AppSAT: Approximately Deobfuscating Integrated Circuits

Kaveh Shamsi*, Meng Li†, Travis Meade*, Zheng Zhao†, David Z. Pan†, and Yier Jin*
*Department of Electrical and Computer Engineering, University of Central Florida
†Department of Electrical and Computer Engineering, University of Texas at Austin
{kaveh, travm12}@knights.ucf.edu, {meng_li,zzhao,dpan}@utexas.edu, yier.jin@eecs.ucf.edu

***Abstract*—In today's diversified semiconductor supply-chain, protecting intellectual property (IP) and maintaining manufacturing integrity are important concerns. Circuit obfuscation techniques such as logic encryption and IC camouflaging can potentially defend against a majority of supply-chain threats such as stealthy malicious design modification, IP theft, overproduction, and cloning.**

Recently, a Boolean Satisfiability (SAT) based attack, namely the SAT attack has been able to deobfuscate almost all traditional circuit obfuscation schemes, and as a result, a number of defense solutions have been proposed in literature. All these defenses are based on the implicit assumption that the attacker needs a perfect deobfuscation accuracy which may not be true in many practical cases. Therefore, in this paper by relaxing the exactness constraint on deobfuscation, we propose the AppSAT attack, an approximate deobfuscation algorithm based on the SAT attack and random testing. We show how the AppSAT attack can deobfuscate 68 out of the 71 benchmark circuits that were obfuscated with state-of-the-art SAT attack defenses with an accuracy of $\frac{1}{2^n}$, n being the number of inputs. AppSAT shows that with current SAT attack defenses there will be a trade-off between exact-attack resiliency and approximation resiliency.

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 sometimes 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]. The primary concerns are: 1) malicious modification of the design, 2) reverse-engineering by delayering and imaging the IC to the end goal of gaining critical information for exploitation, or the theft of intellectual property (IP), and 3) cloning, replicating or overproduction by the foundry.

Circuit obfuscation is among the primary design-stage methods for protecting against the above attacks [1]. Logic encryption/locking [2] and IC Camouflaging [3] are two of the main circuit obfuscation techniques. Logic encryption or key-based obfuscation is based on corrupting the output of the circuit with additional key-inputs to render the circuit useless without a secret key. IC camouflaging is a layout level technique based on creating indistinguishable layout structures for creating obscurity. These techniques can potentially provide a layer of protection against most of the supply chain attacks. For instance, with logic encryption, targeted malicious modification of the design is hindered through the obscurity of the obfuscated circuit and the foundry cannot overproduce the design without the key. In addition, both IC camouflaging and logic encryption hamper IC reverse-engineering.

However, the security of both these obfuscation schemes is questioned by *deobfuscation* attacks [4]–[7]. The most recent and strongest deobfuscation attack is one utilizing Boolean Satisfiability (SAT) solvers which we refer to as the *SAT attack* throughout this paper [4], [5]. This attack utilizes input-output observations (queries) from an unlocked or functional circuit to resolve the key bits or camouflaged layout functionality.

Consequently, a number of defenses against the SAT attack have been recently proposed in literature [6]–[8]. These solutions exponentially increase the number of queries required by the SAT attack to complete successfully. We refer to these schemes as *compound* obfuscation schemes, since they combine low output corruptibility blocks (*point-functions*) with traditional, high output corruptibility obfuscation to obtain a circuit that requires exponential queries to be resolved, while maintaining a sufficiently high output corruptibility as well.

All these defenses are based on the implicit assumption that an attacker needs to obtain a perfect accuracy when learning the functionality of the circuit. This may not be true for many practical cases. For instance, for instruction decoders, the circuit has to operate correctly for a possibly known subset of all input patterns.

In this paper we relax the exactness constraint and present a query based deobfuscation attack by allowing for approximation in the attack framework. This new circuit deobfuscation attack is called AppSAT throughout the paper. AppSAT can deobfuscate the high corruptibility portions of any compound scheme, effectively reducing the defense to a low corruptibility obfuscation which itself is an exponentially good approximation of the original circuit.

