Resynthesis-based Attacks Against Logic Locking

Felipe Almeida, Levent Aksoy, Quang-Linh Nguyen, Sophie Dupuis, Marie-Lise Flottes, Samuel Pagliarini

Resynthesis-based Attacks Against Logic Locking

Felipe Almeida[†], Levent Aksoy[†], Quang-Linh Nguyen[‡], Sophie Dupuis[‡], Marie-Lise Flottes[‡], and Samuel Pagliarini[†]
[†]Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[‡]LIRMM, University of Montpellier, Montpellier, France
Email: [†] {felipe.almeida, levent.aksoy, samuel.pagliarini}@taltech.ee
Email: [‡] {quang-linh.nguyen, sophie.dupuis, marie-lise.flottes}@lirmm.fr

***Abstract*—Logic locking has been a promising solution to many hardware security threats, such as intellectual property infringement and overproduction. Due to the increased attention that threats have received, many efficient specialized attacks against logic locking have been introduced over the years. However, the ability of an adversary to manipulate a locked netlist prior to mounting an attack has not been investigated thoroughly. This paper introduces a resynthesis-based strategy that utilizes the strength of a commercial electronic design automation (EDA) tool to reveal the vulnerabilities of a locked circuit. To do so, in a pre-attack step, a locked netlist is resynthesized using different synthesis parameters in a systematic way, leading to a large number of functionally equivalent but structurally different locked circuits. Then, under the oracle-less threat model, where it is assumed that the adversary only possesses the locked circuit, not the original circuit to query, a prominent attack is applied to these generated netlists collectively, from which a large number of key bits are deciphered. Nevertheless, this paper also describes how the proposed oracle-less attack can be integrated with an oracle-guided attack. The feasibility of the proposed approach is demonstrated for several benchmarks, including remarkable results for breaking a recently proposed provably secure logic locking method and deciphering values of a large number of key bits of the CSAW'19 circuits with very high accuracy.**

***Index Terms*—Logic locking, resynthesis, EDA tools, oracle-less and oracle-guided attacks.**

I. Introduction

Due to the globalized integrated circuit (IC) supply chain, serious security threats, such as hardware Trojans, piracy, overbuilding, reverse engineering, and counterfeiting, have emerged [1]. Many defense techniques, such as watermarking [2], digital rights management [3], metering [4], and logic locking [5], have been introduced over the years to deal with these threats. Among those, logic locking stands out by being a well-established technique and by offering protection against a diverse array of adversaries [6]. Logic locking inserts additional logic driven by key bits so that the circuit behaves as expected only when the secret key is applied.

On the other hand, many efficient attacks have been introduced to overcome the defenses built by logic locking [7]. However, the impact of an electronic design automation (EDA) tool on the manipulation of the locked netlist **before** performing an attack has not been investigated thoroughly. In this work, we explore if EDA tools can be used to make a locked circuit vulnerable to existing logic locking attacks. Thus, the main contributions of this work are three-fold: (i) we introduce a resynthesis procedure that is a **pre-attack** step, where functionally equivalent but structurally different locked circuits are generated by resynthesizing the original locked circuit using different optimization parameters and delay constraints in order to create structural vulnerabilities that can be exploited by existing attacks; (ii) we present an oracle-less (OL) **resynthesis-based attack**, which applies the prominent SCOPE attack [8] to these resynthesized circuits and gathers all its solutions to discover the secret key; (iii) we show that our OL attack can be **combined** with a traditional oracle-guided (OG) attack for further improving the number of correctly deciphered key bits. The last contribution is essential since we consider circuits from the CSAW'19 contest – these circuits compound the use of two logic locking techniques at the same time.

The main finding of this work is that the use of many resynthesized locked circuits enables us to discover values of more key bits, and even the whole key, when compared to a single attack mounted on the original locked netlist.

The remainder of this paper is organized as follows: Section II presents the background concepts and related work. The resynthesis process and the proposed attacks are described in Section III. Experimental results are given in Section IV. Finally, Section V concludes the paper.

II. Background

A. Logic Locking and Threat Models

The procedure of logic locking is applied at the gate-level in the IC design flow, as shown in Fig. 1. Note that the layout of the locked circuit is sent to the foundry without revealing the secret key. After the locked IC is produced and delivered to the design house, the values of the secret key are stored in a tamper-proof memory, before the functional IC is sent to the market.

It is assumed that the gate-level netlist of the locked circuit can be obtained directly by an untrusted foundry or by reverse-engineering a functional IC obtained from the open market. An adversary can also use the functional IC

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 952252 (SAFEST).

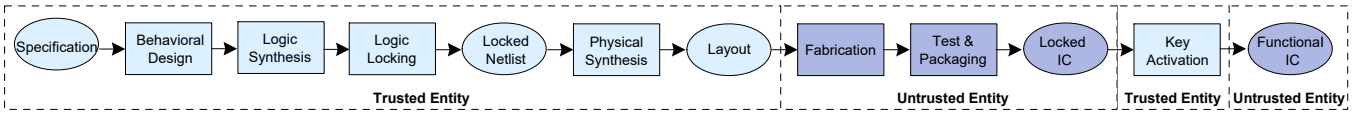


Fig. 1. Conventional logic locking in the IC design flow (adapted from [6]).

