances, particularly those that arise in practice. Thus, it makes sense for us to reduce NOT-DISC-SET-DEC to SAT, and use a SAT solver.

B. Determining a Correct Completion

Assuming that the attacker is able to find a discriminating set of input patterns I, the problem that remains for him is to find a correct completion of the camouflaged circuit C. The following decision problem captures this.

Definition 5. *We define COMPLETION-DEC to be the following decision problem. Given the following three inputs: (i) a camouflaged circuit C, (ii) I, a set of input patterns, and (iii) the output patterns obtained from applying input patterns in I on the black-box circuit, i.e., $\mathcal{C}(I)$. Does there exist a completion X such that $\forall i \in I, C_X(i) = \mathcal{C}(i)$?*

One may ask why we care to pose COMPLETION-DEC, given that the only instances of it of interest to us are those in which the camouflaged circuit C that we input has a correct completion, i.e., those in which the answer to the decision problem is always 'yes.' We address this question after characterizing the the computational complexity of COMPLETION-DEC below.

Theorem 4. *COMPLETION-DEC is in $\mathbf{NP}$.*

Proof: We need to show that for every true instance, there exists an efficiently sized certificate that can be verified efficiently. Such a certificate is simply a completion X such that the completed circuit, C_X, agrees with the black-box circuit $\mathcal{C}$ on all inputs in I. We first observe that the size of X is linear in the size of C because it is linear in the number of gates in C. To verify X, we check: (1) that X is indeed a completion of C, which can be done in time linear in the size of C, and, (2) that $C_X(i) = \mathcal{C}(i)$ for all $i \in I$, which can be done in time $O(|C| \cdot |I|)$. ■

Since COMPLETION-DEC is in $\mathbf{NP}$, a solver for COMPLETION-DEC is able to construct and provide a certificate, i.e., a completion X as discussed above. Therefore, when the input to the solver is a discriminating set of inputs, it provides exactly what we want: a correct completion.

Theorem 5. *COMPLETION-DEC is $\mathbf{NP}$-complete.*

The proof is in the appendix. As with Disc-Set-Dec, a consequence of the above theorem is that it is also unlikely that we will find an efficient algorithm for Completion-Dec, and reduction to SAT is well-motivated.

C. Constructing a Discriminating Set

As we point out in the previous section, given a solver for Completion-Dec and a discriminating set of inputs I for a camouflaged circuit C, we can determine a correct completion for C. The only issue that remains is the identification of such a discriminating set I.

We do this using a process akin to guided refinement [14]. That is, we iterate as follows given access to a solver for Not-Disc-Set-Dec. We begin with inputs $\langle C, I, O\rangle$ with $I = O = \emptyset$ to the solver. If the solver says that that input is true, this means that $\emptyset$ is not discriminating for C.

The solver also returns a certificate, $\langle X_1, X_2, i'\rangle$, as we discuss in the Proof for Theorem 2. In such a certificate, $i' \in \mathcal{I}$ is an input for which two distinct completions for C differ in their outputs. We add i' to I, i.e., set $I \leftarrow I \cup \{i'\}$, and $O \leftarrow O \cup \{\mathcal{C}(i')\}$, and again invoke the solver with the inputs $\langle C, I, O\rangle$.

That is, we "refine" our search for a discriminating set by "guiding" it by adding i' to I in the input to the solver. We repeat this till the solver says that the instance is no longer true. From the definition of Not-Disc-Set-Dec, such an I to which the above procedure converges is a discriminating set of inputs for C.

D. The Attack

Now, we can compose the solvers for Not-Disc-Set-Dec, and Completion-Dec to get a correct completion for C. The composition is that we first determine a discriminating set I by repeatedly calling Not-Disc-Set-Dec as we discuss in the previous section, and then provide that as input along with C and $\mathcal{C}(I)$ to the solver for Completion-Dec. This algorithm is expressed in the following pseudo-code.

```
$I \leftarrow \emptyset$
**while true do**
    $\langle X_1, X_2, i'\rangle \leftarrow N(C, I, \mathcal{C}(I))$
    **if** $\langle X_1, X_2, i'\rangle \neq \epsilon$ **then**
        $I \leftarrow I \cup \{i'\}$
    **else**
        break
    **end if**
**end while**
**return** $M(C, I, \mathcal{C}(I))$
```

Alg. 1: IC Decamouflaging. N is a solver for Not-Disc-Set-Dec, and M is a solver for Completion-Dec, each of which outputs a certificate if the input instance is true, and the special symbol ϵ otherwise.

In the above pseudo-code, N is a solver for Not-Disc-Set-Dec, and M is a solver for Completion-Dec. We assume that N outputs a certificate $\langle X_1, X_2, i'\rangle$ as we discuss in Section III-A if the input instance is true, and the special symbol ϵ otherwise. We assume that M outputs a certificate if it is given as input a true instance of Completion-Dec.

To construct the solver N, we efficiently reduce Not-Disc-Set-Dec to Cnf-Sat, determining whether a boolean formula in conjunctive normal form is satisfiable. Cnf-Sat is known to be **NP**-complete [12], and therefore we know that such a reduction exists. As we mention in the previous section, solvers, such as Minisat [11], exist for Cnf-Sat that are efficient for large classes of input instances.

Such a solver returns not only whether an input instance is true or false, but if it is true, it returns a certificate for it. We can use our reduction to easily map a certificate returned by Minisat to a certificate for Not-Disc-Set-Dec.

