Reverse Engineering Camouflaged Sequential Circuits Without Scan Access

Mohamed El Massad
New York University
Brooklyn, USA
me1361@nyu.edu

Siddharth Garg
New York University
Brooklyn, USA
sg175@nyu.edu

Mahesh Tripunitara
University of Waterloo
Waterloo, Canada
tripunit@uwaterloo.ca

***Abstract*—Integrated circuit (IC) camouflaging is a promising technique to protect the design of a chip from reverse engineering. However, recent work has shown that even camouflaged ICs can be reverse engineered from the observed input/output behaviour of a chip using SAT solvers. However, these so-called SAT attacks have so far targeted only camouflaged combinational circuits. For camouflaged sequential circuits, the SAT attack requires that the internal state of the circuit is controllable and observable via the scan chain. It has been implicitly assumed that restricting scan chain access increases the security of camouflaged ICs from reverse engineering attacks. In this paper, we develop a new attack methodology to decamouflage sequential circuits without scan access. Our attack uses a model checker (a more powerful reasoning tool than a SAT solver) to find a discriminating set of input *sequences*, i.e., one that is sufficient to determine the functionality of camouflaged gates. We propose several refinements, including the use of a bounded model checker, and sufficient conditions for determining when a set of input sequences is discriminating to improve the run-time and scalabilty of our attack. Our attack is able to decamouflage a large sequential benchmark circuit that implements a subset of the VIPER processor.**

I. Introduction

Vendors that provide commercial IC reverse engineering services are an increasing threat to the confidentiality of IC designs. Using chemical etching and high-resolution microscopy, vendors of reverse engineering services have reconstructed gate-level netlists of complex nanometer scale ICs [1], thus compromising the IC designer's intellectual property (IP). IP theft of this nature can negatively impact an IC designer's revenue and competitive advantage.

IC camouflaging is a promising technique to protect the designer's IP against reverse engineering attacks. IC camouflaging works by augmenting a traditional CMOS technology library with so-called *camouflaged* standard cells. A camouflaged standard cell can implement one of many Boolean logic functions, even though its layout looks the same to a reverse engineer regardless of its functionality. Several different techniques have been proposed in literature to implement camouflaged standard cells. These include the use of dummy contacts [2] and threshold-voltage dependent camouflaging [3]–[5]. Given the economic and strategic value of IC camouflaging, there has been considerable research on determining which gates in an IC to camouflage so as to maximize security [6]–[8].

This research was supported in part by NSF Grants #1527072 and #1553419, and a grant from the Semiconductor Research Corporation (SRC). Any views expressed are the authors' own and do not necessarily reflect the views of the NSF or SRC.

Recently, El Massad et al. [9] demonstrated that all existing camouflaging schemes can be broken using the so-called "SAT attack." (A similar technique to defeat logic encryption was concurrently proposed in [10].) The SAT attack assumes that the attacker has access to two functioning copies of the IC. One is reverse-engineered to reconstruct the IC's netlist barring, of course, the functionality of each camouflaged standard cell. The other copy is used to observe the IC's input/output (I/O) behavior.

The goal of the SAT attack is to find a set of inputs that are *sufficient* to deduce the functionality of the camouflaged standard cells. The attack iteratively determines new inputs that prune the attacker's search space. The attack terminates when the camouflaged standard cells have only one unique assignment (or multiple functionally equivalent assignments). New input patterns and the final completion (a completion is an assignment of identities to camouflaged gates) are determined using a SAT solver. El Massad et al. demonstrated empirically that only a small number of inputs are required to exactly decamouflage even large benchmark netlists, and that attackers can do so in the order of minutes. This work has resulted in renewed focus on stronger camouflaging schemes that are secure against SAT attacks [7], [8].

However, one criticism of the SAT attack and its subsequent enhancements [11] is that these attacks have all focused on decamouflaging combinational netlists, i.e., netlists without internal state. Real world ICs, on the other hand, are typically *sequential*, i.e., they implement finite state machines (FSMs) with internal state stored in flip-flops. The SAT attack implicitly assumes that the IC's internal state can be fully controlled and observed via scan chains, thus reducing the problem to that of reverse engineering a combinational netlist. A designer concerned about IP theft, however, can easily block user-mode access to the scan chain using a secure scan interface [12]. Thus, although the attacker still has access to the IC's primary I/Os, they *cannot* control or observe the state of internal flip-flops during IC operation. As we illustrate below, **the SAT attack does not work for ICs with internal state that cannot be accessed via scan chains**. This is a major practical limitation of the SAT attack.

Motivational Example Consider the netlist shown in Fig. 1. The netlist corresponds to the s27 circuit from the ISCAS'89 sequential benchmark suite [13]. Two gates, NAND gate U8

978-1-5386-3093-8/17/$31.00

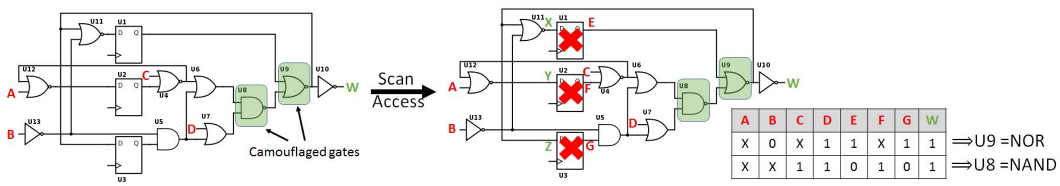


Fig. 1: ISCAS s27 sequential benchmark circuit with two camouflaged gates, G_1 and G_2.

and a NOR gate U9 have each been implemented using a camouflaged standard cell. We assume that the camouflaged standard cell can implement *either* a NAND or a NOR gate. The primary output W of the circuit depends on both the primary inputs (A, B, C, and D) and the state of the flip-flops. We will assume, without any loss of generality, that all flip-flops are initially set using a global (re)set signal.