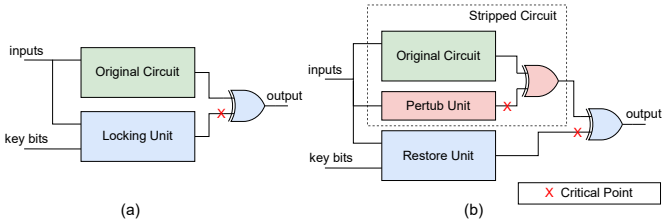


Fig. 2. SAT-resilient logic locking methods: (a) SFLT; (b) DFLT.

programmed with the secret key as an oracle to apply inputs and observe outputs. Thus, in logic locking, there are generally two threat models: OL and OG. In the OL threat model, only the gate-level netlist of the locked circuit is available to the adversary. The adversary has both the netlist of the locked circuit and the functional IC in the OG threat model.

B. Related Work

After the introduction of random logic locking (RLL) using XOR/XNOR gates in [9], earlier work focused on different types of key gates, such as AND/OR, multiplexors, and look-up tables, taking into account the hardware complexity of the locked circuit [5]. However, the OG satisfiability (SAT)-based attack [10] overcame all the defenses existing at that time. Note that the SAT-based attack iteratively finds distinguishing input patterns (DIPs) that rule out wrong keys. To thwart the SAT-based attack and its variants, circuits are locked using a point function that sets a limit on the number of wrong keys which a DIP can eliminate, forcing these attacks to explore an exponential number of queries [6], [11]–[14].

The SAT-resilient methods can be categorized into two groups: single-flip locking technique (SFLT) and double-flip locking technique (DFLT), as shown in Fig. 2. An SFLT has only one critical point, which is responsible to corrupt a protected output under a specific input pattern. Under this category, SARLock [15] adds a comparator and a masking circuit connected with the original netlist in a way that it generates a corruption on one input pattern. Anti-SAT [11] utilizes two complementary AND gate trees, whose output is merged with the original circuit. CASLock [12] is based on the same concept of Anti-SAT, however it uses both AND and OR gates. SKG-Lock [14] uses decoy key bits and provides a tunable output corruption. Note that SFLTs are susceptible to removal attacks [16]–[18]. If an attacker can identify this single critical point, he/she can split the design into a recovered netlist (original) and the locking unit.

A DFLT has two critical points, one that connects the original netlist with a perturbation unit and another one that connects the output of the stripped circuit with the restore unit. Under this category, stripped functionality logic locking (SFLL) [6], [13] initially corrupts an output based on an input combination in the perturbation unit and then, corrects this output only when the secret key is applied in the restore unit. Note that a removal attack becomes inefficient for a DFLT since the original circuit is mixed with the perturbation unit, even though it can easily identify the restore unit. However, there exist efficient structural attacks developed for DFLTs [19]–[22].

Alternative locking techniques have also been introduced. In [23], a technique, which has more than two critical points, called the multi-flip locking technique (MFLT), was proposed. However, it leads to a significant increase in area, power dissipation, and delay when compared to other techniques. Compound logic locking techniques were proposed to overcome the main drawback of a SAT-resilient technique, i.e., its low output corruptibility as can be observed in Fig. 2, by locking a design using both low and high output corruptibility techniques, such as SFLL and RLL, respectively [24]. Recently, efficient attacks have also been introduced against compound logic locking [25], [26].

Moreover, the OL attacks explore patterns in the structure of a locked netlist using statistical analysis [8], [27], [28]. For example, the SCOPE attack [8] is an unsupervised constant propagation technique, which analyzes each key bit of the locked design for critical features that can reveal its correct value after it is assigned to logic 0 and 1 value. These critical features include area, power dissipation, delay, and many other circuit characteristics obtained by a synthesis tool. These features are analyzed using linear regression and machine learning based clustering.

III. Proposed Resynthesis-based Attack

This section describes our resynthesis-based attack in detail. We assume a scenario, where the design house is the only trusted entity. An attacker has all possible reverse engineering, synthesis, and computing tools and has an access to the gate-level locked netlist and the functional IC. In this section, we initially introduce the pre-attack stage, where the locked circuit is resynthesized using different synthesis parameters, leading to a large number of structurally different netlists with the same functionality. Then, we present the OL attack that utilizes these resynthesized netlists in order to find the secret key. Finally, in order to handle the compound logic locking efficiently, we present its modified version, where our proposed OL attack cooperates with an OG attack.

A. The Pre-attack Step: Resynthesis of the Locked Netlist

The locked circuit is synthesized multiple times using a different script each time, where the synthesis parameters are explored in a systematic way. We use the following parameters to increase the number of resynthesized locked circuits:

Synthesis Effort: In a synthesis tool, logic optimizations can be applied with different efforts at different synthesis stages. This flexibility enables a designer to explore the trade-off between the quality of results and run time. The following efforts are considered at the given synthesis stage: low, medium, and high at generic transformations (*syn_gen*); low, medium, and high at mapping (*syn_map*); and low, medium, high, and extreme at optimization (*syn_opt*).

Delay Constraint: To meet performance targets, delay constraints are used to guide the synthesis tool. We initially resynthesize the locked circuit without a delay constraint and find the delay of its critical path, i.e., dcp. Then, in an interval between 0 and dcp, $d-1$ points, which are computed as $(dcp/d)i$ with $1 \leq i \leq d-1$, are set as delay constraints. Note that d is set to 5 in order to generate a large number of resynthesized circuits. Even though some delay constraints are impossible to meet, the synthesis tool *always* generates a netlist equivalent to the original one in terms of functionality.

Maximum Transition: The transition time of a net in a circuit is defined as the longest time required for its driving pin to change its logic value. The maximum transition value was chosen to be 5%, 10%, and 15% of the delay constraint for all the nets in the locked circuit to explore different resynthesized circuits.