To construct the solver M, we similarly efficiently reduce Completion-Dec to Cnf-Sat, and leverage a solver for Cnf-Sat such as Minisat. We discuss our reductions from Not-Disc-Set-Dec and Completion-Dec to Cnf-Sat below, in Section III-E.

Attacker's Effort In each iteration of Algorithm 1, the attacker exercises the black-box with a new input pattern, and calls the solver N once for each such input pattern. In other words, if $|I| = D$ is the size of the discriminating set of input patterns found by Algorithm 1, the attacker would have applied exactly D input patterns to the black-box circuit and called the solver for N, $D+1$ times. In addition, the attacker has to make one call to the solver for M. If the circuit is sequential, the attacker also sets the flip-flops in the IC

E. Reductions to Cnf-Sat

As we mention in the previous section, because both Not-Disc-Set-Dec and Completion-Dec are in **NP**, there exist efficient (polynomial-time) reductions from each of those problems to Cnf-Sat. In this section, we discuss our reductions from those problems to Cnf-Sat.

Our approach is to first reduce each to Circuit-Sat, the problem of determining whether a boolean circuit is satisfiable. Circuit-Sat is also known to be **NP**-complete [12]. We then employ a well-known efficient reduction from Circuit-Sat to Cnf-Sat [28]. A reduction r from problem A to B, in this context, maps instances of A to instances of B, and has the properties that it is efficiently-computable, and an instance a of A is true if and only if the instance $r(a)$ of B is true.

We first discuss our reduction from Not-Disc-Set-Dec to Circuit-Sat. For clarity, we assume that each camouflaged gate has only one of two identities. That is, a completion X can be seen as a bit-vector $x_1, \ldots, x_k$ where the camouflaged circuit C has k camouflaged gates.

Note that there is nothing fundamental about this assumption. That is, even if a gate is allowed to have one l identities, where l is a constant, our reduction is sound with only minor changes, and remains efficient. Specifically, each x_i above, rather than being a bit, becomes a bit string $x_i = y_1 \ldots y_{\log_2(l)}$.

The boolean circuit that is the output of our reduction to Circuit-Sat has $2k + n$ inputs, where n is the number of inputs to C. We label these inputs x_{11}, x_{12}, …, x_{1k}, x_{21}, …, x_{2k}, i_1, …, i_n. Conceptually, if $X_1 = \langle x_{11}, \ldots, x_{1k}\rangle$, $X_2 = \langle x_{21}, \ldots, x_{2k}\rangle$ and $i' = \langle i_1, \ldots, i_n\rangle$, then $\langle X_1, X_2, i'\rangle$

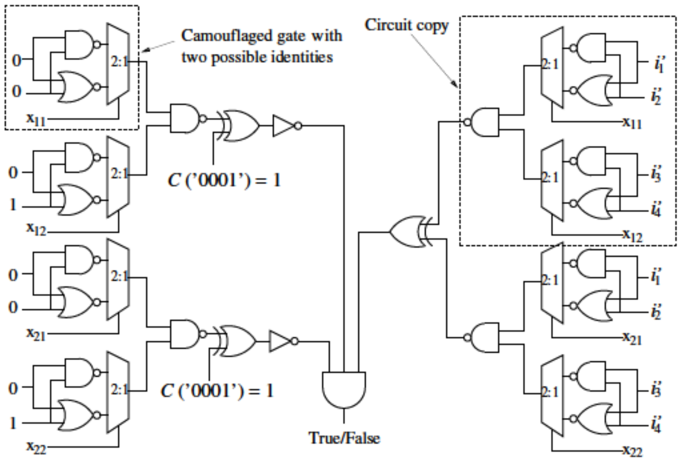


Fig. 4. Equivalent Boolean circuit for NOT-DISC-SET-DEC using the camouflaged circuit in Figure 1 as input. For clarity, we have assumed that each camouflaged gate can be one of NAND, NOR and we assume that the set of input patterns, I, has a single input pattern (—I—=1). That is I={('0001')}.

comprise exactly a certificate for a true instance of Not-Disc-Set-Dec (see Section III-A).

1) We first create two copies of C that we call C_1 and C_2. We replace each camouflaged gate j in C_1 (C_2) with 2 gates, one implementing each of the two possible functions of that camouflaged gate. The outputs are fed into a 2:1 MUX (l:1 MUX in general), controlled by x_{1j} (x_{2j}).
2) We create $|I|$ copies each of C_1 and C_2. Recall that I is part of the input instance of Not-Disc-Set-Dec. The inputs of the two r^{th} copies are fed by the r^{th} input pattern in $|I|$. The outputs of the two the r^{th} copies are compared with the corresponding output obtained from $\mathcal{C}$ using XOR gates. The outputs must match.
3) We create one more copy each of C_1 and C_2 and drive the inputs with $\langle i_1, \ldots, i_n \rangle$. The outputs of the two copies are compared using XOR gates and must differ in at least one bit.
4) We then connect the outputs of all circuits created so far to an AND gate, ensuring that any satisfying assignment for the final circuit will satisfy them all.

Figure 4 shows how the camouflaged circuit in Figure 1 is converted to a Boolean circuit that encodes the Not-Disc-Set-Dec problem for a single input pattern (i.e., when $|I| = 1$).

The reduction from Completion-Dec to Circuit-Sat is similar, except that instead of creating two copies of C for each input pattern in $|I|$, we create only a single copy. The two extra copies created in Step 3 above are also not needed.

Given the combinational circuit representations of these problems, we use the Tseitin transformation [28] to convert the circuits to Cnf-Sat instances, which are then fed to an off-the-shelf solver, Minisat [11].

