On the Approximation Resiliency of Logic Locking and IC Camouflaging Schemes

Kaveh Shamsi, *Student Member, IEEE,* Travis Meade, *Student Member, IEEE* Meng Li, *Student Member, IEEE,* David Z.Pan, *Fellow, IEEE,* Yier Jin, *Member, IEEE,*

***Abstract*—The SAT based attacks are extremely successful in deobfuscating traditional combinational logic locking and IC camouflaging schemes. While several SAT-resilient protection schemes that increase the minimum query count of the attack have been proposed recently, none of them satisfy output corruptibility (error) criteria. Therefore, most of them were combined with high corruptibility schemes to achieve both corruptibility and high query count. These "compound" schemes are successful since existing SAT attacks are agnostic to the corruptibility of the protection scheme. In this paper we propose an approximate SAT based attack framework which focuses on the iterative convergence of an attack towards a better solution. This helps our attack reduce a compound scheme to a standalone SAT-resilient scheme. In addition, we relate the problem of minimum query count to a well known graph problem, and we propose a novel technique to increase the corruptibility of SAT-resilient protection schemes in a controllable manner. This creates protection schemes that have both high query count and corruptibility. Furthermore, due to the approximation resiliency property of these schemes, approximate attacks provide no advantage over exact attacks when attacking them.**

***Index Terms*—Hardware Security, Logic Obfuscation, IC Camouflaging, Logic Locking.**

I. Introduction

INTEGRATED Circuits (IC) are the backbone of modern day computing systems. Modern ICs are produced in a diversified global supply chain with multiple parties possibly from different nations, that carry out design, verification, and fabrication. Therefore, the security and privacy discussions have been initiated in the IC domain over the past decade [1]. While modeling the adversaries in the supply chain is itself a complicated task, the primary concerns can be broadly categorized as: 1) malicious modification of the design by the foundry, 2) reverse engineering by the foundry or end users for gaining critical information to help exploitation, or the theft of intellectual property (IP), and 3) illegal cloning, replicating or overproduction by the foundry.

While policy, surveillance, enforcing patent infringement laws, etc. can help thwart some of the above threats, design/fabrication-based techniques may provide a more cost effective solution. Since the majority of the supply chain threats arise from the fact that the design is being disclosed to malicious actors, hiding and obscuring the design can hinder attacks [1]. Logic encryption/locking [2], IC camouflaging [3], and split-manufacturing [4] are some prominent techniques for partially hiding the IC design from attackers. The first two techniques which are the focus of this paper do not require a trusted foundry and can be implemented by only modification to the design and in some cases, minor changes to the fabrication process.

Logic-locking/encryption or key-based obfuscation is based on corrupting the output of the circuit with additional key-inputs so that the circuit produces incorrect outputs without the correct secret key. The key bits are programmed post-fabrication by a trusted party hiding the design from the foundry and end-users. IC camouflaging is a layout-level technique based on creating indistinguishable layout structures for creating obscurity. These techniques can provide a layer of protection against different supply chain attacks. For instance, with logic locking, targeted malicious modification of the design is hindered through the obscurity of the locked circuit and the foundry cannot overproduce the design without the correct key. In addition, both IC camouflaging and logic locking hamper IC reverse engineering. Both schemes have the same mathematical model, and every logic locking scheme can be converted to an IC camouflaging scheme. Hence, in the rest of the paper whenever the term locking is used the discussion can be applied to camouflaging as well.

The research question is whether these schemes can be made *secure* with *low overhead*. The security is evaluated based on whether the original circuit can be recovered from the protected one under certain adversarial assumptions. The focus of this paper is on the security of combinational logic locking schemes when the attacker is able to query arbitrary input patterns and receive correct outputs on an unlocked circuit. Under this assumption, the most recent and powerful attack utilizes Boolean Satisfiability (SAT) solvers, known as the *SAT attack* [5], [6]. This attack utilizes a special set of input-output observations from the unlocked circuit to form a system of equations and solves it using a SAT solver to resolve the key bits or camouflaged layout functionalities.

A number of "SAT-resilient" logic locking and IC camouflaging schemes were proposed in literature [7], [8], [9], [10]. These schemes exponentially increase the number of queries required by the SAT attack to complete successfully. However, this comes at the cost of reduced output

Manuscript received xxx; revised xxx; accepted June 17, 2018. This work has been partially supported by the Defense Advanced Research Projects Agency (FA8650-18-1-7822). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jean-Luc Danger.

Kaveh Shamsi and Yier Jin are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA (email: kshamsi@ufl.edu, yier.jin@ece.ufl.edu).

Travis Meade is with the Department of Computer Science, University of Central Florida, Orlando, FL (email: travm12@knights.ucf.edu).

Meng Li and David Pan are with the Department of Electrical and Computer Engineering, University of Texas, Austin TX, USA (email: meng_li@utexas.edu, dpan@ece.utexas.edu).

Corresponding Author: Yier Jin.

error/entropy/corruptibility. Hence, most of these schemes propose to combine low output error blocks (*point-functions*) with traditional, high output error schemes to obtain a circuit that requires exponential queries to be resolved, while maintaining a sufficiently high output error as well. Hence, we refer to them as *compound* schemes.

Compound schemes are succesful in thwarting existing SAT attacks since such attacks are oblivious to the error of the logic locking scheme. In this paper however, we focus on the iterative approach of an attack towards a better approximation of the original circuit. We present a SAT-based attack called AppSAT which is able to deobfuscate the high error locking in a compound scheme and is thus an approximate attack. i.e the attack reduces a compound scheme to a low error scheme which itself is a highly accurate approximation of the original circuit. In addition we propose an approximation-resilient locking scheme and demonstrate its resiliency to attacks.

Contributions. This paper specifically delivers the following contributions:

- We provide a formal theory of the combinational locking/deobfuscation problem and link it with probably approximately correct (PAC) machine learning. For the first time we link the error versus query count in logic obfuscation to a well known and open hypergraph problem.
- We propose AppSAT for approximate deobfuscation based on the SAT attack augmented with random querying, intermediate error estimation and query reinforcement. We show the effectiveness of the attack by running it on 71 ISCAS and MCNC benchmark circuits obfuscated with the Anti-SAT+RLL (random XOR/XNOR insertion) [8] compound scheme. We show that the attack is capable of perfectly deobfuscating the high-error portion of this defense for 68 of the benchmark circuits.
- We present wire-disagreement analysis during the AppSAT attack as a tool that can be used to find hot-spots in the circuit during the deobfuscation process. This allows identifying the tip of the point-function in all compound-locked benchmarks that we experimented with. We also present a new SAT-based termination condition for the attack and discuss the limitations of such conditions in the general case.
- We present Diversified Tree Logic (DTL), a small technique for adding approximation resiliency to SAT-resilient schemes. DTL has a higher resiliency against removal attacks, can be used with different SAT-resilient schemes, and provides a simple way to tune the error versus query count with a better bound compared to existing work. We show that an implementation of DTL on benchmark circuits can prevent AppSAT from improving its approximation of the circuit throughout the entire attack. As such an approximate attack on DTL provides no advantage over an exact attack.

Organization. The paper is organized as follows. Section II provides necessary background and preliminaries on logic locking and camouflaging. Section III presents the framework and the different aspects of the AppSAT attack and its implementation. Section IV presents the proposed protection scheme and Section V discusses related work and concludes the paper.

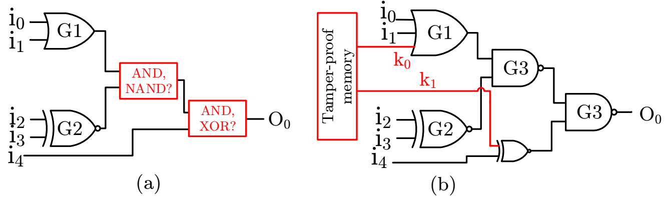

Fig. 1: (a) IC camouflaging by replacing gates with camouflaged gates. (b) Logic encryption with tamper-proof key bits.

II. Preliminaries

A. IC Camouflaging

IC camouflaging [3] techniques are physical-design oriented. Some notable schemes include using dummy connections, doping alterations, and dummy cells/wires in the circuit. A via with a middle gap can deceivingly appear to be connected during reverse-engineering serving as a dummy connection [11]. Doping based approaches [12] alter the doping type of transistors in a gate to change its functionality while it looks similar to the original gate from the top-view[1]. Inserting dummy gates, dummy wires and/or dummy filler metals have been proposed as well [3]. Fig. 1(a) shows gate-level IC camouflaging where gates in the circuit are replaced with dummy-connection-based camouflaged gates which look alike but have different functionalities.