Key Constraints: To direct the synthesis tool to work intensively on the paths that include the keyed logic, a delay constraint, which is impossible to be satisfied, can also be used. In this case, we force the delay between all key bits and all primary outputs to be 1 ps.

Thus, the combination of parameters given above generates $3 \times 3 \times 4 \times 5 \times 3 \times 2 = 1080$ netlists. We eliminate the resynthesized circuits with identical characteristics and keep only the unique ones. Additionally, we prevent the use of XOR/XNOR gates, which can be problematic for the SCOPE attack, during technology mapping. Note that our resynthesis methodology aims to generate different versions of the locked circuit, making it more vulnerable to existing attacks. Thus, any existing attack, either OL or OG, may potentially benefit from this pre-attack strategy to discover the secret key. The resynthesis process is automated for a commercial synthesis tool in a Perl script. This script, which can be modified for other synthesis tools, is available at *https://github.com/Centre-for-Hardware-Security/*.

B. Attacks on the Resynthesized Netlists

Time-efficient attacks are chosen in order to handle a large number of resynthesized circuits. In our OL resynthesis-based attack, SCOPE [8] is used to predict the values of key bits. In its modified version developed for compound logic locking, a query attack is used to find the values of key bits in a deterministic way.

1) Proposed OL Attack: SCOPE is applied to each resynthesized locked circuit and a solution is found. Note that this solution may return a logic 0, 1, or an unknown value for a key bit. Then, the values of key bits deciphered for each netlist are merged into a single solution that represents the overall guess. To do so, for each key bit, k_i with $1 \leq i \leq p$, where p denotes the number of key bits, we initially count the number of solutions, where k_i is deciphered as logic 0 and 1, denoted as dk_i^0 and dk_i^1, respectively. Then, if $dk_i^0 > dk_i^1$ or $dk_i^1 > dk_i^0$, the value of k_i is determined to be 0 or 1, respectively. Otherwise, in the case of a tie, the value of k_i is decided to be unknown.

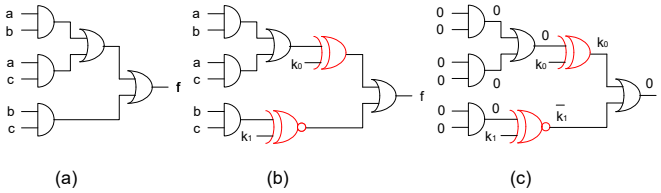


Fig. 3. (a) Majority circuit; (b) Locked majority circuit; (c) Constant propagation on the locked majority circuit.

2) Proposed OG Attack: In order to handle a large number of resynthesized netlists efficiently, we introduce a SAT-based query attack, which can determine the actual values of individual key bits. Note that traditional SAT-based attacks rather attempt to find the whole secret key, which increases the computational effort significantly. In this attack, we initially find queries, i.e., values of inputs of the oracle circuit, using two techniques. The first technique uses the ATPG tool Atalanta [29] to find test patterns for the stuck-at-fault of each key bit on the locked circuit and stores the values of the related primary inputs as queries. The aim is to find input patterns that can propagate each key bit to a primary output, making it observable. The second technique finds queries randomly. The aim is to find input patterns that may make multiple key bits observable at primary outputs. In our experiments, we generate a total of $2p$ queries, where p denotes the number of key bits.

Then, we describe the locked circuit in a conjunctive normal form (CNF) formula $\mathbb{C}$ by expressing each gate in its CNF. Each query is applied to the oracle and the values of primary outputs are obtained. Then, the related input and output values are assigned to the associated nets in the locked circuit, the constant values of these nets are propagated, and the Boolean equations including key bits are derived in a CNF formula $\mathbb{E}$. The SAT problem including the locked circuit in CNF, i.e., $\mathbb{C}$, is augmented with these equations, i.e., $\mathbb{C} = \mathbb{C} \wedge \mathbb{E}$. After all the queries are considered, the SAT problem $\mathbb{C}$ is solved using a SAT solver and the values of key bits are determined. Note that the locked circuit with the found values of key bits behaves exactly the same as the oracle under the given queries, but not under all possible input values. Hence, these key values are not guaranteed to be the values of the secret key.

However, the value found for a key bit can be proved if it is indeed equal to the actual value of the related bit in the secret key using the concept of *proof by contradiction*. To do so, for each key bit, the complement of its found value is added into $\mathbb{C}$ and the SAT solver is run. If there exists no solution to $\mathbb{C}$, i.e., the SAT problem is unsatisfiable, the value of the related key bit is proven to be the one in the found solution.

As a simple example, consider the majority circuit in Fig. 3(a) and suppose that it is locked using XOR/XNOR

TABLE I
DETAILS OF THE ISCAS'85 CIRCUITS.