F. Discussion

Our attack brings opens up new questions as to the (in)efficacy of IC camouflaging as a mechanim to secure circuits against reverse engineering attacks. An attacker's effort, in the context of IC decamouflaging, can be split up into two parts: (i) *query complexity*, i.e., the number of input patterns he has to apply to the black-box circuit, and (ii) *computational complexity*, i.e., the computational effort the attacker needs to spend in determining which input patterns to apply and how to determine the identities of the camouflaged gates from pairs of input-output patterns.

Our work asks and answers some natural questions with regards to both the query complexity and computational complexity of the attack. In particular, the query complexity of the attack is captured via the notion of a discriminating set of input patterns, since these are the *only* inputs the attacker ever applies to the black-box circuit $\mathcal{C}$. Moreover, the computational complexity question is answered through our characterization of the Disc-Set-Dec (and its complement) and Completion-Dec problems.

It is important to point out that our attack procedure does not guarantee to generate a *minimum sized* discriminating set of input patterns, and we do not yet have a complete characterization of the computational complexity of this problem. We believe that the the problem is in **PSPACE** [2], but leave a more thorough investigation of this question as future work.

IV. Experimental Results

We implement our decamouflaging algorithm using C++ in $\approx$ 1400 lines of code, and use MiniSAT [11] as the back-end SAT solver. All experiments were executed on a dual-core 2.8 GHz laptop with 6 GB of RAM and 750 GB disk space. We leave the resource limits of MiniSAT at their default values (68 years for the CPU time limit and $\approx$ 2147 TB for the memory usage limit). The camouflaged gates that we use, like the ones commercially available from Syphermedia [25], implement one of the three Boolean functions, {XOR, NAND, and NOR}.

Along with our decamouflaging attack, we also implemented two techniques to select gates for camouflaging: (1) largest clique [20] (as described in Section II); and (2) random selection, in which the camouflaged gates are picked uniformly at random. We then use our proposed attack procedure to decamouflage circuits that are camouflaged using these two techniques, and verify that we are indeed able to find the correct completion.

As benchmarks, we use circuits from the widely-used ISCAS'85 [5] and ISCAS'89 [6] combinational and sequential benchmark suites for which canonical gate-level level netlists are publicly available. These benchmark circuits range in size from small benchmarks (c432, s298, s400, s444, s713) with only 160-393 gates, to larger benchmarks (c5315, c7552, s5378, s9234, s38584) — in fact, s38584 is the largest benchmark circuit across the two benchmark suites — with between 2400-19000 gates and a wide range of functionalities. c7552 for example contains a 34-bit adder, a 34-bit magnitude comparator using another 34-bit adder, and a parity checker. s9235 and s38584 are sequential circuits derived from industrial test-chips.

Table I provides details of the benchmark circuits. That table also shows the number of camouflaged gates per circuit, which we discuss below under "our attack vs. brute force attack."

TABLE I. BENCHMARK CHARACTERISTICS, AND THE NUMBER OF CAMOUFLAGED GATES PER CIRCUIT USED TO COMPARE OUR ATTACK AGAINST THE BRUTE FORCE ATTACK FROM PRIOR WORK [20]. THE NUMBER OF CAMOUFLAGED GATES IS CHOSEN TO BE THE SAME AS IN THE PRIOR WORK.

B'mark	Inputs	Outputs	Gates	Camouflaged
c432	36	7	160	10
s298	3	6	133	6
s400	3	6	164	7
s444	3	6	181	7
s713	35	23	393	9
c5315	178	123	2406	63
c7552	207	108	3512	65
s5378	35	49	2779	56
s9234	19	22	5597	79
s38584	38	304	19234	128

TABLE II. TIME TO DECAMOUFLAGE USING OUR ATTACK AND BRUTE FORCE ATTACK ON SMALL BENCHMARK CIRCUITS CAMOUFLAGED USING LARGEST CLIQUE.

B'mark	Our Attack	Brute Force [20]
c432	0.42 s	59 μs
s298	0.13 s	729 ns
s400	0.14 s	2 μs
s444	0.2 s	2 μs
s713	0.79 s	19 μs

In addition to the ISCAS benchmark circuits (although with the exception of s38584), prior work [20] has camouflaged certain controller modules from the openSPARC core. However, canonical gate-level netlists for these are not available. Nonetheless, the s38584 benchmark has 1.5$\times$ more gates than the largest openSPARC controller module that has been considered by prior work.

Our Attack Vs. Brute Force Attack Our first goal is to demonstrate that our attack procedure effectively decamouflages circuits that are camouflaged using the largest clique technique. The existing claim is that to decamouflage these circuits, a brute force attack is necessary, and the time complexity of the brute force attack is exponential in the number of camouflaged gates [20].

In this experiment, we camouflaged the same number of gates as in prior work. These numbers are shown in the last column of Table I. We chose the gates to be camouflaged using both largest clique and random camouflaging. We discuss our results first for the small benchmark circuits, and then for the larger benchmarks.

For the five small benchmark circuits, even brute force attacks take within one second to succeed. Thus, results on these benchmarks are not very meaningful, but we nonetheless note that, our attacks are also in the sub-second range. It is worthwhile to note that for c432, we additionally performed an experiment in which we camouflaged *all* 160 gates, and were still successfully able to decamouflage the circuit using our attack. This suggests that no matter how gates in c432 are selected for camouflaging, it can always be decamouflaged by our attack.