Contributions. This paper specifically delivers the following contributions:

- We propose AppSAT for approximate deobfuscation based on the SAT attack augmented with random querying and intermediate error estimation. We show the effectiveness of the attack by running it on 71 ISCAS and MCNC benchmark circuits obfuscated with the Anti-SAT [7] compound scheme. We show that the attack is capable of perfectly deobfuscating the high-corruptibility portion of this defense for 68 of the benchmark circuits.
- We present the approximate attack framework as an efficient way for quantitatively evaluating the resiliency of circuit obfuscation schemes. We specifically study a number of gate-level logic encryption schemes with respect to this measure.

Organization. The paper is organized as follows. Section II provides necessary background and preliminaries on obfusca-

978-1-5386-3929-0/17/$31.00

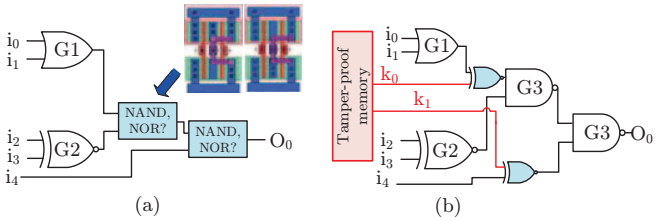

Figure 1: (a) IC camouflaging by replacing gates with camouflaged gates. (b) Logic encryption with tamper-proof key bits.

tion and deobfuscation. Section III presents the framework and the algorithm of the AppSAT attack. Section IV presents an experimental evaluation. Section V presents discussions and Section VI concludes the paper.

II. Background

A. Circuit Obfuscation

IC camouflaging [3] techniques are physical-design oriented. Some notable methods include dummy connections, doping alterations, and dummy cells/wires. A via with a middle gap can deceivingly appear to be connected during reverse-engineering serving as a dummy connection [9]. Doping based approaches [10] 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]. Figure 1 (a) shows gate-level camouflaging where gates in the circuit are replaced with dummy-connection camouflaged gates which look alike but have different functionalities.

Logic encryption, as seen in Figure 1 (b), is based on inserting additional inputs, namely key-inputs (k_0 and k_1 in Figure 1 (b)), to the circuit, and have them corrupt the output functionality. The IP owner configures the correct key values through a tamper-proof memory once the chips return from fabrication. Unlike IC camouflaging, logic encryption helps protect against a malicious foundry as well[2]. Traditional methods for logic encryption in literature include: XOR/XNORing wires selected from the circuit with key-bits [2], [11], [12], replacing gates in the circuit with look-up-tables (LUTs) that store key-bits [13], [14], inserting multiplexors (MUXs) [11] or switch-boxes (SB) [15] that are configured by key-bits, as well as other gate insertion approaches [16].

B. The SAT Attacks

Consider the original circuit to be a Boolean function from input space $I = \{0,1\}^n$ to $O = \{0,1\}^m$, denoted by $C_o : I \rightarrow O$. Almost all logic encryption and camouflaging techniques can be modeled as transforming C_o, to an augmented function, $C_e : I \times K \rightarrow O$, where K is the key space, and there exists a correct key value $k_* \in K$ such that $\forall i \in I,\ C_e(i, k_*) = C_o(i)$. In this way, dummy connections can be modeled as MUXes, camouflaged gates can be modeled as a set of gates arbitrated by a MUX configured by key-bits, and logic encryption schemes can be directly converted.

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

[2]Note that with IC camouflaging the fabrication facility is aware of the true functionality of dummy contacts and other camouflaged layouts.

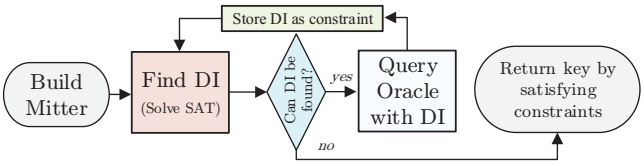

Figure 2: The exact SAT attack flow.