As shown in Fig. 1, an attacker with full scan access can set the output of each flip-flop to any desired value, and can thus treat flip-flop outputs as new primary inputs (inputs E, F and G). Similarly, the attacker can scan out the the input of each flip-flop, and can thus treat flip-flop inputs as new primary outputs (outputs X, Y and Z). The SAT attack, or in fact even the less powerful logic testing based attack proposed by Rajendran et al. [6], can be used to decamouflage the resulting combinational netlist. Fig. 1 shows two inputs that are sufficient to decamouflage gates U8 and U9; the first input reveals the identity of U8 while the second reveals the identity of U9.

Now consider an attacker *without* scan access. The attacker can no longer control flip-flop outputs or observe flip-flip inputs. To apply the SAT attack, an attacker can treat the sequential circuit as a single-stage combinational circuit by repeatedly resetting the flip-flops, applying primary inputs A, B, C and D, and observing the primary output W. Unfortunately, this strategy does not work. As shown in Fig. 2, primary output W equals 0 regardless of the identity of camouflaged gates U8 and U9.

Yet, as shown in Fig. 2, applying a *sequence* of two inputs recovers the correct identities of the two camouflaged gates. That is, the output of the FSM after the second input is applied is 1 if and only if gate U8 is a NAND and gate U9 is a NOR. In general, we note that an attacker might require not only one but multiple input sequences to reverse engineer camouflaged sequential circuits without scan access. Finding a set of input sequences that is sufficient to decamouflage the netlist is the goal of our attack.

In general, removing scan access from a camouflaged sequential circuit makes the reverse engineering problem more challenging for several reasons. For one, an attacker with scan access can arbitrarily set the state of the sequential circuit to states that help discriminate the identities of camouflaged gates (as in the example above). On the other hand, an attacker without scan access must apply a sequence of inputs that lead the sequential circuit to the desired state. However, the input sequence itself depends on the identities of camouflaged gates. Second, an attacker without scan access only observes the primary outputs and must *infer* the next state outputs computationally. Given these challenges, a natural question that arises is the following: does restricting scan chain access for camouflaged sequential circuits enhance their security against reverse engineering attacks? Our new attack seeks to answer this question both foundationally and empirically.

Our Contributions In this paper, we make the following novel contributions.

- We introduce the first attack methodology to decamouflage *sequential* circuits without access to internal state of the flip-flops. Our attack searches iteratively for *input sequences*; each new input sequence eliminates one or more remaining decamouflaging solutions till only correct completions/solutions are remaining.
- We characterize the computational complexity of two important sub-problems in our attack procedure: is a given set of input sequences sufficient to decamouflage the netlist (DISC-SET-SEQ-DEC), and finding a completion that is consistent with a set of input sequences (COMPLETION-DEC). We show that the former problem is in **PSPACE**, while the latter is **NP**. Consequently, our attack uses a *model checker* to find new input sequences, to decide when to terminate and to identify correct completions.
- We propose a practical attack methodology that utilizes a *bounded* model checker and *sufficient* conditions for the DISC-SET-SEQ-DEC problem to reduce the run-time of the attack.
- Our experimental results speak to the strength of our attack; we are able to decamouflage a sequential benchmark that represents a part of the *VIPER* processor netlist with more than 5000 gates in a matter of hours. For benchmarks which our attack fails to fully decamouflage, we still correctly decamouflage up to 30 out of 32 camouflaged gates.

II. RELATED WORK

Techniques for extracting the underlying netlist of integrated circuits via chemical etching, delayering and scanning electron microscopy (SEM) are offered by companies like Chipworks [1] and Degate [14] as part of their commercial reverse-engineering services. These companies also develop and offer software tools to aid in the process of circuit extraction. Torrence et al. [15] provide a detailed overview of the IC reverse engineering process.

Camouflaging technology aims to protect against the misuse of these IC reverse-engineering techniques for piracy and

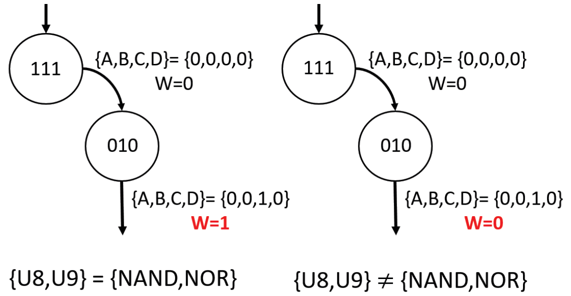


Fig. 2: The two states in the FSMs corresponding to the correct decamouflaging solution and incorrect solutions, respectively, for the circuit in Fig. 1.

copyright infringement. Several proposals have been made, both in academia and industry, for implementing camouflaged cells for use in ASIC processes. These include dummy-contact-based camouflaged cells [2], [8], [16] as well as threshold-voltage dependent gates [5], [8].

Because camouflaged standard cells incur area, delay and power overheads, recent research has focused on determining which gates to camouflage to maximize security. However, all of the work has considered camouflaging only combinational circuits, or equivalently, assumed sequential circuits in which the attacker has scan access. Rajendran et al. [6] showed that randomly selecting gates to camouflage is vulnerable to VLSI testing based attacks, and proposed a new selection scheme that tries to maximize the number of non-resolvable gates. However, this scheme was broken by El Massad [9] and Subramanyan et al.'s SAT attacks. In response to these attacks, [7] and [8] concurrently developed SAT-attack resilient schemes that try to ensure that discriminating sets are exponentially sized. However, as acknowledged by the authors, these schemes also come with a fundamental trade-off: the output corruptability (or error rate) of these schemes is low; that is, incorrect completions agree with correct completions on almost all inputs.