Figure 5 shows the time taken to decamouflage the five large benchmark circuits using our attack and the estimates of how long a brute force attack would take as reported in prior work [20]. Several observations can be made:

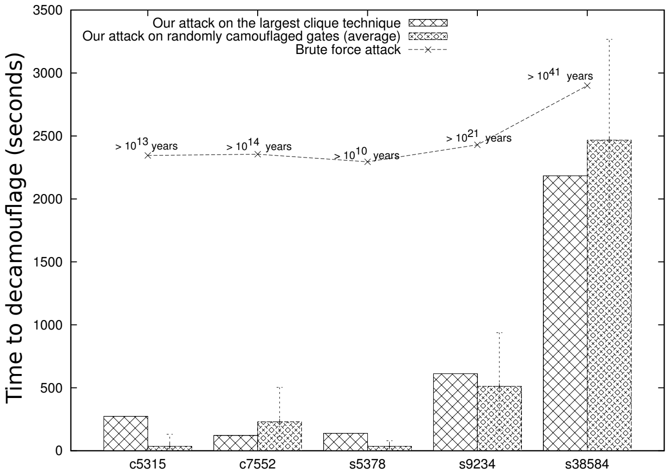


Fig. 5. Time to decamouflage (in seconds) using our attack on large benchmark circuits camouflaged using (a) the largest-clique technique, and (b) random selection of camouflaged gates (average, max, min). Also shown is the estimated time it would take (in years) for a brute force attack to succeed when largest-clique camouflaging is used.

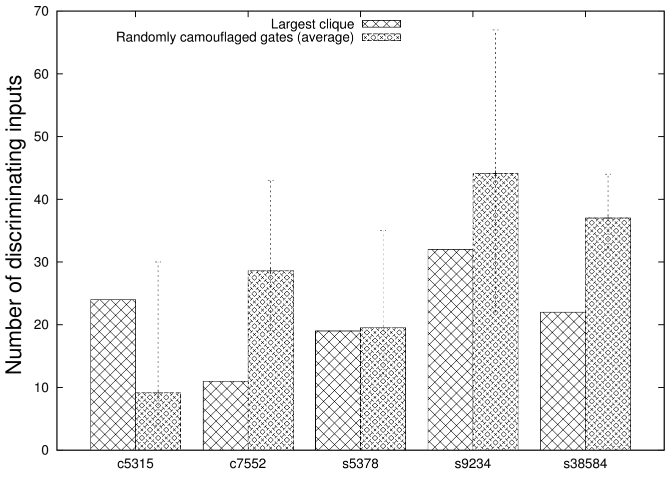


Fig. 6. Number of discriminating inputs for large benchmark circuits camouflaged using (a) the largest clique technique, and (b) random selection of camouflaged gates (average, max, min).

- Our attack is always able to successfully decamouflage circuits regardless of whether largest clique or random camouflaging is used. In all instances, our attack succeeded in less than 2500 seconds (about 40 minutes). This is in stark contrast to the estimates for a brute force attack, which range from 10^{10} to 10^{41} years.
- In addition, the largest clique camouflaging technique seems to offer no more security than random camouflaging from the standpoint of our attack. For two of the five benchmarks (c7552 and c38584), the average time to decamouflage randomly camouflaged circuits is greater than the time taken when largest clique camouflaging is used.

Why are we so successful? The success of our attack can be explained via Figure 6, which plots the size of the discriminating set of input patterns, i.e., the number of inputs that we had to apply to the black-box circuits, in our attack. Across benchmarks, our attack requires fewer than 30 input patterns to succeed when largest clique camouflaging is used. In fact, in every instance, the number of camouflaged gates is at least $2\times$ larger than the number of input patterns in the discriminating set, indicating that each input pattern decamouflages multiple gates. The relatively small size of the discriminating sets that we are able to find is a key empirical result of our work.

Our empirical results call into question the security guarantee provided by the largest clique camouflaging technique of prior work [20]. The security guarantee is: decamouflaging is impractical even if a fixed and relatively small number of gates (as few as 40 gates) are camouflaged using largest clique camouflaging in any circuit. On the contrary, our attack succeeds for every circuit we tried even with 128 camouflaged gates.

We now address the following question: how does the time taken by our attack scale with increasing number of camouflaged gates?

Impact of Increasing the Number of Camouflaged Gates Figure 7 plots the time taken for our attack to succeed on the c5315 benchmark with an increasing number of camouflaged gates that are selected using both largest clique and random camouflaging.

We find that our attack succeeds within three hours even when 350 gates in c5315 are camouflaged. To put this in perspective, camouflaging 350 gates in c5315 incurs a 92% area overhead[2], i.e., it almost doubles the circuit area and yet is not secure from our attack. On the other hand, the existing state-of-the-art would lead a defender (an IC vendor) to believe that camouflaging provides strong security guarantees with significantly lower overhead — only increasing area by 19%, for instance, by camouflaging 63 gates as prior work [20] suggests. As we have noted before, ICs that are the most susceptible to reverse engineering attacks, e.g., those in the embedded computing domain, are also the very sensitive to cost metrics such as area.

To further understand how our attack scales with the number of camouflaged gates, Figure 8 plots the discriminating set of input patterns versus the number of camouflaged gates for c5315. Several interesting observations can be made from this plot:

Linear increase in size of discriminating set: Empirically, we find that the number of input patterns required by our attack, i.e., the discriminating set, increases only linearly with the number of camouflaged gates. Fewer than 60 input patterns are required when 350 gates are camouflaged. From the defender's perspective, this is discouraging because the size of the discriminating set relates to the query complexity of the attack, i.e., the number of times the attacker has to query the black box circuit and call the MiniSAT solver (larger query complexity is better from the defender's standpoint).