Circuit	Original Netlist			p	Locked Netlist			
					Anti-SAT	CASLock	SFLL	SKG-Lock
	#in	#out	#gates		#gates	#gates	#gates	#gates
c2670	157	64	1193	64	1321	1320	1421	1401
c3540	50	22	1669	32	1733	1732	1783	1773
c5315	178	123	2307	64	2435	2434	2523	2514
c6288	32	32	2416	32	2480	2479	2531	2516
c7552	206	105	3512	64	3640	3639	3729	3713

gates as given in Fig. 3(b). Assume that a query is found as $abc = 000$ and thus, the value of its output f is obtained as 0 using the oracle. After propagating these values on the locked circuit as shown in Fig. 3(c), a Boolean equation $k_0 \vee \overline{k_1} = 0$, i.e., $\overline{k_0} \wedge k_1$ in CNF, is obtained. In the SAT solution, the key bit values are found as $k_0 k_1 = 01$. Note that these are the proven key values since a SAT solver guarantees that there exists no solution to the SAT problem $\mathbb{C}$, which is extended by either $k_0 = 1$, i.e., k_0 in CNF, or $k_1 = 0$, i.e., $\overline{k_1}$ in CNF, due to a conflict with the found Boolean equation, i.e., $\overline{k_0} \wedge k_1$ in CNF.

The query attack is run on all the resynthesized circuits and the proven values of key bits in each netlist are combined into a single solution. It is developed in Perl and is equipped with the incremental SAT solver CaDiCaL [30]. It is also available at *https://github.com/Centre-for-Hardware-Security/*.

Finally, the solution of the OG resynthesis-based attack is determined after merging the solution of the SCOPE attack over all resynthesized circuits into that of the query attack on all resynthesized circuits without changing the proven values of key bits.

IV. EXPERIMENTAL RESULTS

This section initially presents the results of the proposed OL resynthesis-based attack on the ISCAS'85 circuits [31] and then, those of the OG resynthesis-based attack on the CSAW'19 circuits [24] including compound logic locking.

A. Results on the ISCAS'85 Circuits

As the first experiment set, five ISCAS'85 circuits were considered. Table I presents their details. For our experiments, these circuits were locked by the Anti-SAT [11], CASlock [12], SFLL [6], and SKG-Lock [14] techniques. Note that while Anti-SAT and SFLL were taken from the NEOS tool [32], we implemented CASLock and SKG-Lock. Table I also presents details of the locked circuits. Note that the number of keys, i.e., p, was determined based on the number of inputs and overhead of the locking technique, and circuit characteristics, i.e., the number of inputs, outputs, and gates, were taken from the gate-level netlist.

Observe from Table I that all logic locking techniques lead to circuits with a number of gates close to each other, whereas the one locked by SFLL has a slightly large number of gates. Besides, the overhead on the number of gates in circuits locked by SFLL varies from 4.7% to 19.1% when compared to original circuits.

In the following subsections, we present the results of the resynthesis process and OL resynthesis-based attack, analyze the impact of synthesis parameters on the performance of the resynthesis process and SCOPE attack, and introduce improvements to the run-time of the resynthesis process.

1) Resynthesis of the Locked ISCAS'85 Circuits: The resynthesis is performed by Cadence Genus with a commercial 65 nm standard cell library. Table II presents the resynthesis results of locked circuits. In this table, *unique* denotes the number of unique locked netlists out of 1080 generated netlists and *area*, *delay*, and *power* stand respectively for the average values of the total area in μm^2, delay in the critical path in ps, and total power dissipation in μW on the unique locked netlists. Finally, *time* is the total run-time of the resynthesis process. The resynthesized netlists were generated on a computing server with Intel Xeon processing units at 3.9 GHz and a total of 1 TB memory.

Observe from Table II that the number of unique netlists is less than half of the total number of generated netlists, i.e., 540, except the *c3540* circuit locked by SKG-Lock. Note that Anti-SAT, CASLock, and SFLL lead to fewer unique netlists when compared to SKG-Lock, which is mainly because the logic added by these techniques is more compact than that added by SKG-Lock, which uses a chain of AND gates. We note that the synthesis tool consumes a large amount of time to fulfill a delay constraint that is impossible to meet, such as strict delay constraints and key constraints described in Section III-A. Hence, the run-time of the resynthesis process depends on the locked circuit and the logic locking technique, and more importantly, if there exists enough room for the synthesis tool to satisfy the constraints.

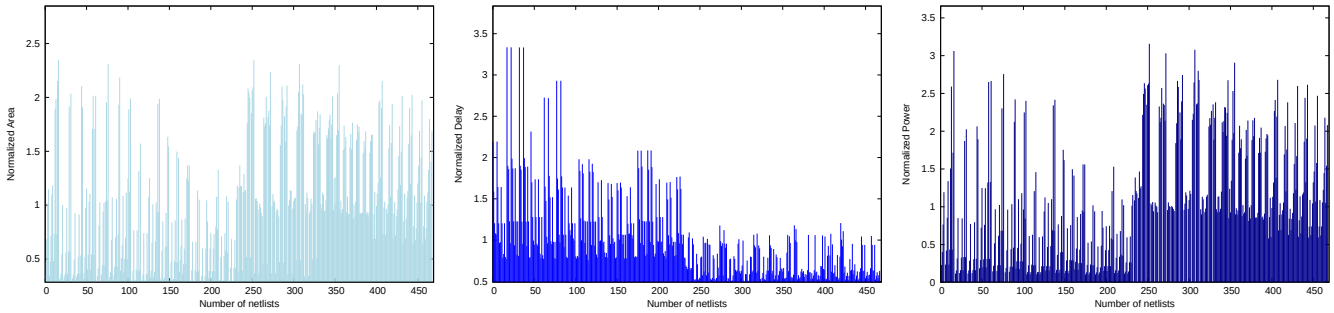
In order to illustrate the diversity of resynthesized netlists, the *c2670* circuit locked by SFLL is considered. Fig. 4 presents the area, delay, and power dissipation of each unique netlist, normalized by their average values given in Table II. Observe that resynthesis generates circuits significantly different from each other in terms of hardware complexity. The standard deviation on area, delay, and power dissipation values of all these netlists are computed as 1578, 235, and 4964, respectively. Note also that in this figure, the netlists after instance number 232 have a distinct profile, since they are generated using key constraints described in Section III-A.