B. Logic Locking

Combinational logic locking as seen in Fig. 1(b) is based on inserting key inputs (k_1 and k_0 in Fig. 1(b)) and key gates into the combinational circuit and ensuring that incorrect keys result in an incorrect output. Some notable traditional logic locking schemes (methods proposed before the SAT attack) include: 1) XOR/XNORing randomly selected wires [2] with key bits, known as Random Logic Locking (RLL) or EPIC; 2) XOR/XNORing wires that have maximum fault impact on the output with key bits, known as Fault based Logic Locking (FLL) [13]; 4) XOR/XNORing wires with key bits that result in a clique among the key bits, known as Secure Logic Locking (SLL) [14]; 3) replacing slices in the circuit with look-up-tables (LUTs) that store key bits [15], [16]; 4) inserting multiplexors (MUXs) [13] or switch-boxes (SB) [17] that are configured by key bits, as well as other gate insertion approaches [18][2].

C. A Formal Model for Locking and Camouflaging

Consider a combinational circuit. The original combinational circuit is a Boolean function from input $I = \mathbb{F}_2^n$ to output $O = \mathbb{F}_2^m$, denoted by $c_o : I \to O$. Logic locking can be modeled as transforming c_o to an augmented function $c_e : I \times K \to O$ where $K = \mathbb{F}_2^l$ is the key space, and there exists a correct key vector $k_* \in K_* \in K$ such that $\forall i \in I,\ c_e(i, k_*) = c_o(i)$. In other words, by traversing the key space a set of Boolean functions denoted by $C = \{c_e(i,k) | k \in K\}$ can be

[1]Special scanning-electron microscopy, capacitive imaging, or selective-etching have succeeded in revealing doping types.

[2]Note that any locking scheme can be converted to a camouflaging scheme by replacing programmable key-bits with camouflaged connectors. Hence the paper mainly discusses locking

obtained from which only some elements are equivalent to c_o. It is useful to think of c_e as a 2-dimensional truth-table with 2^l columns each representing the 2^n-high truth-table of a function in C as seen in Table I.

While logic locking directly fits in the above model, IC camouflaging requires a transformation from the camouflaged logic to an augmented Boolean function $c_e(i, k)$. This can be done by encoding different function possibilities of camouflaged units using key variables. For instance, camouflaged gates can be modeled as a set of gates arbitrated by a MUX where the MUX select lines are controlled by key bits [6] and dummy connections can be modeled with key-controlled MUXs as well. Note that this transformation can have significant effects on the performance of an attack.

D. The SAT Attacks

Given the above formal model, the SAT attack is an algorithm that finds a k_* given access to: 1) the Boolean expression/circuit for c_e which the attacker obtains by delayering, imaging, and reconstructing the netlist of the protected chip. 2) input-output or oracle access to c_o which means that the attacker can query a black-box with arbitrary i to get $c_o(i)$. This is why the SAT attack is considered an *oracle-guided* attack. Correct input-output pairs from the oracle can help disqualify incorrect keys.

Per Algorithm. 1, the baseline SAT attack [19], [6] begins by using a SAT-solver to satisfy a mitter condition, $c_e(i, k_1) \neq c_e(i, k_2)$ with $i_0, \hat{k}_1, \hat{k}_2$. The input i_0 that satisfies the mitter is referred to as a Discriminating Input Pattern (DIP). i_0 is queried on c_o and the the resulting constraint, $(c_e(i_0, k_1) = c_o(i_0)) \wedge (c_e(i_0, k_2) = c_o(i_0))$ is added to the mitter SAT problem and the algorithm repeats the DIP finding with tighter constraints. Once the mitter+constraints SAT problem can no longer be satisfied, there is only one possible truth-table in C that conforms to all DIP-output pairs, which should be the correct function. At this time, solving the constraints alone will return a $k_1 = k_2 = k_+ \in K_*$.

Algorithm 1 Given oracle access to c_o and the expression for C_e return a correct key $k_1 \in K_*$.

```
1: function SATDECRYPT($c_e$, $c_o$ as black-box)
2:     $j \leftarrow 0$
3:     $M \leftarrow c_e(i, k_1) \neq c_e(i, k_2)$
4:     $F_j \leftarrow true$
5:     while $F_j \wedge M$ is solvable do
6:         Solve $F_j \wedge M$ with $i_j$, $k_1$, $k_2$
7:         $o_j \leftarrow c_o(i_j)$
8:         $F_{j+1} \leftarrow F_j \wedge (c_e(i_j, k_1) = o_j) \wedge (c_e(i_j, k_2) = o_j)$
9:         $j \leftarrow j + 1$
10:    end while
11:    satisfy $F_j$ with $k_1$ and $k_2$
12: return $k_1$ as exact key
13: end function
```

The SAT attack has been shown to be able to deobfuscate low overhead traditional logic locking and IC camouflaging schemes in a matter of minutes. It is observed that large circuits, and those that are difficult for SAT solvers to handle such as multiplier circuits can cause slow downs in the SAT attack. However, all logic locking schemes mentioned in Section II-A when used with a reasonable number of key bits and added overhead (up to 20% of the original circuit), on benchmark circuits with up to thousands of gates, can be exactly deobfuscated by the SAT attack [5].

Considering the SAT attack it is reasonable to ask: what constitutes a reasonable overhead for protection? A protection scheme is considered low overhead if its overhead is significantly less than implementing the entire design in programmable logic (FPGA or PLA). Programmable logic consists of an array of look-up-tables with fully configurable interconnections. The large set of possible functions that can be implemented by the regular fabric make it difficult to learn the function without enlisting the entire truth-table. In fact, replacing select modules in the ASIC with programmable units was proposed in [20] as a protection scheme. The main impediment to using this on the entire chip is losing all the performance and energy benefits that come with ASICs as opposed to FPGAs.

E. Increasing Query Complexity

SAT-resilient locking/camouflaging schemes achieve a high query count by ensuring that each query is capable of omitting only a limited number of incorrect keys. Most these schemes are based on point-functions (comparator circuits). A point-function denoted as $P(i, k)$ outputs 1 when $i = k$ and 0 otherwise. The P function can be implemented in both IC camouflaging and logic locking by inverting the inputs to an AND-tree based on a secret [7], [21]. Almost all SAT-resilient schemes can be modeled as using a tree logic to generate a *flip* signal based on the key and the inputs to a logic cone, and then XORing the flip signal with the output of that logic cone. An example is shown in Fig. 2. We list SAT-resilient schemes based on the functionality at their flip signal given i as the n'-long cone input vector ($n' \leq n$ by picking sub-circuits in c_e) and k as the key vector.

- SARLock [7] by Yasin et al. is a logic locking scheme based on inserting a tree-like structure into the circuit with the flip signal being $(i = k) \wedge (k \neq k_*)$.
- CamouPerturb, [21] presented by Yasin et al., is an IC camouflaging technique based on flipping the output of the cone in the original circuit when $i = i_*$, and then correcting the output of the cone by XORing it with the flip signal $(i = i_*) \vee (i = k)$.
- Anti-SAT [8], a logic locking scheme by Xie and Srivastava, uses a $2n'$-long key vector, $k = \langle k_1, k_2 \rangle$, and uses $(i = k_1) \wedge (i \neq k_2)$ as the flip signal.
- Li et al. [22] proposed to find an AND-tree in the circuit and camouflage its inputs and use a dummy XOR to make it appear to the attacker that the output is being flipped with the condition $(i = k)$.

F. Increasing Output Corruptibility

Intuitively, the output of the protected circuit should be incorrect when an incorrect key is applied. For combinational protection schemes we can define the *output corruptibility/error* Cr as:

$$\mathsf{Cr}(c_e, c_o) \equiv \Pr_{i \in I,\ k \in K}[c_e(i, k) \neq c_o(i)]. \quad (1)$$

A corruptibility criteria for a locking/camouflaging scheme

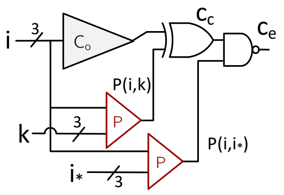

Fig. 2: An exmaple of a SAT-resilient locking scheme.

TABLE I: Truth-table for the circuit in figure 2. Every query disqualifies a single key possibility.

input	output	key							
$I_2I_1I_0$	O	K_0^*	K_1	K_2	K_3	K_4	K_5	K_6	K_7
000	0	0	0	0	0	0	0	0	0
001	1	1	0	1	1	1	1	1	1
010	1	1	1	0	1	1	1	1	1
011	0	0	0	0	1	0	0	0	0
100	0	0	0	0	0	1	0	0	0
101	1	1	1	1	1	1	0	1	1
110	0	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	0	1

can then be defined as:

$$|\Pr_{i \in I,\ k \in K}[c_e(i,k) \neq c_o(i)] - \frac{1}{2}| \leq \alpha(l) \tag{2}$$

where α is a negligible function. This ensures that the probability of a single-output cone disagreeing with the original circuit should be close to 0.5 when the input and key are chosen randomly. If the error is too close to 1 then the attacker can simply take a majority vote of protected circuits with random keys and pass the result through an inverter to obtain the correct functionality.