Solving a system of Boolean equations: We attempt to explain the observed linear trend in query complexity by noting that the attacker's problem is that of solving a system of $m|I|$ Boolean

[2]We estimate area as the sum of the area of each gate/standard-cell before and after camouflaging. The area overhead of camouflaged gates is obtained from the data reported by [20].

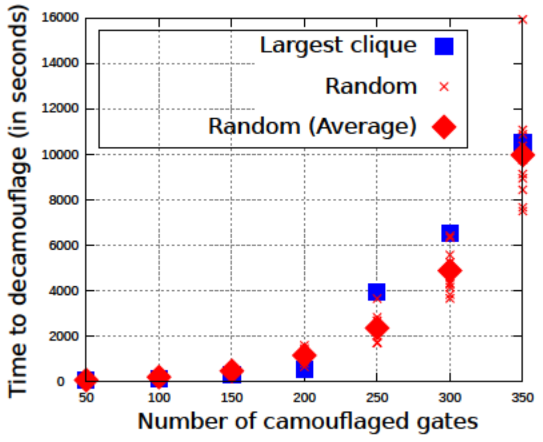


Fig. 7. Time taken to decamouflage the c5315 benchmark for increasing number of camouflaged gates selected using two camouflaging techniques: largest clique (squares) and random selection (crosses). Also plotted is time averaged over all trials for random selection (diamonds).

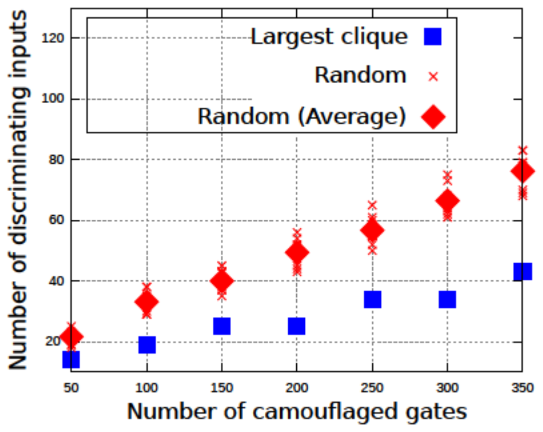


Fig. 8. Size of the discriminating set of input patterns for the c5315 benchmark, corresponding to the data in Figure 7.

equations in k Boolean variables. Recall that m is the number of output bits, k is the number of camouflaged gates and $|I|$ is the size of the discriminating set. For each input pattern, we obtain m Boolean equations, one corresponding to each output, thus giving us $m|I|$ equations in all. As an analogy, consider a system of linear equations over real-valued variables; we know that if there are as many equations as unknowns, a unique solution can be found. This suggests that we expect $|I| \propto \frac{k}{m}$. That is, for constant m, the size of the discriminating set grows linearly with the number of camouflaged gates. This is exactly the trend that we observe empirically.

Largest clique vs. random camouflaging: Random selection of camouflaged gates seems to require more input patterns than the largest clique based technique. At first this might seem surprising, but we note that random selection results in many isolated gates, each of which can require two inputs to decamouflage. Nonetheless, random camouflaging seems to generate easier SAT instances, and thus the total time required to decamouflage these circuits is typically smaller.

V. Discussion

Our work opens up new questions about whether IC camouflaging is, in fact, an effective defense mechanism to secure against reverse engineering attacks. In this section, we discuss potential defense mechanisms that could be employed to protect against our attack, although they all seem to come with significant cost to the defender. In addition we discuss avenues for further strengthening the attack procedure.

Defense Mechanisms As highlighted in Section III-F, there are two components to the attacker's effort: the query complexity, i.e., how many input patterns does the attacker have to apply to the black-box circuit, and the computational complexity, i.e., what is the computational cost of determining the input patterns to apply and, subsequently, finding a correct completion. We discuss potential defense mechanisms that try to increase the attacker's effort for each of these two components.

Increasing attacker's computational effort: The simplest way for a defender to increase the computational cost of the attacker is to increase the number of camouflaged gates. Indeed, although the performance of SAT solvers continues to improve, there are physical limits (for example, the amount of available memory) to the size of problems the SAT solver can handle. We found that when 800 or more gates are camouflaged for c5315, our attack procedure did not terminate within 24 hours. However, as we have observed before, camouflaging is accompanied with significant area, power and delay penalties — even camouflaging 350 gates doubled the area for c5315 which is prohibitive for the cost-sensitive embedded computing domain.

Another approach would be to leverage so-called Hard SAT instances, i.e., SAT instances that are known to be hard [18] because of their structure. Imagine that one of these hard SAT instances is converted to an equivalent boolean circuit, and the output of this circuit is input to a camouflaged gate (via an inverter). This is shown in Figure 9. The only way to decamouflage this gate is for the Hard SAT circuit to output a logic 1, but finding an input pattern that does so is challenging because the underlying SAT instance is hard. We have tried such an approach and observed that indeed, the SAT solver does not successfully find a solution with 24 hours. But we note that the number of added gates in this case is approximately 60000, which is, again, a very large overhead to protect a single gate.