The SAT attack is an algorithm that finds a k_* given 1) the Boolean expression for C_e which the attacker obtains by delayering, imaging, and reconstructing the netlist of the obfuscated chip, and 2) input-output access to C_o. Input-output or oracle access to C_o means that the attacker can query a black-box with i to get $C_o(i)$. Therefore, we categorize the SAT attack as an *oracle-guided* attack. Per Figure 2 the attack begins by using a SAT-solver to satisfy a mitter equation, $C_e(i, k_1) \neq C_e(i, k_2)$, with i_0, k_1, k_2. The input i_0 that satisfies the mitter is referred to as a discriminating input (DI). 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 equation and the algorithm repeats the DI finding with tighter constraints. Once the mitter+constraints SAT problem can no longer be satisfied, solving the constraints alone will return $k_1 = k_2 = k_*$. The attack is extremely powerful against gate-replacement camouflaging [5] and all logic encryption schemes mentioned in Section II-A [4].

C. Point-function Circuit Obfuscation Schemes

A point-function denoted as $P(i, k)$ outputs 1 when $i = k$ and 0 otherwise. A property of this function is that trying to find k by querying i and observing the output, takes an average of $\frac{2^n}{2}$ queries. The P function can be implemented by inverting the inputs of an n-input AND gate based on a secret [6], [17]. Carefully incorporating a point-function into a circuit can result in the exponential increase of the number of DIs required to execute the SAT attack with respect to the number of inputs to P. An example is shown in Figure 3.

Based on point-functions, a number of circuit obfuscation methods have been proposed in literature. Li et al. [18] proposed to find an AND-tree in the circuit and camouflage its inputs, or to XOR a wire from the circuit with a camouflaged P function. Yasin et al. [6] proposed a logic encryption scheme called SARLock based on inserting a tree-like structure into the circuit. CamouPerturb, [17] presented by Yasin et al., is a camouflaging technique based on corrupting and correcting the output of a cone using the P function. Anti-SAT [7], a logic encryption scheme by Xie and Srivastava, flips the output of the circuit for a single input for every possible key, except for the correct key using complementary P blocks.

D. Increasing Output Corruptibility

For a circuit obfuscation scheme we can define the *output corruptibility* measure, Cr, as:

$$\mathsf{Cr} = \underset{i \in I,\ k \in K}{\mathsf{Pr}}[C_e(i,k) \neq C_o(i)].$$

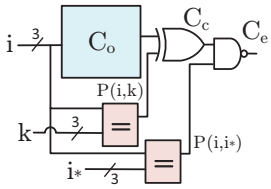

Figure 3: An exmaple of a point-function obfuscation scheme.

Table I: Truth-table for the circuit in Figure 3. 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

Point-function obfuscation schemes, if used alone, result in a very low *output corruptibility*. As seen from Figure 3 the attacker can randomly select any *incorrect* key value, and the obfuscated 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 all proposed point-function schemes, where n is the width of the P function. Since a low corruptibility is not desirable, almost all the schemes discussed in Section II-C propose to combine point-function obfuscation with traditional circuit obfuscation methods. This is why we refer to them as compound schemes. Since most traditional methods such as XOR/XNOR-based [2], [9], [11], [12], MUX [11] or SB-based [15], either were designed to maximize output corruptibility, or naturally result in a high corruptibility, adding them will effectively improve Cr for the overall obfuscated circuit.

III. Approximate Deobfuscation

As was noted in [18] the class of oracle-guided deobfuscation attacks can be modeled with an specific problem in machine-learning called active-learning [19]. In active-learning terminology the *learner* intends to find a *target function*, C_o, within a *hypothesis space*, $\mathcal{C}$, 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 a querying strategy that minimizing the number of queries required to learn the target function [19].

Among the different active-learning querying strategies, one ensemble referred to as uncertainty-sampling or query-by-disagreement (QBD) [20] 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 $\mathcal{C}$ 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.