In this paper, we seek to analyze the security of camouflaging schemes that purport to defend against SAT attacks in a different way, i.e., by removing access to scan chains (instead of reducing output corruptability/error rate as in the schemes proposed by [7] and [8]). While our new attack is successful on a range of benchmarks, we also find that there are some benchmarks whose security is enhanced by removing scan access.

III. Attack Procedure

In this section, we describe our attack procedure. We first precisely describe the attack objective and introduce some notation that aids our exposition. Then, we define two computational problems that form the foundation of our procedure and characterize their computational complexity. Finally, we describe how the procedure works, and the practical choices we made while designing the attack.

A. Problem Formulation

As noted before, we assume that the attacker obtains two copies of the IC. The attacker use the first copy as a black box and exercises it with inputs. Let $\mathcal{C}$ represent the black-box IC. The attacker uses chemical etching and imaging to extract the netlist of the second IC — let C refer to the extracted netlist. A subset of gates in C are camouflaged.

Let m be the number of primary inputs, n be the number of primary outputs, k be the number of camouflaged gates and l be the number of flip-flops (bits of internal state) in $\mathcal{C}$ (and C). For instance, in Fig. 1, $m = 4$, $n = 1$, $k = 2$ and $l = 3$. We assume, without loss of generality, that each camouflaged gate in the IC implements one of t Boolean functions. In the example in Fig. 1, for instance, $t = 2$ because each camouflaged gate is either a NAND or a NOR. A *completion* $X : \{1, 2, \ldots, k\} \rightarrow \{1, 2, \ldots, t\}$ assigns a Boolean function to each camouflaged gate in C. Given a completion X of C, we denote the completed circuit by C_X.

Now consider a sequence of inputs $I = (i_0, i_1, \ldots, i_{p-1})$ of length p applied to $\mathcal{C}$ starting from the initial reset state, s_0. Here, i_0 is the input applied in the first time step, i_1 is the input applied in the second time step, and so on. Let $\mathcal{C}(I) = (\wr_0, \wr_1, \ldots, \wr_p)$ denote the sequence of outputs that $\mathcal{C}$ produces for input sequence I.

Similarly, for a completion X of C, let $C_X(I) = (o_0, o_1, \ldots, o_p)$ denote the sequence of outputs produced by circuit C_X for sequence I, assuming as above that inputs are applied starting from the initial reset state, s_0.

Let $\mathcal{I}$ denote the set of all input sequences of length 2^l, which we refer to as the *universal set of input sequences*. Given $\mathcal{C}$ and C, the goal of our attack is to find a completion X^* such that

$$\forall I \in \mathcal{I}, \quad C_{X^*}(I) = \mathcal{C}(I), \tag{1}$$

that is, we seek an assignment X^* of Boolean functionalities to camouflaged gates such that the outputs of the completed netlist, C_{X^*} agree with outputs of the black-box circuit $\mathcal{C}$ on *all* input sequences of length 2^l. Equivalently, we seek an X^* such that C_{X^*} is sequentially equivalent to $\mathcal{C}$. We call such a completion a *correct completion*. We note that a correct completion is not necessarily unique. However, as we find any correct completion, it means we have successfully reverse-engineered (the Boolean functionality of) $\mathcal{C}$.

B. Foundations of Our Attack

In this section, we define two decision problems that form the foundation of our attack procedure. We start by introducing the notion of a *discriminating set of input sequences*, which generalizes the notion of a discriminating set of inputs that was introduced by El Massad et al. [9] in the context of decamouflaging combinational circuits.

Definition 1. *A set of input sequences* $\mathbf{I} = \{I_0, I_1, \ldots, I_n\}$ *is called* discriminating *if every completion* X *that satisfies*

$$C_X(I_j) = \mathcal{C}(I_j), \quad \forall j \in \{1, \ldots, n\}$$

is a correct completion.

We now articulate the problem of deciding whether a set of input sequences is discriminating for a sequential circuit, and identify the computational complexity class to which the problem belongs.

Definition 2. *We define* DISC-SET-SEQ-DEC *to be the following decision problem. Given the following three inputs: (i) a camouflaged circuit C, (ii) $\mathbf{I}$, a set of input sequences, and (iii) the set of outputs obtained from applying input sequences in $\mathbf{I}$ to the black-box circuit, each time starting from state s_0, i.e., $\mathcal{C}(\mathbf{I}) = \{\mathcal{C}(I_1), \ldots, \mathcal{C}(I_n)\}$, where $\mathbf{I} = \{I_1, \ldots, I_n\}$. Is I a discriminating set for C?*

Theorem 1. DISC-SET-SEQ-DEC *is in* $\mathbf{PSPACE}$.

The proof for the above theorem is in the appendix. We remark on the significance of this result below.

Remark. The fact that DISC-SET-SEQ-DEC is in $\mathbf{PSPACE}$ suggests that a viable strategy for tackling the problem is to reduce it to model checking which is known to be complete for $\mathbf{PSPACE}$. This is exactly what we do in our attack procedure, as we describe in Section III-C.

Remark. The corresponding decision problem for camouflaged combinational circuits, i.e., whether a given set of inputs (note, not input sequences) is discriminating, was found to be in the complexity class co-$\mathbf{NP}$, which is contained in $\mathbf{PSPACE}$.

We next define a problem that captures the computational task of finding a correct completion given a discriminating set of input sequences for a camouflaged circuit.

Definition 3. *We define* COMPLETION-DEC *to be the following decision problem. Given the following three inputs: (i) a camouflaged circuit C, (ii) $\mathbf{I}$, a set of input sequences, and (iii) the outputs obtained from applying inputs in $\mathbf{I}$ on the black-box circuit, i.e., $\mathcal{C}(\mathbf{I})$. Does there exist a completion X such that $\forall I \in \mathbf{I}, C_X(i) = \mathcal{C}(I)$?*

Theorem 2. COMPLETION-DEC *is in* $\mathbf{NP}$.