A third approach that we found the more promising, involves increasing the number of possible identities of a camouflaged gate. The existing IC camouflaging technology allows a camouflaged gate have only one of three Boolean functionalities. However, there are 16 possible two input logic functions. A pertinent question in this context is whether it would be preferable to have fewer camouflaged gates with many possible identities, or more camouflaged gates with fewer identities. Based on some emprical evidence, we believe the answer is the former. Recall that for c5315 we were able to decamouflage up to 350 camouflaged gates (with three possible identities) in less than three hours. Instead, when we allowed each camouflaged gate to pick from one of 16 possible boolean function, even camouflaging 20 gates took

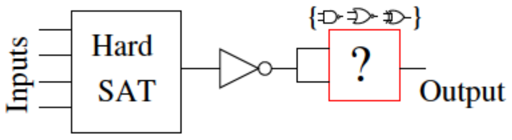


Fig. 9. A challenging decamouflaging problem.

more than three hours to decamouflage. Of course, increasing the number of possible identities of a camouflaged gate itself comes with additional area, power and delay overhead but a precise quantification of these overheads is outside the scope of this work.

Increasing the attacker's query complexity: The attacker's computational effort decreases with access to more powerful computing infrastructure with technology scaling. In this context, the defender's fall back is to increase the attacker's query complexity, i.e., the number of input patterns the attacker has to apply so as to succesfully decamouflage the circuit.

For all the benchmarks we tried. our experiments revealed that the discriminating set sizes (the query complexity) are small. However, the question is, can a defender construct a camouflaged circuit with high query complexity, for instance, exponential in the number of camouflaged gates.

To do so, the attacker can try the following strategy: ensure that each input pattern eliminates exactly one completion; in other words, for each input, all completions but for one completion will output correct values. Thus, the number of input patterns an attacker must try to decamouflage is exponential in the number of camouflaged gates.

However, this approach has at least two problems. First, it is not clear if there exists any selection of gates in the circuit which when camouflaged provide the properties mentioned above. Second, because for every input all but one completion provides correct outputs, it can be shown that any randomly picked completion of the circuit will provide correct outputs for most inputs. Nonetheless, we believe that further investigation into camouflaging techniques that guarantee high query complexity is a promising avenue for further research.

Stronger Attacks Another question that one can ask is whether there exist stronger attacks than the one that we have proposed. As before, this question can be answered both in the context of query complexity and computational complexity.

From a query complexity perspective, the problem of determining the *smallest* size discriminating set is still an open problem. However, it is important to ensure that the benefit of finding smaller sized discriminating sets is not by obviated the computational effort required to find them. Given the already relatively small size of discriminating sets we are able to find using our attack, it is unclear if finding the smallest such set would be a practically meaningful exercise, although it is certainly an interesting theoretical problem.

From the standpoint of computational complexity, this has been exactly our mindset in this paper. We have precisely characterized the computational complexity from the standpoint of an attacker in Section III. In particular, the attacker is unable to escape the intractability that is inherent in Disc-Set-Dec and Completion-Dec.

Sequential Circuits with Partial or No Scan Chains Our attack procedure assumes the target chip is equipped with a scan chain which allows the user to set and observe memory elements within the IC. If the target IC does not have a full scan design, i.e., the chip contains flip-flops that are not part of the scan chain, it is unreasonable to expect that the attacker can easily control all internal signals signals of the circuit — this is known to be even more difficult for circuits with memory elements than it is for combinational circuits. Hence, our attack procedure would need to be altered accordingly. We describe below one way of doing this and leave an investigation as to the effectiveness of the technique as a topic for future work.

We assume the all flip-flops in the chip that are not connected into chains have the capability to be forced to a certain state, set or reset, which is not necessarily the same for all flip-flops. The attacker starts by unrolling the sequential circuit one time, i.e., he removes flip-flops in the circuit that are not connected into a chain and adds, for each removed flip-flop, an input wire that drives each of the gates that were driven by the output of the removed flip-flop.

The attacker then follows Algorithm 1 to determine a discriminating set for the 1-time unrolled circuit, but now he (1) forces the Not-Disc-Set-Dec solver to return an input pattern that assigns either 0 or 1 (depending on whether the respective flip-flop has the capability to be forced into a reset/set state) to the present-state lines of the unrolled circuit (this can be done in the implementation by simply clearing the corresponding Cnf-Sat variables), and (2) he constrains candidate completions for a discriminating set to agree with each other on the next state (as well as output).

When the solver returns with a certificate, the attacker applies the primary-input part of the returned input pattern to the chip and observes the circuit output. Note that the attacker does not need to worry about setting any flip-flops that are not connected into chains. Whichever discriminating set the attacker gets at the algorithm's termination, he is guaranteed that every candidate completion for it agrees with the black-box — on both output and next-state behavior — when flip-flops are initially in the set or reset state. Beginning with this discriminating set as input to the Not-Disc-Set-Dec solver, the attacker then unrolls the circuit twice and follows the attack procedure again to get a discriminating input set for the 2-unrolled circuit.

The attacker repeats this d times, where d is the diameter of the circuit's FSM, after which he arrives at a discriminating set for the original circuit. He then uses this discriminating set to generate a correct completion that agrees with the black-box on all inputs and intitial memory states. If the attacker does not know d for certain, but can estimate it, based on knowledge of circuit's function or familiarity with hardware, then he can decide to stop after that many steps with some confidence that he has reverse engineered the circuit.

VI. Related Work

Several techniques exist to probe the inner structure of an IC in order to determine its functionality. These include scanning electron microscopy (SEM) based imaging [15] and the physical delayering that companies like Chipworks and Degate perform [7], [10]. Torrance et al. [27] provide an excellent overview of these techniques.

To protect against such attacks, several IP protection mechanisms have been proposed based on the same basic idea — to physically implement digital gates in a way so they look indistinguishable to a reverse engineer regardless of their true functionality. These mechanisms include the use of doped implants, channel stops and incomplete vias to prevent an attacker from determining that two transistors are connected, or alternatively to lead an attacker to believe two transistors are connected when they are not [3], [9]. Our decamouflaging attack would work, in principle, for any of these camouflaging techniques.