In order to illustrate the differences in the structure of generated netlists, the *c2670* circuit locked by SKG-Lock is considered. Fig. 5 presents the graphs of two netlists resynthesized using the same synthesis parameters, except for the delay constraint. In this figure, red, green, and blue circles denote the inputs, key bits, and outputs, respectively; the gray triangles represent the gates. Observe that a small change in the delay constraint can lead to a structurally different netlist, where the difference between the number of gates and logic levels is 599 and 12, respectively.

2) Attacks on the Locked ISCAS'85 Circuits: Table III presents the results of the SCOPE attack on the original locked netlists and those of OL resynthesis-based attack on the unique locked netlists generated in the resynthesis process. In this

TABLE II
RESULTS OF RESYNTHESIZED LOCKED ISCAS'85 CIRCUITS.

Technique	Details	c2670	c3540	c5315	c6288	c7552
Anti-SAT	unique	480	537	464	498	439
	area	2357	2803	4112	7265	5387
	delay	504	818	663	2144	694
	power	5518	4934	4297	9403	7479
	time	17h14m51s	1d05h56m12s	1d09h56m22s	3d20h50m46s	1d16h01m13s
CASLock	unique	473	449	488	410	479
	area	2359	3112	4173	7739	5337
	delay	513	874	650	2146	676
	power	5170	3304	3852	10693	6765
	time	15h29m56s	1d11h02m52s	1d06h52m54s	4d03h12m29s	1d16h06m52s
SFLL	unique	468	484	477	523	504
	area	2817	3444	4326	7646	5340
	delay	481	870	697	2144	604
	power	6189	6337	9053	12115	11320
	time	13h13m23s	1d47m51s	21h57m07s	2d22h15m07s	22h40m29s
SKG-Lock	unique	521	541	507	527	521
	area	2673	2773	4646	6293	4774
	delay	936	986	782	2093	874
	power	3881	3831	8160	7201	7822
	time	22h22m01s	1d08h8m27s	1d03h56m15s	2d14h29m32s	1d04h19m



(a) (b) (c)

Fig. 4. Normalized complexity of resynthesized netlists of the c2670 circuit locked by SFLL: (a) area; (b) delay; (c) power.

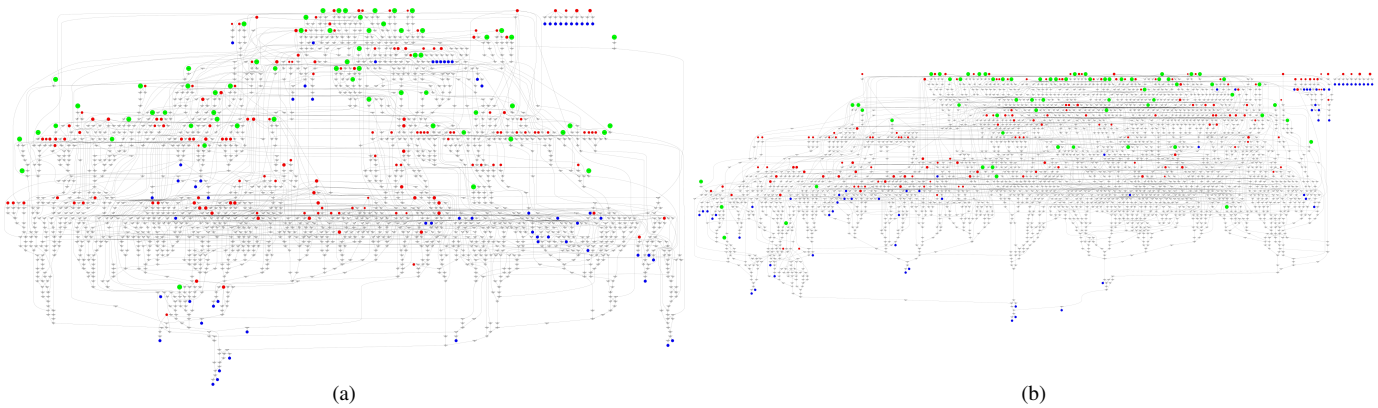
table, *cdk* and *dk* stand respectively for the number of correctly deciphered key bits and the total number of deciphered key bits and *time* is the total time required for the attack. The attacks were also run on the same server used to resynthesize the locked netlists.

Observe from Table III that the SCOPE attack is not entirely successful on any of the original locked netlists. However, the use of resynthesized netlists enables us to decipher the values of a large number of key bits, and even the whole key, e.g., for the *c2670* and *c3540* circuits locked by SKG-Lock. Note that the SCOPE attack can decipher almost all of the key bits using the resynthesized netlists locked by SKG-Lock. While the results on the netlists locked by SKG-Lock are all correct, the ones on the netlists locked by Anti-SAT, CASLock, and SFLL are slightly better than a random guess. The run-time of the SCOPE attack and our resynthesis-based attack depends mainly on the number of gates and keys in the locked design.

To find the SAT resiliency of resynthesized locked circuits, the SAT-based attack of [10] was run on 541 netlists of the *c3540* circuit locked by SKG-Lock with a time limit of 2 days. This circuit was chosen since it has the smallest number of key bits. Note that the SAT-based attack was not able to find the secret key of any resynthesized locked netlists. This experiment indicates that the resynthesis changes only the structure of the circuit as shown in Fig. 5, but maintains its SAT resiliency.