In this paper however, we relax the exactness constraint on deobfuscation. This allows us to evaluate circuit obfuscation in a new light, showing the weakness of existing compound schemes. One way for formulating approximate learning problems is the *probably-approximately-correct* (PAC) setting [21], 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 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 it is clear that we should evaluate approximate attacks based on their accuracy (ϵ) and success rate (λ) with respect to number of queries. In this paper we study compound obfuscation schemes from this perspective. We specifically show that when attacking compound obfuscation 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 obfuscation schemes. The remainder of this section we discuss this in more detail by presenting an approximate deobfuscation algorithm.

A. A SAT-based Approximate Attack (AppSAT)

To model approximate deobfuscation with PAC-learning, let the hypothesis space, $\mathcal{C}$, be the set of realizable functions by varying the key vector. We define $\mathcal{K} : K \rightarrow \mathcal{C}$ to be a mapping from the key vector to functions in the hypothesis space, for which $\mathcal{K}(k*) = C_o$ where C_o is the target circuit. We will write the function that $\mathcal{K}(k)$ returns, as $C_e(x, k)$. The SAT attack begins by satisfying the mitter circuit with a DI, 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 DIs until no more DIs can be found, we can build an approximate SAT attack by terminating the attack in any early step, i.

Let us define $\mathcal{L}_i$ to be the set of input-output observation (queries) collected until step i and $\mathcal{V}_i$ be the version space consistent with $\mathcal{L}_i$. It can be shown that $|\mathcal{V}_i| < |\mathcal{V}_j|$ for all $i > j$. Therefore, the SAT attack is converging on the target function with each step by shrinking the version space. At any step, i, of the algorithm we can use the SAT-solver to find a key vector consistent with $\mathcal{L}_i$ by satisfying the conjunction of all queries. This is equivalent to picking a function from $\mathcal{V}_i$. We denote the error probability of this function with ϵ_i. Different functions in $\mathcal{V}_i$ 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 high error rate. That is, $\epsilon_i < \epsilon_j$ may not be true for every $i > j$. However, on average 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. One approach is to stop when the error (ϵ) falls below a certain limit. However, calculating ϵ precisely would require exponential queries itself. This is because exactly finding ϵ requires comparing the truth tables of the hypothesized function and the target function. Therefore, heuristic methods for estimating the error must be used for larger functions [19].

For the circuit deobfuscation problem, we first note that it is straightforward to analytically represent the error probability

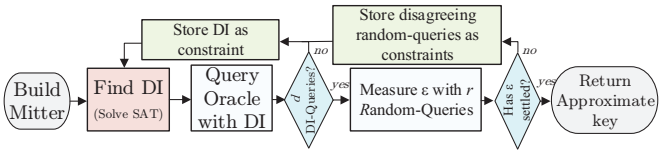

Figure 4: The overall AppSAT algorithm flow.

of point-function schemes specifically. Since any randomly selected key results in an incorrect output for only a single input pattern, we can state that for any $i > 0$, we have $\epsilon_i \sim O(\frac{1}{2^n})$. As for traditional schemes or compound schemes however, finding an analytical, and circuit-independent error value is difficult. In this paper we limit our AppSAT attack to randomly querying input patterns in order to estimate ϵ after every d DI queries, and stop the attack if ϵ stays below a threshold for more than a certain number of times (settlement-threshold). We show that this is sufficient for deobfuscating existing compound schemes with high accuracy. In Section V we will discuss other potential approaches as well.

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 DI 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 DIs. We refer to this technique as query reinforcement. Specifically, the random samples that resulted in a disagreement will be added to the SAT formula as new constraints. Our experimental results show that this approach significantly improves the running time of the attack for a certain accuracy goal. With these discussions in mind Algorithm 1 shows the overall routine of the AppSAT attack and Figure 4 shows the flow of the algorithm.