Proof: A certificate for COMPLETION-DEC is a completion X such that C_X agrees with the black-box circuit on input sequences in $\mathbf{I}$. X is polynomially sized in the input to COMPLETION-DEC (X can be encoded using $k \log_2(t)$ bits). Verifying that C_X agrees with the black box on sequences in $\mathbf{I}$ can be done in time $O(|\mathbf{I}||C|)$ which is polynomial in the size of the input. ■

We note here that the relevance of COMPLETION-DEC is that it yields a correct completion (as a certificate) when the input to the problem is a discriminating set of input sequences. The theorem implies that one can reduce COMPLETION-DEC to a problem that is complete for $\mathbf{NP}$, in particular, to CNF-SAT, and then use an off-the-shelf SAT solver for the resulting SAT instance. Our reduction from DISC-SET-SEQ-DEC to the model checking problem however facilitates an alternative and more convenient approach for COMPLETION-DEC, as we describe below in Section III-C.

C. Practical Attack Procedure

Our attack proceeds iteratively: we maintain a set of input sequences $\mathbf{I}$ that is initially empty. In each iteration, we add one (or more) new input sequences to the set. We stop when we determine that our set of input sequences is discriminating.

However, a naive implementation of this procedure would make a call to an *unbounded* model checker in each iteration, that is, a model checker that searches for input sequences of arbitrary length. Unfortunately, calls to an unbounded model checker can be time consuming. Instead, we add new input sequences of *bounded* length (using a bounded model checker), and increase the bound only when needed. Further, instead of directly calling an unbounded model checker to decide if the current set of input sequences is discriminating (the termination condition for our procedure), we first check two simpler *sufficient* conditions for termination. Our refined attack procedure is described below.

1) Finding New Input Sequences: To find new input sequences to add to our set, we construct a solver M_{BMC} for DISC-SET-SEQ-DEC by reducing DISC-SET-SEQ-DEC to a bounded model checking problem. Given C, $\mathbf{I}$, and $\mathcal{C}(\mathbf{I})$ as defined in Definition 2, and a parameter b that specifies the model checking bound, M_{BMC} returns `true` if and only if for any two completions X_1 and X_2, and every input sequence $I \in \{0,1\}^b$, the following implication is true:

$$C_{X_1}(\mathbf{I}) = C_{X_2}(\mathbf{I}) = \mathcal{C}(\mathbf{I}) \implies C_{X_1}(I) = C_{X_2}(I).$$

Note that if M_{BMC} returns `true`, it does not necessarily mean that the given set $\mathbf{I}$ is indeed a discriminating set of input sequences for our camouflaged circuit. It means only that the solver is unable to find a new input sequence of length at most b that helps to eliminate any of the remaining completions.

If M_{BMC} returns `false` on the other hand, it returns, two completions X_1 and X_2 and a new input sequence $\tilde{I}$ of length at most b such that $C_{X_1}(I) = C_{X_2}(I) = \mathcal{C}(I)$ for all $I \in \mathbf{I}$, but $C_{X_1}(\tilde{I})) \neq C_{X_2}(\tilde{I})$, i.e., C_{X_1} and C_{X_2} agree with the black-box circuit on input sequences in $\mathbf{I}$, but produce different outputs for input sequence $\tilde{I}$.

We call M_{BMC} at every iteration in our algorithm; passing it our camouflaged circuit, our current set of input sequences, and the output of the black-box circuit for each sequence in the set. If M_{BMC} returns with a sequence $\tilde{I}$, we add $\tilde{I}$ to our set of input sequences.

2) Termination Criteria: As we stated previously, our M_{BMC} solver cannot decide whether a set of input sequences $\mathbf{I}$ is discriminating. For this, we need to call an unbounded model checker. However, before calling the unbounded model checker, we check for two conditions that are sufficient to show that $\mathbf{I}$ is discriminating. The intuition behind performing these checks before calling an unbounded model checker is that we expect them to be computationally less time consuming.

1) **Unique Completion (UC):** We check to see if there is only one remaining completion that agrees with the black-box circuit on the current set of input sequences. Specifically, we try to find two distinct completions that agree with the black-box circuit on the current set of input sequences, i.e. we try to find two completions X_1 and X_2 such that

$$C_{X_1}(\mathbf{I}) = C_{X_2}(\mathbf{I}) = \mathcal{C}(\mathbf{I}) \quad \text{and} \quad \mathrm{X}_1 \neq \mathrm{X}_2$$

Proposition 1. *If no such X_1 and X_2 exist, then* $\mathbf{I}$ *is a discriminating set for* C.

Proof: Follows immediately from the definition of a discriminating set of input sequences. ∎

2) **Combinational Equivalence (CE)** Next, we check whether all completions that agree with the black-box circuit on the current set of input sequences are *combinationally* equivalent with respect to both output and next state function. That is, if we denote by $\tilde{C}_X(i,s) = (o, s')$ the pair of output o and next state s' of completed circuit C_X when in state s, and at the application of input i, we ask whether

$$C_{X_1}(\mathbf{I}) = C_{X_2}(\mathbf{I}) = \mathcal{C}(\mathbf{I}) \implies \tilde{C}_{X_1}(i,s) = \tilde{C}_{X_2}(i,s) \quad \forall i \in \{0,1\}^m, s \in \{0,1\}^l. \tag{2}$$

Proposition 2. *If Condition (2) holds for set* $\mathbf{I}$ *and camouflaged circuit* C*, then* $\mathbf{I}$ *is a discriminating set for* C.