3) Redundant Synthesis Runs: Observe from Tables II and III that the total run-time of the proposed attack is dominated by the resynthesis process. However, it is possible to reduce the time required to resynthesize the locked netlist by removing redundant synthesis runs without sacrificing any unique netlists. For example, it is observed that the *high* value of the *syn_gen* parameter given in Section III-A can be removed from the parameter list, since all possible synthesis scripts including this parameter generate the same circuit when this parameter is *low* or *medium*. Thus, the number of generated circuits, i.e., 1080, reduces to 720.

4) Convergence on the Number of Deciphered Keys: It is also observed that the number of key bits deciphered by the SCOPE attack on all unique resynthesized netlists can actually be obtained using a small number of netlists. Fig. 6 presents the number of deciphered key bits along the unique resynthesized netlists of the *c2670* circuit locked by SKG-Lock. Observe from this figure that although a large number of unique netlists increases the quality of the SCOPE attack, actually a small number of unique netlists, 147 in this



(a) (b)

Fig. 5. Graphs of resynthesized netlists generated using a difference in the delay constraint dc: (a) dc is 990 ps; (b) dc is 496 ps.

TABLE III
RESULTS OF OL ATTACKS ON THE LOCKED ISCAS'85 CIRCUITS.

Circuit	Anti-SAT				CASLock				SFLL				SKG-Lock			
	SCOPE		Resynthesis		SCOPE		Resynthesis		SCOPE		Resynthesis		SCOPE		Resynthesis	
	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time	cdk/dk	time
c2670	0/0	4s	37/64	34m18s	0/0	4s	35/64	33m47s	0/0	4s	34/64	37m32s	32/32	4s	64/64	44m37s
c3540	0/0	3s	17/32	21m27s	0/0	3s	17/32	18m12s	0/0	2s	19/32	21m29s	17/17	2s	32/32	24m30s
c5315	0/0	5s	38/64	42m34s	0/0	5s	30/64	43m54s	0/0	5s	33/64	46m23s	32/32	5s	62/62	52m06s
c6288	0/0	3s	18/32	29m08s	0/0	3s	16/32	27m18s	0/0	3s	16/31	33m19s	16/16	3s	31/31	34m24s
c7552	0/0	6s	38/64	45m31s	0/0	6s	47/64	49m13s	0/0	6s	38/63	52m26s	32/32	6s	61/61	56m45s

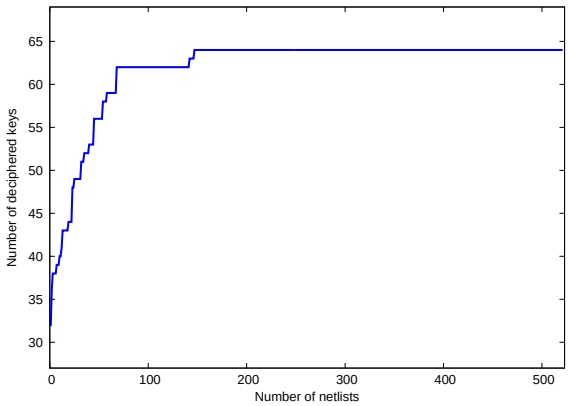


Fig. 6. Convergence on the number of deciphered keys over the number of resynthesized netlists in the SCOPE attack.

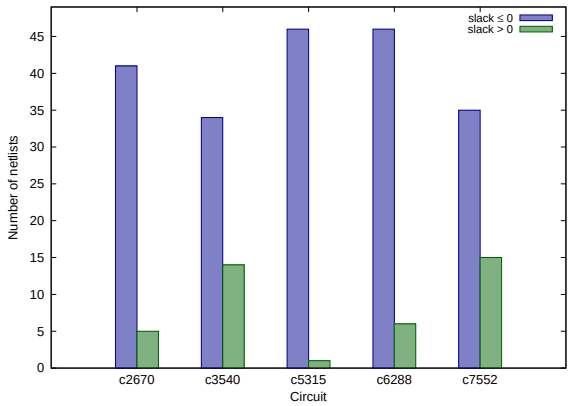


Fig. 7. Classification of resynthesized netlists based on their slack values on promising solutions of SCOPE attack.

case, is sufficient to achieve the same result as when all 521 unique netlists are considered. We note that a similar situation was also observed on circuits locked by other techniques.

5) Promising Resynthesized Netlists: Moreover, it is observed that the SCOPE attack is more successful on specific resynthesized netlists. To find a set of synthesis parameters that enables the SCOPE attack to decipher more key values, we initially define two categories of netlists based on the slack time of the design, i.e., the difference between the required and arrived time in the critical path, as follows: i) netlists with a slack value less than or equal to 0; ii) netlists with a slack value greater than 0. The slack value of a design gives indeed a rough idea of the effort put in by the synthesis tool; for the netlists in the first category, the synthesis tool works extremely hard to meet the delay constraint, trying many logic transformations and optimization techniques.

Then, the solutions of the SCOPE attack on all possible 1080 netlists are obtained and sorted based on the number of deciphered key bits in descending order. The top 10% of these sorted netlists are categorized based on their slack values.

Fig. 7 presents the results of this experiment on the circuits locked by SKG-Lock. Observe that the netlists that enable the SCOPE attack to decipher more key values generally have a slack value less than or equal to 0. Thus, to generate such circuits, one can easily add strict delay constraints or key constraints as described in Section III-A. We note that a similar result was also observed on resynthesized netlists locked by other techniques.