Similar to IC camouflaging, recent work [4], [17] proposes to insert programmable logic blocks (similar to those used in field programmable gate arrays or FPGAs) to hide part of the design intent. As opposed to a camouflaged gate, a programmable logic block can implement any k-input function. As we have discussed in Section V, the ability to implement any k-input Boolean function increases the difficulty of IC decamouflaging, but also comes at significantly increased cost.

With a similar intent to protect IP, key-based obfuscation techniques have been proposed. These techniques augment a circuit with an additional set of inputs (key bits) and ensure that the the circuit outputs correct values only when the correct key is applied [19], [22]. In theory, our decamouflaging attack can be used to defeat key based obfuscation as well, given access to input-output pairs from a functional circuit. In fact, a camouflaged gate can be thought of as a compound logic gate with one or more key bits as input that determine its functionality. This can be observed in Figure 4, where the input to the 2:1 MUX serves as a key bit. Nonetheless, we do not address key-based circuit obfuscation in this work.

While IC camouflaging is meant to obfuscate the design intent for an attacker in the field (i.e., after the IC has been shipped out), there have also been recent attempts to obfuscate the circuit netlist in the context of a malicious entity in an IC foundry (fabrication facility). This can accomplished via split manufacturing [13], [21], i.e., fabricating only a part of the IC in an insecure foundry. Here, the attacker makes use of structural properties of the circuit to reverse engineer the missing gates and wires, instead of the functional properties as is done for IC decamouflaging.

In an entirely different problem domain, there has been work also on oracle-guided program synthesis [14]. That work optimizes implementations of bit-manipulating programs by iteratively applying an SMT solver to find a candidate implementation that agrees with a reference implementation on a given set of inputs, queries an oracle to determine if the candidate is equivalent to the reference, and if not, uses a counter-example from the oracle to refine the candidate implementation. An important difference is that that work assumes access to an equivalence-checking oracle, whereas we do not.

We note that there has been some recent work on "reverse engineering" digital circuit netlists [16], [24], but reverse engineering is used in a very different context here. The goal of this work is to abstract a flattened netlist of gates into a set of interconnected modules (such as adders, comparators and so on), which is very different from our work.

VII. Conclusion

We have strongly refuted claims in recent work [20] regarding the effectiveness of a technique proposed in that work for IC camouflaging. Specifically, that work appears to establish that by camouflaging only a small set of gates chosen judiciously, an attacker is forced to undertake "several thousands of years" of work. In contrast, we have constructed an attack that shows that it takes the attacker only a few minutes given very modest computing resources. In constructing the attack, we have provided several additional insights into IC camouflaging as a security mechanism. We have introduced the notion of a discriminating set of inputs that soundly captures an attacker's difficulty. Our empirical assessment using the same realistic benchmark circuits that prior work has used shows that the discriminating sets are surprisingly small, thereby providing insight into why our attack is so effective. Underlying our attack procedure is a fresh, complexity-theoretic mindset, that has allowed us to intuit the computational complexity of two underlying problems for which an attacker needs solvers. We have shown how the solvers can be constructed via reductions to SAT, and the use of an off-the-shelf SAT solver. In addition, we have provided insights into the (in)effectiveness of IC camouflaging as a security mechanism. Our work serves as a strong caution to IC designers in this regard.

As future work, we plan to pursue several threads of research. As mentioned in Section VI, there are other IC obfuscation techniques proposed in literature besides camouflaging. These include key-based and programmable logic based obfuscation. We believe that our attack can be generalized to these settings as well, allowing us to investigate the security of these alternate techniques. At the same time, we would like to explore the problem of finding the minimum sized discriminating input set, both from a complexity-theoretic and practical stand-point. Finally, we are interested in further exploring the potential defense mechanisms to protect against our attack that we discussed in Section V.

References

[1] I. T. AG, "Semiconductor & system solutions - infineon technologies," August 2014. [Online]. Available: http://www.infineon.com/

[2] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. [Online]. Available: http://books.google.ca/books?id=nGvI7cOuOOQC

[3] J. P. Baukus, L. W. Chow, and W. M. Clark Jr, "Digital circuit with transistor geometry and channel stops providing camouflage against reverse engineering," Jul. 21 1998, uS Patent 5,783,846.

[4] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[5] F. Brglez, "Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN," in *Special session on ATPG and fault simulation, Proc. IEEE International Symposium on Circuits and Systems, June 1985*, 1985, pp. 663–698.

[6] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Circuits and Systems, 1989., IEEE International Symposium on*. IEEE, 1989, pp. 1929–1934.

[7] Chipworks, "Reverse Engineering Software," http://www.chipworks.com/en/technical-competitive-analysis/resources/reerse-engineering-software, last accessed May 2014.

[8] ——, "Inside the Apple Lightning Cable," http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-apple-lightning-cable/, Oct. 2012.

[9] L.-W. Chow, J. P. Baukus, and W. M. Clark Jr, "Integrated circuits protected against reverse engineering and method for fabricating the same using vias without metal terminations," Sep. 14 2004, uS Patent 6,791,191.

[10] Degate, "Reverse engineering integrated circuits with degate," http://www.degate.org/documentation/, last accessed May 2014.

[11] N. Eén and N. Sörensson, "An Extensible SAT-solver," in *Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds. Springer Berlin Heidelberg, 2004, vol. 2919, pp. 502–518. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24605-3_37

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[13] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in *Presented as part of the 22nd USENIX Security Symposium*. USENIX, 2013, pp. 495–510.