Proof: Since $\tilde{C}_{X_1}(i,s) = \tilde{C}_{X_2}(i,s)$ for all possible inputs i and all possible states s, it follows that $C_{X_1}(I) = C_{X_2}(I)$ for every input sequence I. Thus, $\mathbf{I}$ is discriminating for C. ∎

We check Condition (2) by calling a solver CE that we construct. We give it as input: (i) the camouflaged circuit C, (ii) our current set of input sequences $\mathbf{I}$, and (iii) the outputs $\mathcal{C}(\mathbf{I})$ of the black-box circuit for input sequences in $\mathbf{I}$. If the solver returns `true`, we terminate.

3) **Unbounded Model Check (UMC)** If the preceding two checks fail, we finally call a solver M_{UMC} that we have constructed. The 'U' is for "Unbounded." We give it as input: (i) our camouflaged circuit C, (ii) the current set of input sequences $\mathbf{I}$, and, (iii) the outputs $\mathcal{C}(\mathbf{I})$ of the black-box circuit. The solver M_{UMC}, unlike M_{BMC}, returns `true` if and only if $\mathbf{I}$ is a discriminating set for C.

3) Finding a Correct Completion: Since the COMPLETION-DEC problem is in **NP**, it can be reduced to a CNF-SAT instance and solved using a SAT solver. However, we note that our model checking based solver for DISC-SET-SEQ-DEC has the property that if $\mathbf{I}$ is a discriminating set for C, then a correct completion is encoded in *every* initial state of the model. As such, we do not need to call an external SAT solver to find a correct completion. We simply ask our model checker to choose any element from the set of initial states. We denote this procedure for the COMPLETION-DEC problem as solver N. N takes as input the camouflaged circuit C, the current set of input sequences $\mathbf{I}$, the outputs $\mathcal{C}(\mathbf{I})$ of the black-box circuit, and outputs a correct completion X^* if $\mathbf{I}$ is discriminating.

4) Complete Algorithm: Our complete algorithm is expressed as Algorithm 1. In the algorithm, we start with an initial bound $b = 0$ for our BMC solver, i.e., M_{BMC}. At every iteration, if we determine that we need to continue, i.e., all three checks described in the previous section fail, we increase the value of b by a fixed increment *bmc_inc*, and we continue until at least one of the three checks succeeds. At the end, once we arrive at a discriminating set of input sequences for our camouflaged circuit, we employ the technique described in Section III-C3 to find a correct completion for our circuit.

Algorithm 1: Scan-Chain-Free Decamouflaging.

```
I ← ∅, b ← 0
while true do
    b ← b + bmc_incr
    ⟨X1, X2, Ĩ⟩ ← M_BMC(C, I, C(I), b)
    if ⟨X1, X2, Ĩ⟩ ≠ true then I ← I ∪ Ĩ
    else if UC(C, I, C(I)) or CE(C, I, C(I)) or
        M_UMC(C, I, C(I)) then break
return N(C, I, C(I))
```

D. Implementation of Solvers

We now describe how the solvers referred to in the previous section are implemented using a model checker. Model checkers take as input a model of an FSM represented as a Kripke structure. We begin by describing an FSM model for the camouflaged circuit C, and the Kripke structure that we use as input for the model checker.

Corresponding to each completion X of circuit C is an FSM $(\mathcal{I}, \mathcal{O}, \mathcal{S}, s_0, \sigma_x, \omega_x)$ where:

- $\mathcal{I} = \{0,1\}^m$ is the input alphabet of the FSM, the set of all possible inputs,
- $\mathcal{O} = \{0,1\}^n$ is the output alphabet of the FSM, the set of all possible outputs,
- $\mathcal{S} = \{0,1\}^l$ is the set of states of the FSM, a set of all l-bit Boolean vectors,
- $s_0 \in S$, the initial state of the black-box circuit,
- σ_X is the state-transition function, $\sigma_X : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$,
- ω_X is the output function, $\omega : \mathcal{S} \times \mathcal{I} \to \mathcal{O}$.

For an input sequence $I = (i_0, i_1, \ldots)$, we can write $C_X(I)$ in terms of σ_X and ω_X as follows:

$$C_X(I) = (\omega_X(i_0, s_0), \omega_X(i_1, \sigma_X(s_0, i_0)), \ldots)$$

Our M_{UMC} solver takes an instance $\langle C, \mathbf{I}, \mathcal{C}(\mathbf{I})\rangle$ of DISC-SET-SEQ-DEC and transforms it into a model checking instance as follows. Let $\mathcal{X} = \{0, \ldots, t-1\}^k$ be the set of all possible completions. We define a set of atomic propositions AP to be a singleton, consisting of an atomic proposition `equiv`, the semantics of which we clarify below. We build a Kripke structure M over AP from the tuple $\langle C, \mathbf{I}, \mathcal{C}(\mathbf{I})\rangle$ that is input to the DISC-SET-SEQ-DEC problem. The characteristics of M are as follows:

- The set of initial states of M is $\mathcal{S} \times \mathcal{X} \times \mathcal{S} \times \mathcal{X} \times \mathcal{I}$, i.e., any state in M is a 4-tuple of the form (s_1, X_1, s_2, X_2, i) where s_1 and s_2 are l-bit vectors, X_1 and X_2 are completions, and i is a m-bit vector representing an input.
- The set of initial states of M is $\{(s_0, X_1, s_0, X_2, i) : X_1, X_2 \in \mathcal{X}, i \in \mathcal{I}$ and $C_{X_1}(\mathbf{I}) = C_{X_2}(\mathbf{I}) = \mathcal{C}(\mathbf{I})\}$.
- M's transition relation, $\mathcal{R}$ is defined as follows: $\mathcal{R} = ((s_1, X_1, s_2, X_2, i), (s_1', X_1, s_2', X_2, i')) : s_1, s_2, s_1', s_2' \in \mathcal{S}, \quad X_1, X_2 \in \mathcal{X}, \quad i, i' \in$

$\mathcal{I}$ such that $\sigma_{X_1}(i, s_1) = s'_1$ and $\sigma_{X_2}(i, s_2) = s'_2\}$. That is, the system can transition from a state (s_1, X_1, s_2, X_2, i) to a state $(s'_1, X_1, s'_2, X_2, i')$ if and only if when C_{X_1} and C_{X_2} are in states s_1 and s_2, respectively, and we apply input i to both circuits, C_{X_1} and C_{X_1} transition to states s'_1 and s'_2 respectively. The transition relation also requires that X_1 and X_2 retain their initial values throughout the evolution of the model.