6) Structural Analysis: In order to improve the performance of the resynthesis process, the logic cone, which is the locking technique is applied on, can be extracted and resynthesized. Note that the output of this logic cone is a single primary

TABLE IV
RESULTS OF THE RESYNTHESIS PROCESS AND OL RESYNTHESIS-BASED ATTACK ON ENTIRE CIRCUIT AND LOGIC CONE.

Circuit	Entire Circuit			Logic Cone		
	unique	time	cdk/dk	unique	time	cdk/dk
c2670	468	13h13m23s	34/64	319	7h46m26s	34/64
c3540	484	1d47m51s	19/32	320	6h29m35s	16/32
c5315	477	21h57m14s	33/64	313	7h6m16s	32/64
c6288	523	2d22h15m7s	16/31	302	6h20m57s	19/32
c7552	504	22h40m29s	38/63	279	6h57m14s	38/63

TABLE V
DETAILS OF THE LOCKED CSAW'19 CIRCUITS.

Circuit	Details			Number of key bits		
	#in	#out	#gates	RLL	SFLL-rem	Total
small	522	512	15995	40	40	80
medium	767	757	24008	60	60	120
large	1452	1445	36584	80	80	160
bonus	892	1746	23004	128	128	256

output, while its inputs are primary inputs, but not necessarily all the primary inputs of the locked design. Thus, the run-time of the resynthesis process can be decreased, since the logic cone has a small number of inputs, outputs, and gates when compared to the whole locked circuit.

Table IV presents details on the resynthesis process on entire locked circuits and logic cones when the circuits locked by SFLL are used. Observe that the resynthesis process on a logic cone generates less number of unique designs and takes significantly less time without a significant loss on the solution quality when compared to the resynthesis process on the entire circuit. We note that similar results were also observed on circuits locked by other techniques.

B. Results on the CSAW'19 Circuits

As the second experiment set, we used the state-of-the-art locked circuits from the CSAW'19 contest [24]. Details of these circuits are given in Table V. Note that two logic locking techniques – RLL [9] and SFLL-rem [13] – are applied together to lock a circuit.

In the following two subsections, we present the results of the resynthesis process and the resynthesis-based attack.

1) Resynthesis of the Locked CSAW'19 Circuits: Table VI presents the resynthesis results of locked circuits. Observe that the number of unique resynthesized netlists is larger than half of the total number of generated netlists, i.e., 540. As the hardware complexity of designs increases, the run-time of the resynthesis process increases. We note that diverse netlists in terms of complexity are obtained, e.g., the standard deviation on area, delay, and power dissipation values of all the locked netlists of the *small* circuit are computed as 8526, 1029, and 20074, respectively.

2) Attacks on the Locked CSAW'19 Circuits: Table VII presents results of the attacks obtained, after they are applied to the original locked netlist, denoted as *OLN*, and all unique resynthesized netlists, denoted as *URNs*. In this table, *prv* stands for the number of proven values of key bits. Note that since the secret key is **not publicly available**, the *cdk* values are omitted for the SCOPE and resynthesis-based attacks.

TABLE VI
RESULTS OF RESYNTHESIZED LOCKED CSAW'19 CIRCUITS.

Circuit	unique	area	delay	power	time
small	557	18935	1631	23571	5d3h22m28s
medium	569	26080	1745	31284	6d12h24m16s
large	567	31348	1798	24610	5d21h42m10s
bonus	560	20643	1758	19090	4d14h44m29s

TABLE VII
RESULTS OF ATTACKS ON THE LOCKED CSAW'19 CIRCUITS.

Circuit-Netlist	SCOPE		Query		Resynthesis	
	dk	time	prv	time	dk	time
small - OLN	19	20s	39	1m21s	40	1m41s
small - URNs	77	4h10m42s	40	1d10h4m37s	79	1d14h15m19s
medium - OLN	32	41s	58	6m37s	59	7m18s
medium - URNs	117	8h33m56s	58	3d19h12m13s	120	4d3h46m9s
large - OLN	30	1m7s	79	6m19s	79	7m26s
large - URNs	152	12h56m15s	80	3d2h52m11s	159	3d15h48m26s
bonus - OLN	64	1m46s	118	3m2s	120	4m48s
bonus - URNs	233	16h7m17s	125	1d20h29m22s	252	2d12h36m39s

Observe from Table VII that the original SCOPE attack could only decipher a small number of key bits, all of which belong to RLL, and the query attack can prove the values of a large number of key bits, all of which again belong to RLL, on the original locked circuits. Thus, the resynthesis-based attack could only decipher the RLL key bits on the original locked circuits. However, the use of resynthesized circuits makes the SCOPE attack decipher more key bits that also belong to SFLL-rem and makes the query attack prove the values of more key bits that belong to RLL. Thus, the resynthesis-based attack could decipher almost all the values of the secret key, proving almost all the values of the key bits of RLL. Note that all the unknown key bits belong to SFLL-rem. Observe that the run-time of attacks increases, as the number of gates and key bits increases.

After the values of key bits of the CSAW'19 circuits were determined, they were sent to the contest organizers for evaluation. Table VIII presents the results of the resynthesis-based attack along with those of other techniques which participated in the contest.

Observe from Table VIII that our proposed attack can determine all the key bits of RLL correctly, even though there are unproven key bits in the *medium* and *bonus* circuits as shown in Table VII. This observation implies that the guesses of the SCOPE attack on those key bits are actually correct. Moreover, the proposed technique can decipher the key bits of SFLL-rem with a number of deciphered key bits greater than any other OL technique with high accuracy.