D. Compound Obfuscation Under Approximate Attacks

As we discussed earlier, due to the low output corruptibility of point-function obfuscations, these methods are often combined with high corruptibility obfuscations to form the compound obfuscation scheme. An important result of this paper is that adding a high corruptibility obfuscation to the point-function schemes does not contribute to the overall security. The AppSAT attack is capable of deobfuscating the traditional portions of the compound obfuscation as if the point-function scheme was not present. This observation is key to utilizing the AppSAT attack to defeat various flavours of compound schemes.

To explain this phenomena, consider the truth table of the point-function scheme shown in Table I. Per Table I, in the beginning of the algorithm all input patterns are potential DIs. In other words, the point-function obfuscation imposes a loose constraint on which inputs should be queried. A high corruptibility obfuscation 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 DIs is equivalent to running the attack on a standalone high corruptibility obfuscation scheme as if the point-function was not present. As we will see in Section IV-B, 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.

Algorithm 1 Given oracle access to C_o and the expression for C_e return an approximate key k_1.

function ApproxSATDecrypt(C_e, C_o as black-box)
  $j \leftarrow 0$
  $M \leftarrow C_e(i, k_1) \neq C_e(i, k_2)$
  $F_j \leftarrow true$
  while $F_j \wedge M$ is solvable **do**
    Solve $F_j \wedge M$ with i_j, k_1, k_2
    $O_j \leftarrow C_o(i_j)$
    $F_{j+1} \leftarrow F_j \wedge (C_e(i_j, k_1) = O_j) \wedge (C_e(i_j, k_2) = O_j)$
    $j \leftarrow j + 1$
    satisfy F_j with k_1, k_2
    every n rounds **do**
    for $x \in RandomPatterns$ **do**
      if $C_e(k_1, x) \neq C_o(x)$ **then**
        $FailedPatterns \leftarrow FailedPatterns + 1$
        $F_{j+1} \leftarrow F_{j+1} \wedge (C_e(k_1, x) = C_o(x))$
      end if
    end for
    $\epsilon \leftarrow \frac{FailedPatterns}{InputSpaceSize}$
    if $\epsilon < ErrorThreshold$ **then**
      $SettleCount \leftarrow SettleCount + 1$
      if $SettleCount > SettleThreshold$ **then**
        return k_1 as approximate key
      end if
    else
      $SettleCount \leftarrow 0$
    end if
  end while
  satisfy F_j with k_1
return k_1 as exact key
end function

IV. Experimental Results

A. Methodology

For our experimentation, we implemented both attacks and defenses in a C++ framework. The framework utilizes Minisat [22] for SAT-solving and includes the conventional exact SAT attack augmented with intermediate key vector extraction, error estimation/calculation, 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. We launch AppSAT on compound obfuscation schemes on benchmark circuits and and also compare different SAT attacks and defenses on small circuits evaluating query reinforcement and the approximation resiliency of high corruptibility schemes.

B. Evaluation Results

We evaluated Algorithm 1 on the benchmark circuits that are listed in Table II. The benchmark circuits were obfuscated with Anti-SAT[3] integrated with random XOR/XNOR gate insertion [2]. Then the AppSAT attack was launched with the following parameters: after every 12 iterations of the SAT

[3]While our experiments are based on the Anti-SAT method, the attack can easily be used against any other point-function scheme including CamouPerturb [17], SARLock [6], and Li's [18] Methods.

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

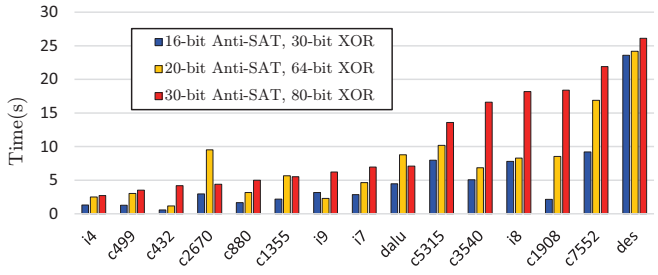

Figure 5: Running time for decrypting ISCAS and MCNC combinational benchmarks.