- M's labeling function, $\mathcal{L}$, is defined as follows: $L(s_1, X_1, s_2, X_2, i) = \{\texttt{equiv}\}$ if $\omega_{X_1}(s_1, i) = \omega_{X_2}(s_2, i)$, otherwise $L(s_1, X_1, s_2, X_2, i) = \emptyset$, i.e., the proposition equiv is true in a state (s_1, X_1, s_2, X_2, i) if and only if when C_{X_1} and C_{X_2} are in states s_1 and s_2, respectively, and we apply input i to both circuits, C_{X_1} and C_{X_1} produce the same output.

We have the following proposition:

Proposition 3. *For a given Disc-Set-Seq-Dec instance $\langle C, \mathbf{I}, \mathcal{C}(\mathbf{I})\rangle$, if the structure M constructed as above satisfies the specification $\mathbf{G}$ equiv, that is, for every $i \in \mathcal{I}$ and every $X_1, X_2 \in \mathcal{X}$, $M, (s_0, X_1, s_0, X_2, i) \models \mathbf{G}$ equiv, then $\langle C, \mathbf{I}, \mathcal{C}(\mathbf{I})\rangle$ is a true instance of Disc-Set-Seq-Dec, that is, $\mathbf{I}$ is a discriminating set for C.*

Proof: (Sketch) We can interpret the structure M as follows. M represents the reachable states of a system that consists of two completed circuits C_{X_1} and C_{X_2} of C that agree with the black-box circuit on every input sequence in $\mathbf{I}$. The two completions are simultaneously exercised at every step with the same input. The equiv proposition holds at any state if the two completions produce the same output for the input applied in that state. If the equiv proposition holds on every path of M, as expressed by the linear temporal logic formula $\mathbf{G}$ equiv, then we have that any two such C_{X_1} and C_{X_2} are equivalent. Since a correct completion has to necessarily agree with the black-box circuit on every input sequence, it follows that both X_1 and X_2 are equivalent to *some* correct completion, and are therefore, by extension, correct completions themselves. By the definition of a discriminating set of input sequences, then, it follows that $\langle C, \mathbf{I}, \mathcal{C}(\mathbf{I})\rangle$ is a true instance of Disc-Set-Seq-Dec. ■

The above structure can be described using NuSMV's input language, with size at worst polynomial in the given Disc-Set-Seq-Dec instance. The M_{UMC} solver thus produces a description of the structure M corresponding to the input Disc-Set-Seq-Dec instance, and invokes a model checker asking to verify the specification $\mathbf{G}$ equiv on M. If the model checker says that M satisfies the specification, M_{UMC} returns true, otherwise M_{UMC} parses the counterexample returned by the model checker for a tuple $\left\langle X_1, X_2, \tilde{I}\right\rangle$ to be returned to the caller, as expressed in Algorithm 1. We implement the solvers M_{UC} and M_{CE} using similar calls to a model checker on (slight variations of) the structure M.

IV. Experimental Results

In this section, we describe our experimental results. We begin by describing our experimental setup, and then analyze the strength of our attack empirically.

A. Experimental Setup

We implement our attack procedure using C++ in ≈ 700 lines of code, and use NuSMV [17] as the back-end model checker. All experiments were executed on an Intel(R) Xeon CPU E5-2650 processor. We assume a camouflaged standard cell library that can implement either a NAND or a NOR function. Note that the camouflaged standard cell library in [6] also implements XOR functions (in addition to NAND and NOR), but we did not observe any instances of XOR in the benchmarks that we use.

We implemented two techniques to select which gates to camouflage: (1) the *output corruptibility* + *non-resolvable* (OC/NR) technique proposed by [6], which is secure against VLSI test based attacks, and (2) random selection, in which the camouflaged gates are picked uniformly at random from the set of eligible gates in the circuit.

We use circuits from the ISCAS'89 [13] and the ITC'99 [18] sequential benchmark suites. The characteristics of the benchmarks in terms of the number of inputs, outputs, flip-flops and gates is shown in Table I. More details about these benchmarks can be found in [13] and [18]. We note that the b14 benchmark implements a subset of the VIPER processor. For the s38584 benchmark, all our attack runs crashed, presumably because the model checking instances we generated in our procedure for s38584 were too large for NuSMV to handle. As such, we do not report any further results for this benchmark.

B. Experimental Results

Table II shows the results of our attack on the scheme in which gates are randomly camouflaged. We created 10 different camouflaged circuits for every benchmark, each with a different random selection of 32 camouflaged gates. The table plots (1) the number of discriminating input sequences, (2) the maximum length of an input sequence in the discriminating set, (3) the time taken by our attack, and (4) the number of attack runs (out of 10) that were successful and, for successful attacks, the termination condition that provided the correct completion.

Several observations are in order. Out of 160 attacks, 135 runs were successful. We were unable to decamouflage any instance of the s9234 benchmark, only decamouflaged one instance of the s5378 benchmark, and seven instances of the s400 and s444 benchmarks. On the other hand, our attack decamouflaged all instances of the b14 benchmark, one of the largest that we tried.