SAT-resilient schemes, if used alone, result in a very skewed output corruptibility. As seen from Fig. 2 the attacker can randomly select any *incorrect* key value, and the locked circuit will return a correct output for all but one input pattern. It is easy to show that $\mathsf{Cr} \in O(\frac{1}{2^{n'}})$ for SAT-resilient schemes, when the maximum query count is achieved, with n' being the width of the input to the cone.

Most SAT-resilient schemes discussed in Section II-E acknowledge this limitation and hence propose to combine the SAT-resilient protection with traditional high corruptibility schemes. We refer to such combinations as compound schemes. Since most traditional methods such as XOR/XNOR-based [11], [2], [13], [23], MUX [13] or SB-based [17], either were designed to maximize output corruptibility, or naturally result in a high corruptibility, adding them will effectively improve Cr for the overall locked circuit.

III. Approximate Deobfuscation

As was first noted in [22] oracle-guided deobfuscation attacks can be modeled with a specific problem in machine-learning called active-learning [24]. In active-learning terminology the *learner* intends to find a *target function* c_o, within a *hypothesis space* C, by querying a black-box that implements c_o. The subset of the hypothesis space that is consistent with the so far observed query set $\mathcal{L}$, is called the *version space* $\mathcal{V}$. The main goal in active-learning is to find an adaptive querying strategy that minimizes the number of queries required to learn the target function [24].

Among the different active-learning querying strategies, one ensemble referred to as uncertainty-sampling or query-by-disagreement (QBD) [25] best represents the SAT attack. By querying the oracle on input patterns that result in a disagreement among different functions in the version space, the attacker trims down the version space to functions equivalent to c_o. Existing SAT attacks terminate when no more disagreeing inputs can be found, at which point if C included c_o to begin with, the attack guaranties to find it. In this sense the existing SAT attacks are precise and we thus refer to them as exact SAT attacks.

However, exact SAT attacks are limited in that they are completely agnostic to the corruptibility of the protection scheme. This is why SAT-resilient schemes are successful in thwarting them even though they do not satisfy any corruptibility criteria. In this paper we study approximate attacks that are concerned with the iterative convergence of an algorithm towards a better key solution. An approximate oracle-guided attack will require the protection scheme to satisfy stronger security criteria. Note that if approximation resiliency is not considered as the resiliency metric, a simple comparison with a large key vector will secure the IC, which is far from our intuitive expectation from locking.

One way for formulating approximate learning problems is the *probably-approximately-correct* (PAC) setting [26], in which we specify that an algorithm A, with a probability of λ (probably), will return an ϵ-approximation (approximately correct) of the target function c_o. An ϵ-approximation of the target function c_o is a function that disagrees with c_o on at most an ϵ proportion of the input space. The approximate oracle-guided attacks discussed in this paper can be modeled with this formulation.

Based on the PAC model we can define approximation-resiliency criteria. A protection scheme is approximation resilient if there exists no algorithm that can learn the original circuit given the obfuscated circuit with negligible ϵ, and high success rate λ, in feasible time and space. We show empirically in this paper that existing compound schemes are not approximation-resilient. We specifically show that when attacking compound schemes, we can achieve an accuracy of $\epsilon \in O(\frac{1}{2^{n'}})$, in an empirically similar order of running time that it would take to deobfuscate traditional schemes with high success rate, assuming the width of the SAT-resilient scheme is n'. The remainder of this section we discuss this in more detail by presenting different aspects of an approximate attack.

A. A SAT-based Approximate Attack (AppSAT)

The current SAT attacks begin by satisfying the mitter circuit with a DIP, x_0. Subsequently, the oracle is queried with x_0 and a constraint is added to the SAT-formula. Whereas for the exact SAT attack the algorithm continues to find DIPs until no more DIPs can be found, we can build an approximate SAT attack by terminating the attack in any early step, t.

Let us define $\mathcal{L}_t$ to be the set of input-output observation (queries) collected until step t, and $\mathcal{V}_t$ be the version space consistent with $\mathcal{L}_t$. It can be shown that $|\mathcal{V}_t| < |\mathcal{V}_z|$ for all $z < t$. Therefore, the SAT attack is converging on the target function with each step by shrinking the version space. At any step, t, of the algorithm we can use the SAT-solver to find a key vector consistent with $\mathcal{L}_t$ by satisfying the conjunction of all queries. This translates to picking a function from $\mathcal{V}_t$. We denote the error probability of this function with ϵ_t defined as: $\epsilon_t(f) = |\{i | i \in I, f(i) \neq c_o(i)\}|/|I|$.

Different functions in $\mathcal{V}_t$ can have different ϵ (error) values with respect to the target function. Therefore, if the SAT-solver randomly picks a function from the version space, we can see how this conforms to the probably-correct property of PAC-learning, as the randomly selected function can have a higher error rate than previous steps. That is, $\epsilon_t < \epsilon_z$ may not be true for every $t > z$. However, overall we expect ϵ to decrease throughout the attack.

B. Error Estimation

In essence, the AppSAT attack avoids the exponential query count with compound schemes by stopping the querying process early. An important requirement is knowing *when* to stop querying. The intuitive approach is to stop when ϵ_t reaches a desired level. However, calculating ϵ precisely would require exponential queries itself since the truth-tables of the hypothesis and oracle need to be compared. We first note that it is straightforward to analytically represent the error probability of SAT-resilient schemes specifically. We can simply state that for any $t > 0$, we have $\epsilon_t \in O(\frac{1}{2^{n'}})$ for a maximum query SAT-resilient scheme. As for traditional schemes or compound schemes however, finding an analytic, and circuit-independent error value is difficult.

In this paper we perform random input pattern queries in order to estimate error. After every d number of DIP queries, we pick a function from the version space and compare it on r randomly selected patterns with the oracle to compute ϵ_t. We terminate the attack if ϵ_t stays below a threshold for more than a certain number of iterations (settlement-threshold). Note that random queries are cheaper than SAT-generated DIPs. An alternative approach for error estimation is to use the test patterns of the obfuscated circuit for error estimation as they result in higher flip coverage of the circuit wires.

C. Random Query Reinforcement

In the AppSAT attack, as was discussed previously, the accuracy of the resulting function has to be assessed using queries at every d number of DIP iterations. Since the learner is already paying the delay penalty for random queries for assessing error, we can use these queries as constraints in addition to DIPs. We refer to this technique as query reinforcement. Specifically, only the random samples on which the hypothesis disagrees with the oracle, will be added to the SAT formula as new constraints. Our experimental results show that this approach significantly improves the attack performance.

D. SAT-based Early Termination Conditions

A deobfuscation attack called Double-DIP (DDIP) was proposed in [27] and [10] that targets specifically the SARLock compound scheme. DDIP modifies the DIP mitter condition to find input patterns that can create disagreements among 4 key classes. The DDIP mitter condition is,

$$\Big(c_e(i,k_1) = c_e(i,k_3) = y_1\Big) \wedge \Big(c_e(i,k_2) = c_e(i,k_4) = y_2\Big)$$
$$\wedge\ (y_1 \neq y_2) \wedge (k_1 \neq k_3) \wedge (k_2 \neq k_4)$$

The i input that satisfies this condition is called a doubly differentiating input pattern (DDIP). A DDIP will eliminate at least two incorrect keys when queried. Therefore, the DDIP condition becomes unsatisfiable when attacking certain SAT-resilient schemes such as SARLock where each query is only removing a single incorrect key as seen in Table I. If SARLock is combined with a traditional scheme, then the DDIP attack will stop only when SARLock key bits are the only unresolved key bits ensuring that ϵ_t has reached $2^{-n'}$.

The main limitation of the DDIP attack is that slightly modifying the locking scheme can defeat it. For instance, if a single dummy key bit which is not affecting the output of the circuit, is added to SARLock, then the truth-table is duplicated in the key dimension. This causes all input patterns to immediately become DDIPs. In fact the Anti-SAT point-function scheme uses two key vectors resulting in a total key length of $2n'$ replicating the SARLock truth-table $2^{n'}$ times. As a result in the truth-table resulting from Anti-SAT each query disqualifies not one incorrect key but $2^{n'} - 1$ keys rendering the DDIP attack ineffective. Extending the attack to a q-DIP scenario where q different keys are to be discarded with each query will not help, since we must ensure $q \geq 2^{n'}$ which results in an exponentially large mitter CNF-SAT problem.