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 times are shown in Figures 5 and 6. All tests were launched in parallel. The experiment was completed in 10 minutes, memory usage remained below 8GB, and no circuit was queried more than 1200 times. The returned key value consists of the XOR/XNOR high corruptibility key bits and the Anti-SAT key bits. We utilized the equivalence checker from [4] to verify the correctness of the XOR/XNOR key bits. For 41 out of the 43 combinational circuits the XOR/XNOR key bits were a perfect match. 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.

In another experiment, we generated a set of random circuits

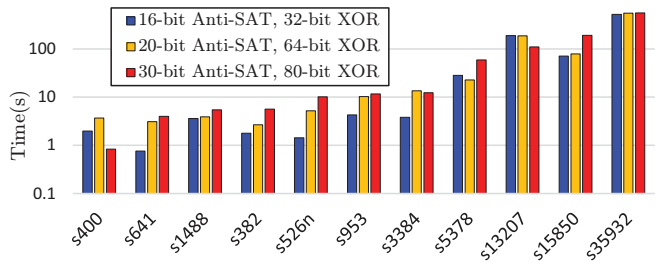

Figure 6: Running time for decrypting unrolled ISCAS sequential benchmarks.

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 [23] to synthesize randomly generated truth-tables resulting in circuits with 200-300 gates, and then obfuscated them using fault based XOR/XNOR [11] and MUX [11] obfuscation, clique based XOR/XNOR [12] obfuscation, and random XOR/XNOR obfuscation [2] using [4]. We performed the exact SAT attack on these circuits and recorded ϵ with each iteration as seen in Figure 7a. We then added a 9-input point-function using Anti-SAT's approach and performed the exact SAT attack again while recording the error which is shown in Figure 7b. 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 7c showing the importance of random query reinforcement. Furthermore, the fault based XOR/XNOR [24] method shows a higher approximation resiliency as seen from its square shape ϵ plot.

V. Discussions

We provide two discussions regarding the AppSAT attack:

Deobfuscating Point-Functions: a main result of the AppSAT attack is that it is able to reduce the problem of deobfuscating a compound obfuscation to one on a point-function obfuscation. While this in many cases is sufficient for the attacker the question is whether a standalone point-function scheme can be broken. As for Anti-SAT, authors in [25] proposed a signal probability based attack to find and remove the Anti-SAT block [25]. In our attack framework we were able to observe that for Anti-SAT + XOR/XNOR obfuscation schemes, the key bits that lead to the XOR/XNOR converge after a few iterations as seen in Figure 8. This can assist the probability based attack in finding the Anti-SAT key bits and removing the obfuscation block allowing full recovery of the original circuit.

CamouPerturb [17] on the other hand, prevents removal attacks by synthesizing the original circuit with the output being flipped for a particular input pattern, i_*, and then adds P blocks to *correct* the output for this pattern. If the attacker finds and removes the P block, she is left with a circuit that will fail for a single unknown input pattern, i_*. In this case we first note that large P blocks will incur large overhead and are difficult to hide, and the attacker can potentially brute-force point-functions of less than 30 inputs. For a 30-input P block, if 10^4 input patterns are tested every second, i_* can be found in 30 hours at worst. Secondly, even without finding i_* the attacker may be able to build a functional equivalent of C_o. For instance, in any point-function scheme a majority vote among $C_e(i,k)$ for three different k values will always return the correct result.

Improving the Attack: The approximate attack can be optimized by tuning the different parameters in the algorithm with respect to the size of the circuit and the obfuscation secret. Furthermore, the querying and error estimation can