We hypothesize that even though the s400 and s444 benchmarks are small, they are hard to decamouflage because they have a relatively small number of primary inputs and outputs (3 PIs and 6 POs) compared to the number of bits of internal state (21 FFs each). We note that our attack required relatively long input sequences of length up to 90 for the instances in which we successfully decamouflaged these benchmarks. We do show in Section IV-B1, however, that on all of the instances of s400 and s444 for which we are unsuccessful, we are able to correctly recover at least 30 of 32 camouflaged gates.

Another interesting observation is that our UC and CE termination conditions that try to avoid calls to an unbounded

TABLE I: Benchmark characteristics.

B'mark	#PIs	#POs	#FFs	#Gates
s344	9	11	15	160
s349	9	11	15	161
s382	3	6	21	158
s400	3	6	21	162
s444	3	6	21	181
s526	3	6	21	193
s820	18	19	5	289
s832	18	19	5	287
s953	16	23	29	395
s1196	14	14	18	529
s5378	35	49	179	2779
s9234	19	22	228	5597
s38584	12	278	1452	19253
b04	11	8	66	628
b08	9	4	21	183
b14	32	54	245	5678

TABLE II: Results of proposed attack on FSMs camouflaged using random selection. Also noted are the termination conditions: unique completion (UC), combinational equivalence (CE) and unbounded model checker (UMC).

B'Mark	# Disc Inputs		Max Steps		Time (s)		Termination
	min	max	min	max	min	max	UC/CE/UMC
s344	3	5	10	10	11	37	10/0/0
s349	3	7	10	10	15	69	10/0/0
s382	25	36	50	60	3482	41129	10/0/0
s400	18	34	50	90	4921	526499	6/0/1
s444	16	35	50	90	3379	52984	2/0/5
s510	7	15	30	40	300	29121	10/0/0
s526	29	39	120	120	37979	139252	10/0/0
s820	14	20	10	10	506	1030	10/0/0
s832	12	21	10	10	370	1211	10/0/0
s953	10	22	10	10	365	1709	10/0/0
s1196	14	44	10	10	795	2386	10/0/0
s5378	7	-	30	-	1350	-	0/1/0
s9234	-	-	-	-	-	-	0/0/0
b04	4	9	10	10	31	151	10/0/0
b08	26	117	20	20	619	10527	10/0/0
b14	14	21	10	10	14308	34273	10/0/0

model checker enhance success of our attack. For example, on the one instance of c5378 that we decmaouflaged, we were able to terminate because our combinational equivalence (CE) check succeeded quickly; a call to unbounded model checker to decide whether to terminate was still running after several hours.

Table III shows the same data, but this time for the OC/NR based camouflaging approach [6] and are qualitatively similar. Since the procedure is deterministic, we generate only one camouflaged netlist for each benchmark. It is interesting to note that OC/NR does not seem to provide any additional security against the model checking attack compared to the random camouflaging scheme.

Our empirical attack results provide mixed evidence as to our central question: does blocking access to scan chains increase the immunity of camouflaged sequential circuits against reverse engineering attacks. On the one hand, using our proposed model checking attack, we were able to decamouflage relatively large benchmark circuits. On the other, benchmarks like s5378 and s9234 have been decamouflaged in prior work [9] assuming scan access. Withholding scan access does seem to increase the security for these benchmarks.

TABLE III: Results of proposed attack on FSMs camouflaged using the OC/NR technique from [6]. Also noted are the termination conditions: unique completion (UC), combinational equivalence (CE) and unbounded model checker (UMC).

B'Mark	# Disc Inputs	Max Steps	Time (s)	Termination
s344	4	10	14	UC
s349	3	10	8	UC
s382	34	60	17713	UMC
s400	26	60	14803	UMC
s444	36	80	150569	CE
s510	13	40	703	UC
s526	29	80	13001	UC
s820	23	10	1508	UC
s832	16	10	253	UC
s953	11	10	127	UC
s1196	17	10	1150	UC
s5378	-	-	-	-
s9234	-	-	-	-
b04	7	10	191	UC
b08	31	20	1734	UC
b14	13	40	19026	UC

1) Partial Completions: We tried recovering the identities of as many camouflaged gates as possible for benchmarks that we could not successfully decamouflage every gate. We do this using a technique similar to the one proposed in [10], i.e., any camouflaged gate that is assigned the same identity by all remaining completions (those not eliminated by the set $\mathbf{I}$) can be assigned that identity. We call this a partial completion.

Based on this technique, we found that we were able to correctly identify 30 out of 32 camouflaged gates for the six s400 and s444 benchmark instances that we could not fully decamouflage. Fig. 3 plots the histogram of the number of correctly decamouflaged gates for the nine runs of the s5378 benchmark on which we were unsuccesful. We observe that between 21 and 29 out of 32 gates are correctly decamouflaged, significantly reducing the attacker's search space. For instance, in the two cases where our partial completion attack recovered 29 camouflaged gates, the attacker has only 16 remaining possibilities.

2) Impact of Number of Camouflaged Gates: So far, we have assumed in our experiments that only 32 gates are camouflaged. We increased the number of camouflaged gates for the s1196 benchmark from 32 to 256 (including 10 runs for each) and found that we are able to decamouflage the circuit in each case. Fig. 4 plots the run-time of our attack and the number of input sequences required to decamouflage (size of discriminating set) each instance. An interesting observation is that although the number of input sequences required to decamouflage the circuit increases with increasing number of camouflaged gates, the length of the input sequences in the discriminating set was always at most 10.