We discuss a possible improvement herein. While in Anti-SAT each input pattern disagrees with the oracle on an exponential number of keys, an invariant is that each function in the version space disagrees with the oracle circuit on no more than a single input pattern. As a result, once the high corruptibility schemes are resolved during the deobfuscation, the functions $c(i,k)$ that remain in the version space $\mathcal{V}_i$ have a hamming distance of 2 among their truth-tables. Hence a terminating condition can be designed to search for 3 different input patterns that have differing outputs in the version space:

$$\Big(c_e(i_1,k_1) \neq c_e(i_1,k_2)\Big) \wedge \Big(c_e(i_2,k_1) \neq c_e(i_2,k_2)\Big)$$
$$\wedge\ \Big(c_e(i_3,k_1) \neq c_e(i_3,k_2)\Big)$$
$$\wedge\ (i_1 \neq i_2) \wedge (i_2 \neq i_3) \wedge (i_1 \neq i_3)$$

This mitter can be expanded to find t input patterns that create inter-disagreement in the version space. This will ensure that the functions in the version space have inter hamming distances no larger than t. However, the size of this mitter grows with $\binom{t}{2}$.

This mitter is successful against SARLock with dummy keys and Anti-SAT when the Anti-SAT block wraps around the primary inputs and outputs of the circuit. However, this condition is a dual of the DDIP mitter in the input dimension. i.e. if a dummy input is added to the circuit or the Anti-SAT block is attached to the internal wires of the circuit, there is the possibility that the truth-table can get replicated along the input dimension. Once this occurs, there will be more than 3 inputs on which all functions in the version space disagree. Although the error rate relative to the size of the truth-table remains at $1/2^{n'}$, however, the termination condition will no longer assist in detecting that high corruptibility schemes have been resolved. Hence, it remains difficult to use such conditions in the general case. SAT-based model counting [28] can provide a way for counting the error, however, using existing SAT counting tools we were unable to achieve promising results. Furthermore, our proposed protection scheme renders such termination conditions ineffective.

E. Compound Schemes Under Approximate Attacks

As we discussed earlier, due to the low output corruptibility of SAT-resilient schemes, these methods are often combined with high corruptibility schemes. An important result of this

Algorithm 2 Given oracle access to c_o and the expression for c_e return an approximate key k_1 with parameters ErrorTheshold, and SettleThreshold

```
1: function ApproxSATDecrypt($c_e$, $c_o$ as black-box)
2:     $j \leftarrow 0$
3:     $M \leftarrow c_e(i, k_1) \neq c_e(i, k_2)$, $F \leftarrow true$
4:     while $F \wedge M$ is solvable do
5:         Solve $F \wedge M$ with $\hat{i}$, $k_1$, $k_2$
6:         $\hat{o} \leftarrow c_o(\hat{i})$
7:         $F \leftarrow F \wedge (c_e(\hat{i}, k_1) = \hat{o}) \wedge (c_e(\hat{i}, k_2) = \hat{o})$
8:         every $d$ rounds do
9:             AnalyzeError($F$ , $k_1$, $c_e$, $c_o$)
10:            CheckTermination($F$ , $c_e$, $c_o$)
11:            WireDisagreement($i_j$, $k_1$, $k_2$, $c_e$)
12:    end while
13:    satisfy $F$ with $\hat{k}_1$, $\hat{k}_2$
14: return $\hat{k}_1$ as exact key
15: end function

16: function AnalyzeError($F$, $k_1$, $c_e$, $c_o$ as black-box)
17:     for $\hat{x} \in RandomPatterns$ do
18:         if $c_e(\hat{x}, k_1) \neq c_o(\hat{x})$ then
19:             $FailedPatterns \leftarrow FailedPatterns + 1$
20:             $F \leftarrow F \wedge (c_e(\hat{x}, k_1) = c_o(\hat{x}))$
21:         end if
22:     end for
23:     $\epsilon \leftarrow \frac{FailedPatterns}{numRandomPatterns}$
24:     if $\epsilon <$ ErrorThreshold then
25:         $SettleCount \leftarrow SettleCount + 1$
26:         if $SettleCount >$ SettleThreshold then
27:             return $k_1$ as approximate key
28:         end if
29:     else
30:         $SettleCount \leftarrow 0$
31:     end if
32: end function

33: function WireDisagreement($i$, $k_1$, $k_2$, $c_e$)
34:     for wires $w_e$ in BFS search from outputs of $c_e$ do
35:         if $w_e(i, k_1) \neq w_e(i, k_2)$ then
36:             mark $w$ as live
37:         end if
38:     end for
39: end function
```

paper is that adding a high corruptibility scheme to the SAT-resilient schemes does not contribute to the overall security. The AppSAT attack is capable of deobfuscating the traditional portions of the compound protection as if the SAT-resilient scheme was not present.

To explain this phenomena, consider the truth table of the SAT-resilient scheme shown in Table I. Per Table I, in the beginning of the algorithm all input patterns are potential DIPs. In other words, the SAT-resilient scheme imposes a loose constraint on which inputs should be queried. A high corruptibility scheme can be visualized as adding highly disagreeing columns to this table. Then it can be seen that the freedom of the attack algorithm in picking DIPs is equivalent to running the attack on a standalone high corruptibility scheme as if the point-function was not present. As we will see in Section III-H, if the SAT-solver gets trapped into solving the point-function only, random query reinforcement can still help with excluding highly disagreeing functions in the version space.

F. AppSAT Wire-Disagreement Maps

It is possible to map the disagreements among functions in the version space to the internal wires in c_e. Every DIP excites a disagreement at the output wires of the two circuit copies in

TABLE II: Benchmark circuits.

ISCAS sequential			
circuit	#inputs	#out	#gates
s382	24	27	392
s400	24	27	414
s641	54	42	459
s526n	24	27	494
s1488	14	25	843
s953	45	29	950
s3384	226	201	1966
s5378	214	228	5183
s13207	700	790	11248
s15850	611	684	13192
s35932	1763	1728	31833

ISCAS & MCNC combinational			
circuit	#inputs	#out	#gates
c432	36	7	160
c499	41	32	202
i4	192	6	338
c880	60	26	383
c1355	41	32	546
c1908	33	25	880
c2670	157	64	1193
i9	88	63	1315
i7	199	67	1581
c3540	50	22	1669
dalu	75	16	2298
c5315	178	123	2307
i8	133	81	2464
c7552	207	108	3512
des	256	245	6437

M resulting from different key bits in k_1 and k_2. If we compare the Boolean state of all the wires in $c_e(i, k_1)$ with their corresponding wires in $c_e(i, k_2)$, we can get a sense of which parts of the circuit are currently being resolved serving as a heat-map for unresolved portions of the circuit. In compound schemes once the high corruptibility bits are resolved, the wire-disagreements form a cone rooted at the flip-signal. This can assist in removing such structures, or in general identifying which parts of the circuit graph are being actively analyzed by the attack and which parts seem to have already resolved. Note that this is superior to the AppSAT-guided removal attack in [29] which analyzes the compound obfuscation key-vector throughout AppSAT to find settled key bits and trace them to the tip of the tree. That attack can be thwarted by applying a transformation to the compound key that levels the flip rate across different bits. Conversely, the routine presented here performs one-to-one wire-disagreement testing starting from the output and is hence oblivious to transformations applied to the key. Algorithm. 2 shows the overall flow of AppSAT including the wire-disagreement analysis.

G. Experiments: Framework

For our experimentation, we implemented both attacks and defenses in a C++ framework. The framework utilizes Minisat [30] for SAT-solving and includes the conventional exact SAT attack augmented with intermediate key vector extraction, error estimation/calculation, SAT termination conditions, and random query reinforcement routines. All tests were performed on a quad-core Intel Xeon E3 processor with a 3.4GHz CPU, and 16GB memory.

H. Experiments: Attacking Compound Schemes

We evaluated Algorithm 2 on the benchmark circuits that are listed in Table II. The benchmark circuits were locked with Anti-SAT[3] integrated with RLL (random XOR/XNOR locking) [2]. Then the AppSAT attack was launched with the following parameters: after every 12 iterations of the SAT attack we query 50 randomly generated patterns to estimate ϵ and then store the disagreeing patterns as constraints. The settlement-threshold was set to 5.

The running time of AppSAT against the Anti-SAT+RLL scheme are shown in Figures 3 and 4. No circuit in these tests was queried more than 1200 times. We performed SAT-based equivalence checking to verify the correctness of the RLL key

[3]While our experiments are based on the Anti-SAT method, the attack can easily be used against other SAT-resilient schemes including CamouPerturb [31], SARLock [7], and Li's [22] Methods.

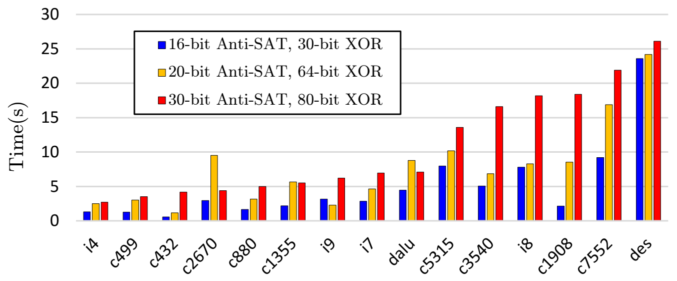

Fig. 3: Running time for decrypting ISCAS and MCNC combinational benchmarks.