[14] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 1. IEEE, 2010, pp. 215–224.

[15] W. T. Lee, "Engineering a device for electron-beam probing," *Design & Test of Computers, IEEE*, vol. 6, no. 3, pp. 36–42, 1989.

[16] W. Li, A. Gascon, P. Subramanyan, W. Y. Tan, A. Tiwari, S. Malik, N. Shankar, and S. A. Seshia, "WordRev: Finding word-level structures in a sea of bit-level gates," in *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 67–74.

[17] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 243.

[18] D. Mitchell, B. Selman, and H. Levesque, "Hard and easy distributions of SAT problems," in *AAAI*, vol. 92. Citeseer, 1992, pp. 459–465.

[19] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[20] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 709–720. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516656

[21] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 2013, pp. 1259–1264.

[22] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1069–1074.

[23] SEMI, "Innovation is at Risk: Losses of up to $4 Billion Annually due to IP Infringement," http://www.semi.org/en/Issues/IntellectualProperty/ssLINK/P043785, last accessed May 2014.

[24] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1277–1280.

[25] SypherMedia, "Syphermedia library circuit camouflage technology," http://www.smi.tv/solutions.htm, last accessed May 2014.

[26] R. Torrance, "The state-of-the-art in Semiconductor Reverse Engineering at Chipworks," http://www.chesworkshop.org/ches2009/presentations/12_Invited_Talk_III/CHES2009_torrance.pdf, last accessed July 2014.

[27] R. Torrance and D. James, "The state-of-the-art in IC reverse engineering," in *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 363–381.

[28] G. Tseitin, "On the Complexity of Derivation in Propositional Calculus," in *Automation of Reasoning*, ser. Symbolic Computation, J. Siekmann

and G. Wrightson, Eds. Springer Berlin Heidelberg, 1983, pp. 466–483. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-81955-1_28

Appendix

As a formal language, Not-Disc-Set-Dec can be written as follows: Not-Disc-Set-Dec $= \{\langle C, IO\rangle : C$ is a camouflaged circuit, and IO is not a discriminating set for $C\}$. Now let ϕ be an instance of SAT, i.e. ϕ is a Boolean formula to be checked for satisfiability. We can consider ϕ as a Boolean circuit with a single output node in which every other node has a fan-out of 1. Add a camouflaged gate g that can implement one of $\{$NAND,NOR,XNOR$\}$ functions and use the output of ϕ to drive both of g's inputs (the true function of g is irrelevant to the reduction). Call the new circuit C. We show that ϕ is satisfiable if and only if $\langle C, \{\}\rangle \in$ Not-Disc-Set-Dec; i.e., if and only if the empty set is not a discriminating set for C. Note that for the empty set, the set of candidate completions for C consists of all three possible completions (which we get by mapping g to one of its three possible functions).

Assume ϕ is satisfiable, i.e. $\phi \in$ SAT. By definition, a satisfying assignment for ϕ sets the output of the formula to 1. Assume such a pattern is applied to C. As the output of ϕ drives both of g's inputs, g will output 0 if its true identity is NAND and 1 if its true identify is XNOR. As we have two distinct completions in the candidate set that produce different outputs for the same input pattern, the empty set cannot be a discriminating set for C, and hence $\langle C, \{\}\rangle \in$ Not-Disc-Set-Dec.

Now assume ϕ is $\langle C, \{\}\rangle \in$ Not-Disc-Set-Dec. This means that there exists an input pattern for which two of the three possible completions produce different outputs. This pattern cannot set ϕ's output to 0, as all three possible completions output 1 when ϕ's output to 0. Thus, the input pattern must set ϕ's output to 1, which means ϕ is satisfiable.

Completion-Dec can also be written as a formal language. Completion-Dec $= \{\langle C, IO\rangle : C$ is a camouflaged circuit, IO is a set of input-output pattern pairs for some circuit that has the same number of inputs and outputs as C, and there exists a completion X of C such that $\forall (i, o) \in IO, C_X(i) = o\}$. Let ϕ be an instance of SAT. We consider ϕ as a Boolean circuit. For each variable (input wire) in ϕ, we add a camouflaged gate that can implement one of $\{$NAND,NOR$\}$ functions and use the output of the gate to drive the input wire (the true functions of these camouflaged gates are, again, irrelevant to the reduction). We also create a new input wire and and drive the inputs of each of the added camouflaged gates using the wire and its negation. Call the new circuit C. Note that C has one input and one output. We show that ϕ is satisfiable if and only if $\langle C, \{(0, 1\}\rangle \in$ Completion-Dec, i.e. if and only if a completion exits for C that produces an output of 1 when 0 is applied at the input. First, note that the camouflaged gates' outputs are completely determined by their true functions (a NAND gate will output 1 and a NOR gate will output 0 regardless of what input is applied to the circuit). Now assume $\langle C, \{0, 1\}\rangle \in$ Completion-Dec. By definition, then, a completion exists that produces 1 when 0 is applied as input. In this completion, if we look at the outputs of the once-camouflaged gates, they give us an input pattern which when applied to ϕ, causes it to output 1, i.e., a satisfying assignment for ϕ. Similarly, assume a satisfying assignment exits for ϕ. This assignment will cause ϕ to output 1 if applied at ϕ's inputs. Whatever that pattern is, we can always find a completion in which this pattern is applied at ϕ's inputs when 0 is applied to C (we simply make gates corresponding to TRUE variables NANDs and those corresponding to FALSE variables NORs). Hence, $\langle C, \{(0, 1\}\rangle$ will also be a true instance of Completion-Dec.