V. Conclusion

In this paper, we proposed the first attack methodology for reverse-engineering camouflaged sequential circuits without assuming that the attacker has scan chain access. We have identified the computational complexity of two underlying subproblems on which our attack procedure relies, and show that the problem of determining when a given set of input sequences is sufficient to decamouflage a circuit is in **PSPACE**. Based on this observation, we have developed a practical

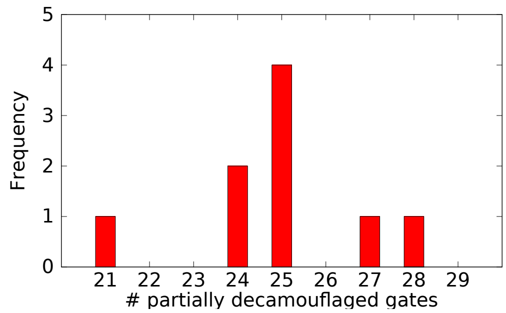
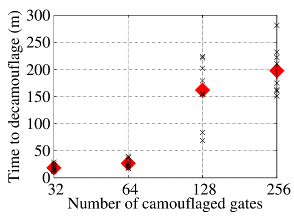
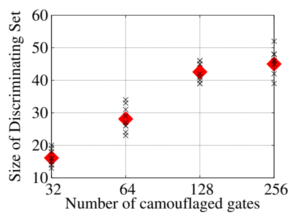


Fig. 3: Histogram of number of partially decamouflaged gates for the s5378 benchmark across nine runs on which our attack did not successfully recover every camouflaged gate.



(a) Attack time.



(b) Discriminating set size.

Fig. 4: Effect of increasing number of camouflaged gates on attack performance on the s1196 benchmark on (a) attack runtime, and (b) size of the discriminating set of input sequences.

and scalable attack procedure that makes iterative calls to a bounded model checker.

Our attack is effective on the majority of the benchmarks we tested, including a large sequential benchmark with more than 5000 gates. However, there are benchmarks that the attack does not fully decamouflage, suggesting that removing scan access may indeed be helpful in increasing the resiliency of some circuits against reverse engineering attacks. The attack motivates the need for further research into camouflaging mechanisms for sequential circuits that leverage the attacker's lack of access to internal state to further enhance resilience against our attack.

References

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

[2] SypherMedia. Syphermedia library circuit camouflage technology. http://www.smi.tv/solutions.htm. Last accessed May 2014.

[3] Lawrence E Larson. Convertible multi-function microelectronic logic gate structure and method of fabricating the same, September 8 1992. US Patent 5,146,117.

[4] Robert H Walden. Dynamic circuit disguise for microelectronic integrated digital logic circuits, April 13 1993. US Patent 5,202,591.

[5] Maria I Mera Collantes, Mohamed El Massad, and Siddharth Garg. Threshold-dependent camouflaged cells to secure circuits against reverse engineering attacks. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, pages 443–448. IEEE, 2016.

[6] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 709–720, New York, NY, USA, 2013. ACM.

[7] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. Camoperturb: secure IC camouflaging for minterm protection. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, pages 1–8. IEEE, 2016.

[8] Meng Li, Kaveh Shamsi, Travis Meade, Zheng Zhao, Bei Yu, Yier Jin, and David Z Pan. Provably secure camouflaging strategy for ic protection. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, pages 1–8. IEEE, 2016.

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

[10] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*, pages 137–143. IEEE, 2015.

[11] Duo Liu, Cunxi Yu, Xiangyu Zhang, and Daniel Holcomb. Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 433–438. EDA Consortium, 2016.

[12] Jeremy Lee, M Tebranipoor, and Jim Plusquellic. A low-cost solution for protecting IPs against scan-based side-channel attacks. In *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*, pages 6–pp. IEEE, 2006.

[13] Franc Brglez, David Bryan, and Krzysztof Kozminski. Combinational profiles of sequential benchmark circuits. In *Circuits and Systems, 1989., IEEE International Symposium on*, pages 1929–1934. IEEE, 1989.

[14] Degate. Reverse engineering integrated circuits with degate. http://www.degate.org/documentation/. Last accessed May 2014.

[15] Randy Torrance and Dick James. The state-of-the-art in IC reverse engineering. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 363–381. Springer, 2009.

[16] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference*, pages 83–89. ACM, 2012.

[17] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[18] Fulvio Corno, Matteo Sonza Reorda, and Giovanni Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of computers*, 17(3):44–53, 2000.

Appendix

Proof of Theorem 1: We prove that Disc-Set-Seq-Dec is in **PSPACE** by describing an algorithm for solving Disc-Set-Seq-Dec that requires an amount of space that is at worse polynomial in the size of the input instance. Recall that the input to Disc-Set-Seq-Dec is a tuple $C, \langle C, \mathbf{I}, \mathbf{O}\rangle$ where C is a camouflaged circuit, $\mathbf{I}$ is a set of input sequences, and $\mathbf{O}$ is a set of output sequences corresponding to an input sequence in $\mathbf{I}$. A precondition on $C, \langle \mathbf{I} \rangle$ is that there exists at least one completion X of C such that $C_X(\mathbf{I}) = \mathbf{O}$. The algorithm we propose is as follows:

For every pair of completions X_1 and X_2 in $\mathcal{X}$, we check whether $C_{X_1}(\mathbf{I}) = \mathbf{O}$ and $C_{X_2}(\mathbf{I}) = \mathbf{O}$. This can be done using $O(|C|)$ space, where the size of C is the number of inputs of wires plus the number of gates in C. If any of X_1 or X_2 do not satisfy the condition, we move on to the next pair; otherwise, we check whether C_{X_1} and C_{X_2} agree with each other on every input sequence of length l. Again, this can be done in $O(l|C|)$ space. If we find an input sequence of length l for which C_{X_1} and C_{X_2} produce different outputs, we return `false`. If we exhaust every pair of completions in $\mathcal{X}$, we return `true`.