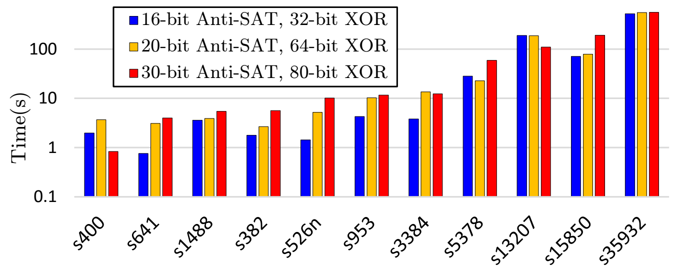

Fig. 4: Running time for decrypting unrolled ISCAS sequential benchmarks.

bits. For 41 out of the 43 combinational circuits the RLL key bits were recovered correctly. The 2 failed cases both were the c2670 circuit which has an internal AND-tree and hence key bits behind the tree could not be found. 1 out of 28 sequential benchmarks failed the equivalence test, again due to internal low corruptibility blocks. In both cases the invariant is that the recovered function is a highly accurate approximation of the target function. The wire-disagreement analysis also settled towards a cone-shaped structure that we expected once the RLL bits are resolved, allowing an attacker to spot the output of the Anti-SAT block on all benchmarks.

I. Experiments: Error Profiles

The AppSAT error profile can serve as a visual tool for analyzing the approximation resiliency of protection schemes. We generated a set of random circuits with a small number of inputs (9), so that the error value can be precisely calculated by a sweep of the 512 input patterns. We used ABC [32] to synthesize randomly generated truth-tables resulting in circuits with 200-300 gates, and then locked them using FLL [13], MUX based obfuscation [13], SLL [23], and RLL [2] using[5]. We first performed AppSAT with no query reinforcement on these circuits and recorded ϵ with each iteration as seen in Figure 5a. We then added a 9-input point-function using Anti-SAT's approach and performed a similar attack with the error plotted in Figure 5(b). As can be seen, Anti-SAT is able to heavily delay achieving a good accuracy due to the SAT-solver getting trapped in the point-function. We then attack the same circuits using AppSAT with 10 random samples being reinforced each iteration and the improvements can be seen in Figure 5(c). Furthermore, the FLL [13] method shows a higher approximation resiliency as seen from its square shape ϵ plot. This is primarily due to the fact that even a single key bit not being resolved correctly can have a high effect on the corruptibility of the circuit.

J. Deobfuscating SAT-resilient Schemes

AppSAT reduces a compound locking scheme to a SAT-resilient scheme. While this in many cases is sufficient for the attacker, the question is whether a standalone SAT-resilient scheme can be broken. There are several methods for attacking such schemes. First is removal attacks [29] where the attacker finds the flip signal in the circuit and sets it to a constant which can be assisted by the wire-disagreement analysis in AppSAT. However, CamouPerturb [31] and [33] proposed designing a circuit with a striped functionality and then using a flip signal to correct the functionality. As a result, removing the flip signal will not return the original circuit. Another possible attack is the "bypass attack" presented in [10]. In this attack using the SAT-solver, two different keys, k_1 and k_2, are selected from the version space of the point-function scheme. The disagreeing input pattern set $BI = \{i \in I | c_e(i,k_1), c_e(i,k_2)\}$ is collected using iterative calls to the SAT-solver and then the output of the circuit is fixated to the value of the oracle circuit for all patterns in BI. Another method for attacking a standalone SAT-resilient schemes is to use a majority vote of 3 or n different c_e replicas which are fed with different randomly selected keys. Such a circuit will always produce the correct output due to the low corruptibility of the locking scheme. The second two methods leave the attacker with a functionally-correct circuit, however, the size and delay of this circuit will be more than the original circuit.

IV. Approximation Resilient Protection

A. Query Complexity Versus Output Corruptibility

In this paper we present a tree-based approximation resilient obfuscation scheme. Before we discuss the scheme, we show that there is a fundamental trade-off between the minimum number of queries required to resolve a locked circuit using oracle-guided attacks, and different corruptibility/error measures. Consider the truth-table view of the locked circuit. The error matrix $E = (e_{ik})$ of a locking scheme is a $2^n \times 2^l$ binary matrix where $e_{ik} = 1$ if $c_e(i,k) \neq c_o(i)$ and $e_{ik} = 0$ otherwise.

If there are a total of M 1s in this matrix, corruptibility is:

$$\begin{aligned} \mathsf{Cr}(c_e, c_o) &= \Pr_{i \in I,\ k \in K}[c_e(i,k) \neq c_o(i)] \\ &= \frac{|\{(i,k) | c_e(i,k) \neq c_o(i)\}|}{|I||K|} = \frac{M}{2^{n+l}}. \end{aligned} \tag{3}$$

Hence the overall corruptibility is determined by the total number of disagreeing locations in the 2D truth-table. The minimum query count depends on how these ones are distributed throughout the matrix and is generally a complicated covering problem. Here we focus on when the ones are distributed evenly where each column has a fixed number of ones X. This ensures that every incorrect key has at least X incorrect locations. Approximation resiliency can be achieved by exponentially growing X.

A *hypergraph* $\mathcal{H}$ is a generalization of a graph denoted by (V,E), where $V = \{v_i | i \in I\}$, is the set of vertices where I is the set of vertex indices, and $E = \{e_j | e_j \subseteq V \wedge j \in J\}$, is the set of *hyperedges*, J being the edge index set, and each hyperedge can group several vertices together. A hypergraph can be represented by a $|V| \times |E|$ incidence matrix $A = (a_{ij})$ where $a_{ij} = 1$ if $v_i \in e_j$ and $a_{ij} = 0$ otherwise.

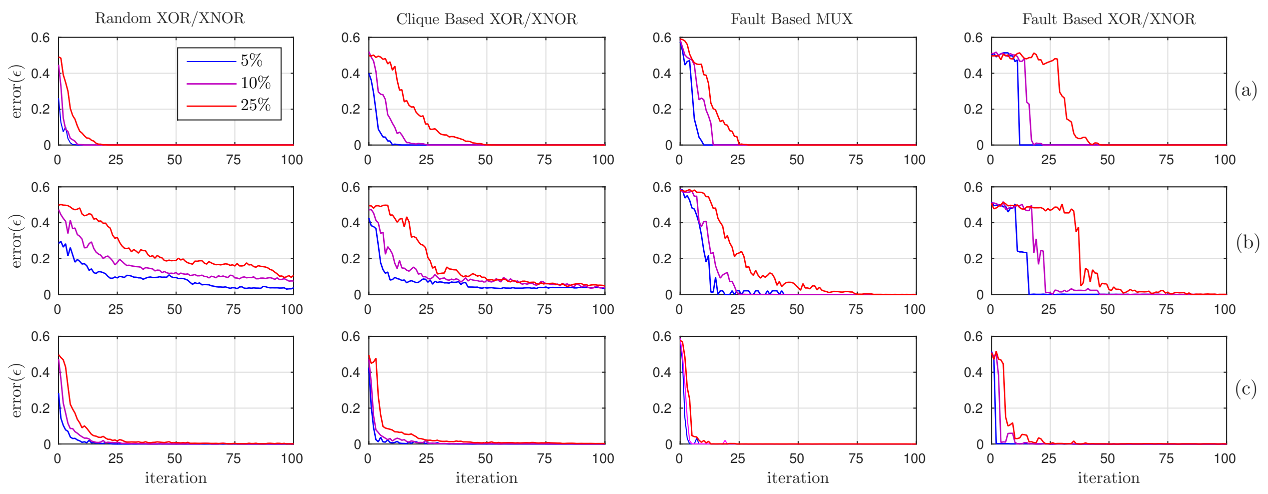

Fig. 5: Exact SAT attack error values (ϵ) versus iterations for: a) high corruptibility schemes, and b) the circuits in (a) augmented with an Anti-SAT block. Figures in the (c) row show the error trend when 10 random queries were reinforced in each step resulting in less iterations and time.

An r-uniform hypergraph is a hypergraph where each hyperedge connects a fixed number of vertices r. i.e. $\forall e_i \in E, |e_i| = r$. A *hitting-set* or *transversal* in a hypergraph is a set $T \subseteq V$ such that T has a non-zero intersection with all of the $|E|$ edges. The size of the smallest transversal in a hypergraph is referred to as the *transversal number* of the hypergraph $\tau(H)$. As a result, the following connection can be made between transversals in r-uniform hypergraphs, and the minimum query set required to disqualify all keys in a locking scheme that ensures a fixed error count X for each incorrect key:

Proposition 1: The minimum number of queries QC required to deobfuscate a circuit c_e to c_o, where c_e has exactly X erroneous truth-table entries per incorrect key, is equal to the transversal number of the X-uniform hypergraph $\mathcal{H}$ described by the error matrix for c_e versus c_o.