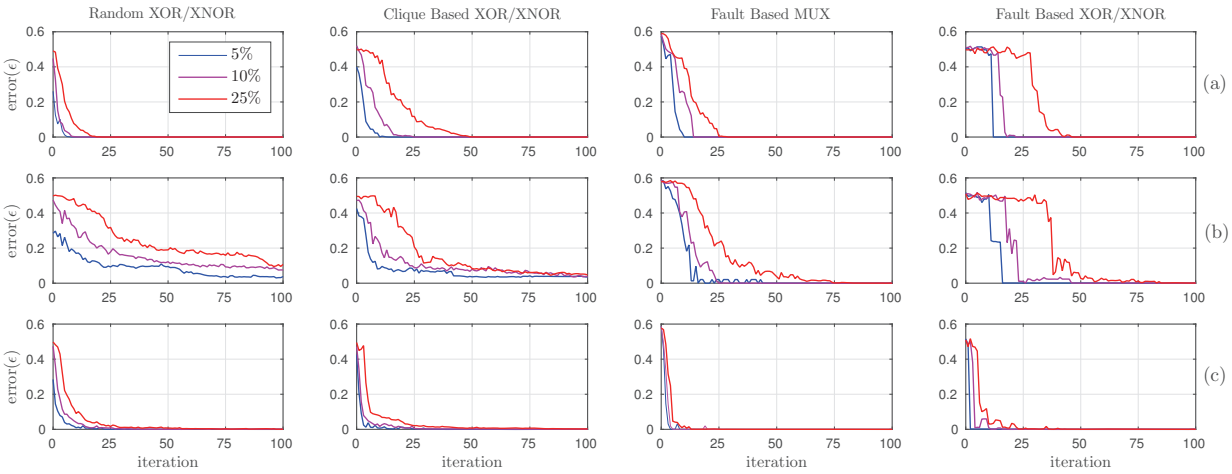

Figure 7: Exact SAT attack error value (ϵ) in every iteration for: a) high corruptibility obfuscation schemes, and b) the circuits in (a) augmented with an Anti-SAT block. Figures in the (c) row show the error trend when 10 random queries were reinforced in each step resulting in less iterations and time.

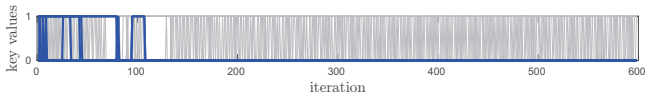

Figure 8: Key bit convergence. The blue plots are the XOR/XNOR key bits all of which settle after a 100 iterations while the rest continue to change.

be improved by using more expressive input patterns such as automatically generated test patterns for C_e. In addition, test patterns of C_o, or specific input patterns, such as known instructions for an instruction decoder, can help approximate C_o on input patterns that are important to the attacker.

VI. Conclusion

In this paper an approximate flow was introduced for attacking state-of-the-art circuit obfuscation schemes and the effectiveness was shown on benchmark circuits. With the introduction of this attack, the main challenge in circuit obfuscation seems to be breaking the trade-off between output corruptibility and resiliency to oracle-guided attacks.

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," 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 *DAC*. IEEE, 2014, pp. 1–5.

[4] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *HOST*. IEEE, 2015, pp. 137–143.

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

[6] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," 2016, pp. 236–241.

[7] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 127–146.

[8] K. Shamsi, W. Wen, and Y. Jin, "Hardware security challenges beyond cmos: Attacks and remedies," in *(ISVLSI), 2016 IEEE*. IEEE, 2016, pp. 200–205.

[9] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," 2013.

[10] 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.

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

[12] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," 2012, pp. 83–89.

[13] A. C. Baumgarten, "Preventing integrated circuit piracy using reconfigurable logic barriers," 2009.

[14] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid stt-cmos designs for reverse-engineering prevention." ACM, 2016, p. 88.

[15] 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.

[16] 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." IEEE, 2014, pp. 49–54.

[17] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Camoperturb: secure ic camouflaging for minterm protection." ACM, 2016, p. 29.

[18] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for ic protection," in *International Conference On Computer Aided Design (ICCAD)*, 2016, pp. 28:1–28:8.

[19] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[20] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," 1994, pp. 148–156.

[21] 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.

[22] N. Sorensson and N. Een, "Minisat v1. 13-a sat solver with conflict-clause minimization," *SAT*, vol. 2005, p. 53, 2005.

[23] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 24–40.

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