V. CONCLUSIONS

This work has shown that EDA tools can be used to generate variants of locked circuits that may be vulnerable to existing logic locking attacks and such circuits can be generated using a specific set of synthesis parameters. It was shown that the run-time of the proposed technique can be improved using a small number of resynthesized netlists without diminishing its solution quality. Experimental results clearly indicated that the use of many resynthesized circuits enables existing attacks to decipher values of a large number of key bits with high

TABLE VIII
RESULTS OF ATTACKS ON THE LOCKED CSAW'19 CIRCUITS.

Approach	Attack Scenario	Circuit							
		small (40+40)		medium (60+60)		large (80+80)		bonus (128+128)	
		RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem	RLL	SFLL-rem
Key sensitization [33]	OG	40/40	—	60/60	—	80/80	—	—	—
Hamming distance-based attack [24]	OG	30/30	—	50/50	—	72/72	—	—	—
Automated analysis + SAT [24]	OG	11/18	—	31/50	—	10/34	—	—	—
Sub-circuit SAT [24]	OG	17/17	—	29/29	—	—	—	—	—
Redundancy-based [27]	OL	28/28	4/12	35/35	23/28	45/45	0/51	66/66	8/27
Bit-flipping attack [34]	OG	40/40	—	60/60	—	80/80	—	—	—
Topology guided attack [28]	OL	15/32	—	19/50	—	36/73	—	75/108	—
Resynthesis-based attack	OG	40/40	20/39	60/60	29/60	80/80	35/79	128/128	55/124

accuracy. Hence, the resynthesis of a locked circuit can be utilized as a pre-attack step for many existing attacks in order to improve their success rate. As future work, we plan to consider other synthesis parameters, such as fanout, capacitance limits, and wire loads, which enable synthesis tools to generate different circuits. Also, we aim to incorporate other commercial and open source EDA tools into the resynthesis process to generate different unique netlists.

ACKNOWLEDGMENT

The authors thank Nimisha Limaye for evaluating the keys found by the proposed technique on the CSAW'19 benchmarks.

REFERENCES

[1] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[2] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *DAC*, 1998, pp. 776–781.

[3] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote Activation of ICs for Piracy Prevention and Digital Right Management," in *ICCAD*, 2007, pp. 674–677.

[4] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, 2012.

[5] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *J. Electron. Test.*, vol. 35, no. 3, pp. 273–291, 2019.

[6] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *ACM CCS*, 2017, pp. 1601–1618.

[7] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Threats on Logic Locking: A Decade Later," in *GLVLSI*, 2019, pp. 471–476.

[8] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE TVLSI*, vol. 29, no. 8, pp. 1529–1542, 2021.

[9] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *DATE*, 2008, pp. 1069–1074.

[10] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *HOST*, 2015, pp. 137–143.

[11] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE TCAD*, vol. 38, no. 2, pp. 199–207, 2019.

[12] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 175–202, 2019.

[13] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-Based Perspective," *IEEE TCAD*, vol. 39, no. 12, pp. 4439–4452, 2020.

[14] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *ISVLSI*, 2021, pp. 114–119.

[15] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *HOST*, 2016, pp. 236–241.

[16] 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 *Cryptographic Hardware and Embedded Systems*, 2017, pp. 189–210.

[17] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[18] A. Sengupta, N. Limaye, and O. Sinanoglu, "Breaking CAS-Lock and Its Variants by Exploiting Structural Traces," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, p. 418–440, 2021.

[19] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," in *DATE*, 2019, pp. 936–939.

[20] F. Yang, M. Tang, and O. Sinanoglu, "Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) – Unlocked," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2778–2786, 2019.

[21] Z. Han, M. Yasin, and J. Rajendran, "Does Logic Locking Work with EDA Tools?" in *USENIX Security Symposium*, 2021, pp. 1055–1072.

[22] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability Assessment Tool and Attack for Provably-Secure Logic Locking Techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[23] Y. Liu, M. Zuzak, Y. Xie, A. Chakraborty, and A. Srivastava, "Strong Anti-SAT: Secure and Effective Logic Locking," in *ISQED*, 2020, pp. 199–205.

[24] B. Tan *et al.*, "Benchmarking at the Frontier of Hardware Security: Lessons from Logic Locking," 2020. [Online]. Available: https://arxiv.org/abs/2006.06806

[25] M. John, A. Hoda, R. Chouksey, and C. Karfa, "SAT Based Partial Attack on Compound Logic Locking," in *Asian Hardware Oriented Security and Trust Symposium*, 2020, pp. 1–6.

[26] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based Attack on Compound Logic Locking Techniques," in *DATE*, 2021, pp. 1166–1171.

[27] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *DATE*, 2019, pp. 540–545.

[28] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "TGA: An Oracle-Less and Topology-Guided Attack on Logic Locking," in *ASHES*, 2019, p. 75–83.

[29] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Tech. Rep. 12-93, 1993.

[30] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[31] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," in *ISCAS*, 1985, pp. 663–698.

[32] K. Shamsi, "Netlist Encryption and Obfuscation Suite," 2021. [Online]. Available: https://bitbucket.org/kavehshm/neos/src/master/

[33] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *DAC*, 2012, pp. 83–89.

[34] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based Bit-Flipping Attack on Logic Encryptions," in *DATE*, 2018, pp. 629–632.