It is rather obvious that increasing X reduces the size of the minimum transversal, also that $\tau(H)$ is bounded by a function of $|V|$ and X. However, a tight analytic bound for general regular hypergraphs remains an open problem [34]. Some noteworthy results include Alon's bound [35]:

$$\mathsf{QC} \leq (2^n + 2^l)\frac{\ln X}{X} \tag{4}$$

, and the bound obtained by Chvátal et al. [36]:

$$\mathsf{QC} \leq \frac{2^n + 2^l \lfloor \frac{X}{2} \rfloor}{\lfloor \frac{3X}{2} \rfloor} \tag{5}$$

Zhou et al. [37] presented the bound $\mathsf{QC} \leq (2^n + 2^l)\frac{1}{X}$ for logic locking for the special case when $n = l$ using a matrix covering problem. However, the above hypergraph bounds clearly exceed Zhou's bound which we found to be due to an error in their proof. In fact the locking scheme that we present herein exceeds their proposed bound on error.

B. Diversified Tree Logic (DTL)

Consider the point-function tree logic that generates a flip signal shown in Figure 6. The comparator circuit is comprised of a layer of XOR gates that lead to an AND-tree. The AND-tree has an onset size of 1 (it outputs 1 for only a single input pattern). Assume the AND-tree is constructed only with 2-input AND gates. The idea in DTL is to replace some gates in this AND-tree with OR/NAND/XOR gates, to control the onset size and shape. For instance, if t gates in the first layer of the AND-tree are replaced with OR/NAND gates, the onset size increases from 1 to 3^t, and if replaced with t XOR gates, the onset size increases from 1 to 2^t. Integrating such a tree carefully into existing SAT-resilient schemes results in locking schemes with a controllable minimum error count X for all incorrect keys.

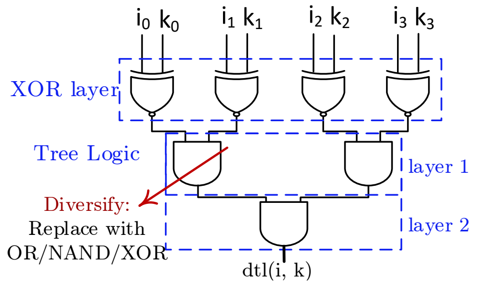

Fig. 6: Diversifying the tree logic by replacing AND gates with OR/NAND/XOR gates results in an increased controllable corruptibility.

The DTL technique described herein has several benefits. The designer can tune the error value by selecting the number and level of AND gates to replace ensuring a certain error per key value. Replacing several gates in the tree results in a diversified tree which helps hinder removal attacks that use functional analysis and signal probability [38] as compared to tree constructed solely by AND or OR gates. In addition, an exponentially large X renders SAT termination conditions and bypass attacks ineffective.

Replacing an AND gate in the tree with an OR/NAND/XOR gate loosens the equality condition on the inputs to the point-function. The original AND-tree point-function $P(i, k)$ outputs 1 only when $(i = k)$. If we replace a gate in the first layer of the tree with an OR, the tree outputs 1 for two additional cases since the OR gate has an onset of size 3. For a general layer lr the onset consists of 1 case where the two sub-vectors

integration method	gate-type	first layer		layer lr (lr is from 0 to $\lceil \log n \rceil$)	
		corruptibility per key (X)	minimum query count	corruptibility per key (X)	minimum query count
SARLock	XOR	2^t	2^{n-t}	$(2(2^{2^{lr-1}}-1))^t$	$\geq 2^n/X$
	OR	3^t	2^{n-t}	$(1+2(2^{2^{lr-1}}-1))^t$	$> 2^n/X$
	NAND	3^t	2^{n-t}	$(1+2(2^{2^{lr-1}}-1))^t$	$> 2^n/X$
Anti-SAT	XOR	2^t	2^{n-t}	$(2(2^{2^{lr-1}}-1))^{t-1} \leq X \leq (2(2^{2^{lr-1}}-1))^t$	$\geq 2^n/X$
	OR	$3^{t-1} \leq X \leq 3^t$	$> 2^{n-t}$	$(1+2(2^{2^{lr-1}}-1))^{t-1} \leq X \leq (1+2(2^{2^{lr-1}}-1))^t$	$> 2^n/X$
	NAND	$3^{t-1} \leq X \leq 3^t$	$> 2^{n-t}$	$(1+2(2^{2^{lr-1}}-1))^{t-1} \leq X \leq (1+2(2^{2^{lr-1}}-1))^t$	$> 2^n/X$

TABLE III: query complexity versus corruptibility per key for different DTL gate flips and integration methods.

in the tree match, and two cases for when one sub-vector matches the key but the other does not, allowing it to take on $2^{2^{lr-1}}-1$ possibilities. This results in $X = 1 + 2(2^{2^{lr-1}}-1)$ erroneous locations in each column. Replacing t gates in layer lr raises this value to the power of t. A similar number holds when using NAND gates whereas when using XOR gates, the 1 is dropped from the X value since the onset size of XOR is 2 rather than 3. The minimum query count for gate replacements in a general layer lr (lr ranges from 1 to $\lceil \log(n) \rceil$), or combining different gates can be difficult to represent in closed-form. However, an invariant is that the minimum query count remains above $2^n/X$ since each query can remove at most X incorrect keys and X is the onset size of the diversified tree. These results can be seen in Table III.

C. Integrating DTL with SAT-Resilient Schemes

The DTL block can be integrated with existing SAT-resilient schemes in order to increase their corruptibility. Three integration methods are described herein.

SARLock: the circuit diagram for this method is depicted in Figure 8(a). In this scheme we begin by constructing a flip signal using the output of a DTL block $dtl(i,k)$. However, the output of the DTL block $dtl(i,k)$ as seen in Figure 7 has no correct key columns as there is at least a single one in each column. Therefore, a P block can be used to mask the flip signal when the correct key is applied. The resulting flip signal which is XORed with a wire in the circuit is: $dtl(i,k) \wedge (k \neq k_*)$. In this integration method the P block that forms part of the flip signal is "key-only logic"; i.e. there are no primary inputs that affect the wires in this block. Therefore, the attacker can remove this logic and model the entire key-only logic cone with a single new key bit. Inserting a shuffler that shuffles input signals and keys before they enter the block can alleviate this issue at the cost of higher overhead.

Anti-SAT: unlike SARLock, Anti-SAT does not contain key-only logic. DTL can be integrated with the double-key style in Anti-SAT. This is seen in Figure 8(b). In this scheme a DTL tree is constructed and is replicated to form the flip condition: $dtl(i,k_1) \wedge \overline{dtl(i,k_2)}$ where $k = \langle k_1, k_2 \rangle$ is the overall key vector. In this scheme, the first condition in the flip signal, $dtl(i,k_1)$ creates the DTL error matrix seen in Figure 7. The second condition replicates this matrix in the key dimension $2^{|k_1|}$ times resulting in $2^{2|k_1|}$ columns. In each replica a number of the rows in the matrix are masked (all set to 0) creating a number of correct keys in each table replica. Due to this masking, this integration style can degrade the corruptibility for some keys. However, as seen in Table III the X value stays within some bound depending on the number of replaced gates.

Corrupt and Correct: SARLock-DTL and Anti-SAT-DTL can be vulnerable to removal attacks. Removal-resiliency can be created by modifying the original logic cone to corrupt its output, and then using a P block to correct the output of the modified cone [21], [33]. DTL can be incorporated in a corrupt and correct manner as well where a controllable number of input patterns are first corrupted, and then they are corrected using a DTL generated flip signal. As seen in Figure 8(c) a DTL block with a fixed pattern i_* can be used to first corrupt the output of the original circuit: $c_{crpt} = c_o \oplus dtl(i,i_*)$. Then the c_{crpt} is resynthesized to mix the added logic into the selected cone. Note than since DTL has a more diverse tree structure, it has a higher chance of getting mixed in with the original logic after resynthesis improving its stealthiness. Then the output is corrected using a flip signal: $c_e = c_{crpt} \oplus flip$. The flip signal is generated as $flip = dtl(i,k) \vee dtl(i,i_*)$. This flip signal will mask (set to 0) a number of rows in the DTL error matrix, effectively creating a number of correct key columns. This integration style similar to Anti-SAT-DTL can degrade corruptibility for some keys, however, it has an enhanced resiliency against removal attacks [21]. Note that an additional $P(k,k_*)$ block can be added to the flip signal to minimize the effect of masking to only the correct key column at the cost of additional overhead and creating key-only logic. In order to prevent the attacker from deriving the corrupted patterns from the DTL structure, functional obfuscation can be added to the correction DTL.

D. Experiments

We implemented DTL in our C++ framework. We used the DTL method applied to SARLock, where only the first layer of tree logic was diversified with OR gates since it produce the highest corruptibility in a robust manner. The c432 benchmark circuit which has 36 input wires and 160 primitive gates was locked with SARLock-DTL with varying tree widths and number of OR gates inserted in the first layer. The SAT attack was then run on the locked netlists. It can be seen from Table IV, with 25 inputs to the tree and 10 OR gates inserted in the first layer, the SAT attack runtime reaches the two hour time limit. Such a scheme ensures that 2^{15} queries are required to resolve the key and that each incorrect key has $3^{10} = 59049$ erroneous locations in the function that it selects. Since c432

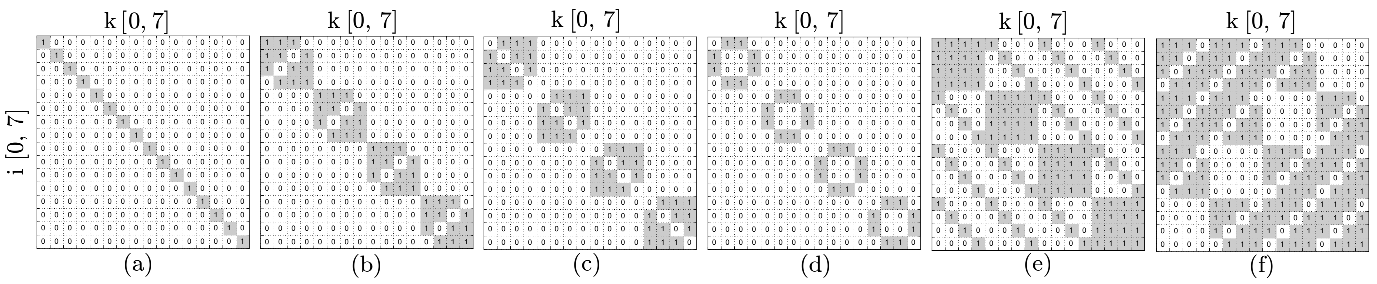

Fig. 7: Output pattern of $dtl(i, k)$ for 8-bit i and k. (a) AND-tree. (b) one NAND gate in the first layer ($X = 3$). (c) one OR gate in the first layer ($X = 3$). (d) one XOR gate in the first layer ($X = 2$). (e) one OR gate in second layer ($X = 7$). (f) two OR gates in first layer ($X = 3^2$).

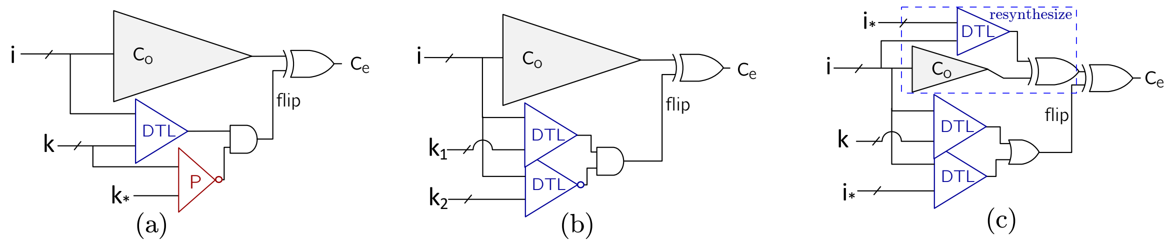

Fig. 8: Different SAT-resilient schemes integrated with DTL. (a) SARLock-DTL. (b) Anti-SAT-DTL. (c) corrupt and correct DTL scheme.

num or-gates		6		8		10		12	
tree width	overhead	#DI	runtime (s)	#DI	runtime (s)	#DI	runtime (s)	#DI	runtime (s)
15	0.38	379	3.08	102	0.26	29	0.04	7	0.02
16	0.40	815	17.99	167	0.92	53	0.09	15	0.03
17	0.43	1420	72.48	357	2.91	92	0.34	27	0.05
18	0.45	3324	369.48	545	14.21	182	1.13	52	0.12
20	0.50	11717	4844.56	2819	356.27	365	12.63	125	0.71
25	0.63	TO	TO	TO	TO	TO	TO	2132	1197.09

TABLE IV: Running time and query count of SAT attack on SARLock-DTL for various tree sizes and OR gate counts in the first layer. Overhead is in terms of pre-synthesis gate count.

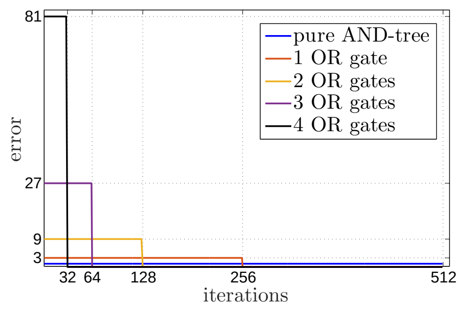

Fig. 9: AppSAT error profile for SARLock-DTL with first layer OR gates inserted.

is our smallest benchmark these results can be significantly improved with larger circuits.

We also ran AppSAT on an 8-input netlist and plotted the error profile for different number of OR gates inserted into the circuit. The result can be seen in Figure 9 which shows how the AppSAT error value does not improve throughout the attack since every incorrect key ensures a fixed number of erroneous input patterns. However, as the error is increased, the query count is also exponentially reduced due to the fundamental trade-off. Using the OR-gate diversified tree however, the corruptibility increases with 3^t but query count drops with 2^t which is better than the XOR-based tree proposed in [37].

V. Discussions and Conclusion

In this paper we showed that there is a fundamental trade-off between corruptibility and query count. This trade-off was first shown in [37]. However, the OR-gate based DTL method proposed herein exceeds the bound presented in [37] due to an error that we discovered in their proof. The bound that we proposed is based on a well known and long standing problem in graph theory.

We presented DTL which can be used to improve the corruptibility of several existing SAT-resilient schemes. To compare this to related work, Anti-SAT's construction [39] uses an arbitrary p value which is the onset size of the Anti-SAT function. By increasing this p higher error values may be achieved but the paper discouraged this and instead proposed compounding for increasing error rates which was proven insecure in this paper. It also did not provide a systematic way for constructing higher p values which is what DTL does. In other related work, Yasin et al. [33], [40] presented SAT-resilient schemes that creates per key corruptibility values higher than one. They use several methods for this. One is to use m different P blocks to create m erroneous truth-table entries and another is to use a hamming-distance detection block that will create $\binom{n}{m}$ erroneous entries. The DTL methods are lighter since they use a single tree for creating an onset of size X. Another method is based on using a look-up-table (LUT) for corrupting and correcting arbitrary input patterns.

The LUT based method can only hide a linear number of input patterns while our X values are exponential (3^t for OR-gate based DTL). Zhou et al. also presented a construction in [37] that uses an AND-tree with all first layer gates replaced with XORs. This scheme is a special case of DTL and has a lower error-query product compared to OR/NAND-diversified trees. Furthermore, we proposed several integration methods that can hinder key-only logic removal and functional-analysis-based removal with a diversified tree while the construction in [37] uses a fixed tree with key-only logic.

Another idea for preventing SAT attacks is to use cryptographic functions. Cryptographic functions are known to be difficult to invert. In fact using SAT solvers to solve a system of input-output equations to resolve the key is commonly studied in the cryptography domain. [37] proposes using the Goldraich's one-way function to hinder the SAT attack. In our work we also studied inserting rounds of a lightweight cipher such as PRESENT into the circuit. However, the main challenge is that cryptographic functions defeat SAT solvers either through large integer multiplication (RSA family of cryptosystems) or through a repetition of diffusion and confusion layers (block ciphers). Inserting multiplication units can incur enormous overhead which can exceed that of implementing the circuit in programmable logic. Iterative block ciphers on the other hand, achieve their one-wayness only after repeating a round a significant number of times. For instance, up to 3 rounds of PRESENT each of which containing a 1000 gates where reversed with our SAT attack when PRESENT [41] was inserted in the path of the key in RLL. Plus it results in key-only logic as it is a key-only transformation. Optimized SAT solver based algebraic attacks [42] can significantly increase the number of rounds that can be attacked. Hence, overhead and hiding the cipher round in the circuit can become a limiting factor in using such methods.

Cyclic logic locking was proposed in [43] as a SAT-resilient method. However, it was later shown that a variant of the SAT attack called the CycSAT [44] is able to deobfuscate cyclic logic locking in seconds by prepossessing the circuit. Even though cyclic locking may not contribute to the SAT-resiliency of DTL, it can still be added to the overall circuit to disrupt structural and wire-disagreement analysis of the locked circuit helping to hide the DTL logic.

To conclude, in this paper a general approximate SAT-based attack was presented and a new theoretical foundation for the problem of logic locking and deobfuscation was proposed. Furthermore, a novel technique was proposed that with minor modifications can improve the corruptibility of several existing SAT-resilient logic locking schemes at low overhead.

REFERENCES

[1] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, p. 6, 2016.

[2] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proc. Design, Automation and Test in Europe*, ser. DATE '08, 2008, pp. 1069–1074.

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

[4] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Design, Automation and Test in Europe*, 2013, pp. 1259–1264.

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

[6] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (ic) decamouflaging: Reverse engineering camouflaged ics within minutes." in *Network and Distributed System Security Symposium (NDSS)*, 2015.

[7] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*, 2016, pp. 236–241.

[8] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *Proc. Int. Conf. on Cryptographic Hardw. and Embed. Systems*. Springer, 2016, pp. 127–146.

[9] K. Shamsi, W. Wen, and Y. Jin, "Hardware security challenges beyond cmos: Attacks and remedies," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 200–205.

[10] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," in *Proc. Int. Conf. on Cryptographic Hardw. and Embed. Systems*, 2017, pp. 189–210.

[11] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM Conf. on Computer & Communications Security*, 2013.

[12] M. Shiozaki, R. Hori, and T. Fujino, "Diffusion programmable device: The device to prevent reverse engineering." *IACR Cryptology ePrint Archive*, vol. 2014, p. 109, 2014.

[13] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[14] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2016.

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

[16] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid stt-cmos designs for reverse-engineering prevention," in *Proc. IEEE/ACM Design Automation Conf.* ACM, 2016, p. 88.

[17] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra, "Tpad: Hardware trojan prevention and detection for trusted integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 521–534, 2016.

[18] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *IEEE Int. Online Testing Symp.* IEEE, 2014, pp. 49–54.

[19] P. Subramanyan, N. Tsiskaridze, W. Li, A. Gascon, W. Y. Tan, A. Tiwari, N. Shankar, S. Seshia, and S. Malik, "Reverse engineering digital circuits using structural and functional analyses," *IEEE Trans. on Emerging Topics in Computing*, vol. 2, no. 1, pp. 63–80, 2014.

[20] B. Liu and B. Wang, "Embedded reconfigurable logic for asic design obfuscation against supply chain attacks," in *Proc. Design, Automation and Test in Europe*, 2014, pp. 1–6.

[21] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Camoperturb: secure ic camouflaging for minterm protection," in *Proc. Int. Conf. on Computer Aided Design*, 2016.

[22] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for ic protection," in *Proc. Int. Conf. on Computer Aided Design*, 2016, pp. 28:1–28:8.

[23] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. IEEE/ACM Design Automation Conf.*, 2012, pp. 83–89.

[24] S. Dasgupta and J. Langford, "A tutorial on active learning," in *Proc. Int. Conf. on Machine Learning*, 2009.

[25] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proc. Int. Conf. on Machine Learning*, 1994, pp. 148–156.

[26] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant, "A general lower bound on the number of examples needed for learning," *Information and Computation*, vol. 82, no. 3, pp. 247–261, 1989.

[27] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Proc. IEEE Great Lakes Symp. on VLSI*. ACM, 2017, pp. 179–184.

[28] W. Wei and B. Selman, "A new approach to model counting," in *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 2005, pp. 324–339.

[29] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. on Emerging Topics in Computing*, 2017.

[30] N. Sörensson and N. Eén, "Minisat 2.1 and minisat++ 1.0-sat race 2008 editions," *SAT*, p. 31, 2009.

[31] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Camoperturb: secure ic camouflaging for minterm protection," in *Proc. Int. Conf. on Computer Aided Design*. IEEE, 2016, pp. 1–8.

[32] B. L. Synthesis and V. Group, "Abc a system for sequential synthesis and verification," 2016, [Online] http://people.eecs.berkeley.edu/~alanmi/abc/.

[33] M. Yasin, A. Sengupta, M. Ashraf, M. Nabeel, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. ACM Conf. on Computer & Communications Security*, 2017, pp. 1–1.

[34] C. Bujtás, M. A. Henning, and Z. Tuza, "Transversals and domination in uniform hypergraphs," *European Journal of Combinatorics*, vol. 33, no. 1, pp. 62–71, 2012.

[35] N. Alon, "Transversal numbers of uniform hypergraphs," *Graphs and Combinatorics*, vol. 6, no. 1, pp. 1–4, 1990.

[36] V. Chvátal and C. McDiarmid, "Small transversals in hypergraphs," *Combinatorica*, vol. 12, no. 1, pp. 19–26, 1992.

[37] H. Zhou, "A humble theory and application for logic encryption," Cryptology ePrint Archive, Report 2017/696, 2017, https://eprint.iacr.org/2017/696.

[38] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-sat," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 342–347.

[39] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[40] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "Ttlock: Tenacious and traceless logic locking," in *Hardware Oriented Security and Trust (HOST), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 166–166.

[41] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *CHES*, vol. 4727. Springer, 2007, pp. 450–466.

[42] P. Jovanovic and M. Kreuzer, "Algebraic attacks using sat-solvers," in *Conference on Symbolic Computation and Cryptography*, 2010, p. 7.

[43] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proc. IEEE Great Lakes Symp. on VLSI*, 2017, pp. 173–178.

[44] H. Zhou, R. Jiang, and S. Kong, "Cycsat: Sat-based attack on cyclic logic encryptions," in *Computer-Aided Design (ICCAD), 2017 IEEE/ACM International Conference on*. IEEE, 2017, pp. 49–56.

Kaveh Shamsi received the B.S. degree in Electrical Engineering from the Sharif University of Technology, Tehran, Iran, in 2014, the M.S. degree in Computer Engineering from the University of Central Florida, Orlando, US in 2017. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at the University of Florida Florida, Gainesville, US under the supervision of Dr. Yier Jin. His research focuses on analog and digital circuit design, formal methods, and algorithms in hardware security. He is the recipient of the HOST'17 best paper award and the GLSVLSI'18 best paper candidate.

Meng Li received his B.S. degree in Microelectronics from Peking University, Beijing, China in 2013. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering, University of Texas at Austin, under the supervision of Prof. David Z. Pan. His research interests include hardware-oriented security, reliability, power grid simulation acceleration and deep learning. He received the first place in the grand final of ACM student research competition in 2018, best poster (Presentation) award in ASPDAC Ph.D. forum in 2018, gold metal in ACM ICCAD stduent research competition in 2017, and university graduate fellowship from UT Austin in 2013. He also received the best paper award in HOST'17 and best paper candidate in GLSVLSI'18.

Travis Meade Travis Meade received his B.S. degree in Mathematics from University of Central Florida, Orlando, Florida, US in 2012. In 2017 he received his Computer Science Ph.D. from University of Central Florida while working under Shaojie Zhang and Yier Jin. His research focuses on Intellectual Property protection, and Integrated Circuit Reverse Engineering for the sake of Hardware Security. He was a best paper award recipient at ASPDAC'16 for his work in Reverse Engineering. Additionally, he was a recipient for the best paper award at HOST'18.

David Z. Pan is currently the Engineering Foundation Professor at The University of Texas at Austin. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, physical design, analog design automation, and CAD for emerging technologies. He has published over 280 refereed technical papers, and is the holder of 8 U.S. patents. He has graduated over 20 PhD students who are now holding key academic and industry positions. He has served as a Senior Associate Editor for ACM Transactions on Design Automation of Electronic Systems (TODAES), an Associate Editor for a number of other journals. He has served in the Executive and Program Committees of many major conferences, including ASPDAC 2017 Program Chair and ICCAD 2018 Program Chair. He has received a number of awards for his research contributions, including the SRC 2013 Technical Excellence Award, DAC Top 10 Author in Fifth Decade, ASP-DAC Frequently Cited Author Award, 16 Best Paper Awards, among others. He is a Fellow of IEEE and SPIE.

Yier Jin is currently an associate professor in the ECE Department at the University of Florida. He received his PhD degree in Electrical Engineering in 2012 from Yale University after he got his B.S. and M.S. degrees in Electrical Engineering from Zhejiang University, China, in 2005 and 2007, respectively.

His research focuses on the areas of trusted embedded systems, trusted hardware intellectual property (IP) cores and hardware-software co-protection on computer systems. He proposed various approaches in the area of hardware security, including the hardware Trojan detection methodology relying on local side-channel information, the post-deployment hardware trust assessment framework, and the proof-carrying hardware IP protection scheme. He is also interested in the security analysis on Internet of Things (IoT) and wearable devices with particular emphasis on information integrity and privacy protection in the IoT era. He is awarded the DoE Early CAREER Award in 2016 and is the best paper award recipient of DAC'15, ASP-DAC'16, HOST'17, ACM TODAES'18, and GLSVLSI'18.