

KRATT: QBF-Assisted Removal and Structural Analysis Attack Against Logic Locking

Levent Aksoy[†], Muhammad Yasin[‡] and Samuel Pagliarini^{†§}

[†]Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia

[‡]Department of Computer and Software Engineering, National University of Sciences and Technology, Islamabad, Pakistan

[§]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA

Email: levent.aksoy@taltech.ee, m.yasin@ceme.nust.edu.pk, pagliarini@cmu.edu

Abstract—This paper introduces KRATT, a removal and structural analysis attack against state-of-the-art logic locking techniques, such as single and double flip locking techniques (SFLT and DFLT). KRATT utilizes powerful quantified Boolean formulas (QBFs), which have not found widespread use in hardware security, to find the secret key of SFLT and DFLT for the first time. It can handle locked circuits under both oracle-less (OL) and oracle-guided (OG) threat models. It modifies the locked circuit and uses a prominent OL attack to make a strong guess under the OL threat model. It uses a structural analysis technique to identify promising protected input patterns and explores them using the oracle under the OG model. Experimental results on ISCAS'85, ITC'99, and HeLLO: CTF'22 benchmarks show that KRATT can break SFLT using a QBF formulation in less than a minute, can decipher a large number of key inputs of SFLT and DFLT with high accuracy under the OL threat model, and can easily find the secret key of DFLT under the OG threat model. It is shown that KRATT outperforms publicly available OL and OG attacks in terms of solution quality and run-time.

Index Terms—logic locking, removal attack, structural analysis, quantified Boolean formula, satisfiability

I. INTRODUCTION

In the globalized semiconductor industry, fabless design houses outsource the fabrication of their integrated circuits (ICs) to foundries, giving rise to security threats in the case of an untrusted foundry, such as piracy, overproduction, and reverse engineering, which harm the semiconductor industry financially and may even undermine national security [1]. Many techniques, such as watermarking, digital rights management, metering, and logic locking [2], have been introduced for protection against these security threats. Among these techniques, logic locking, which inserts additional logic with key inputs into the original design, has been a promising solution to many security threats. It ensures that the locked design behaves the same as the original one only when the secret key is provided. Otherwise, it generates a wrong output.

In logic locking, there are generally two main attack scenarios: (i) in the oracle-less (OL) threat model, the adversary has only the locked netlist obtained either by reverse-engineering the layout at the untrusted foundry delivered by the design house or by reverse-engineering the functional IC obtained from the market; (ii) in the oracle-guided (OG) threat model, the adversary also has the functional IC, which can be used as an oracle to apply inputs and observe outputs. An important milestone in logic locking is the OG satisfiability (SAT)-based

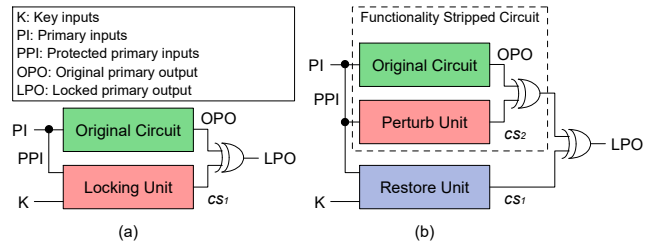


Fig. 1. State-of-the-art logic locking techniques: (a) SFLT; (b) DFLT.

attack [3], which broke all the logic locking techniques existing at that time. The SAT-based attack iteratively finds distinguishing input patterns (DIPs), which eliminate wrong keys. Thus, the state-of-the-art logic locking techniques aimed to increase the run-time for iterations and/or the number of iterations in the SAT-based attack [4]–[12]. As shown in Fig. 1, they are generally grouped into two categories: single flip or double flip locking techniques (SFLT and DFLT). SFLT [4]–[7] use a single critical signal cs_1 , which corrupts the original circuit for wrong keys. DFLT [8]–[12] use a critical signal cs_2 to corrupt the original design for a specific input and use a critical signal cs_1 to correct this corruption.

Over the years, many efficient attacks have been proposed against these state-of-the-art techniques under the OL and OG threat models [13]–[25]. However, they consider either OG or OL threat model [13]–[18], target particular locking techniques [19]–[22], and depend on commercial tools [23], [24]. Moreover, the removal attack of [25] can obtain the original circuit by removing the locking unit of SFLT from the locked circuit. However, in different scenarios, e.g., when there is no possibility to sell/fabricate the original circuit or the objective of the adversary is overproduction, finding the secret key is more valuable than obtaining the original circuit.

In this paper, we introduce a removal and structural analysis attack under both OL and OG threat models, called KRATT, developed not only for a specific logic locking technique but for a large number of SAT-resilient SFLT and DFLT. Importantly, KRATT does not rely on commercial tools. The main contributions of this paper are three-fold: (i) it represents the problem of finding the secret key of SFLT as a quantified Boolean formula (QBF) problem; (ii) it shows an efficient way of determining the values of the secret key of SFLT and DFLT with high accuracy under the OL threat model using a circuit modification technique and the prominent OL

x_3	x_2	x_1	K^7	K^6	K^5	K^4	K^3	K^2	K^1	K^0	
0	0	0	0	0	0	0	0	0	0	1	
0	0	1	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0

Fig. 2. Behavior of a Boolean function locked by a point function.

attack SCOPE [18]; (iii) it introduces a novel approach of breaking DFLTs using structural analysis under the OG threat model. Although the use of QBF in logic locking has been hypothesized [19], to the best of our knowledge, it has never been used in breaking logic locking techniques. The manipulation of the locked circuit for the sake of an OL attack has also not been considered before. Although finding the traces of protected primary inputs has been considered [19], [21], finding them using a SAT formulation has not been proposed. Experimental results on a comprehensive set of locked circuits show that KRATT can easily break SFLT's [4]–[7] under the OL threat model, where the QBF formulation can lead to the secret key of SFLT's [4]–[6]. It can decipher a large number of key inputs of DFLTs [9], [10] with high accuracy under the OL threat model. It can also break DFLTs [9], [11] using a little effort under the OG threat model. Although KRATT can handle a large number of locking techniques, there are still challenging techniques [9], [12], for which it is hard to find the secret key. Although they are out of the scope of KRATT, we describe how techniques of KRATT can be used to construct the original circuit for those techniques.

The rest of this paper is organized as follows: The background concepts are given in Section II. KRATT is described in Section III and experimental results are given in Section IV. Section V discusses KRATT on challenging logic locking techniques and finally, Section VI concludes the paper.

II. BACKGROUND

A. Preliminaries

A Boolean logic function, $\varphi : \mathcal{B}^n \rightarrow \mathcal{B}$, where $\mathcal{B} = \{0, 1\}$, over n variables x_1, \dots, x_n maps each truth assignment to 0 or 1. The logic function φ in *sum of products* (SOP) form, aka *disjunctive normal form* (DNF), is a disjunction of r products p_1, \dots, p_r , where a *product* $p_i = l_1 \cdot l_2 \cdot \dots \cdot l_j$, $i \leq r$ and $j \leq n$, is a conjunction of literals. A *literal* l_k , $k \leq n$, is either a variable x_k or its complement $\overline{x_k}$. A *minterm* is a product including a literal for each variable, i.e., $j = n$. An *implicant* in SOP form is also a product if and only if it evaluates f to 1. Similarly, φ in *product of sums* (POS) form, aka *conjunctive normal form* (CNF), on n variables is a conjunction of t sums s_1, \dots, s_t , where a *sum*, $s_i = l_1 + l_2 + \dots + l_j$, $i \leq t$ and $j \leq n$, is a disjunction of literals. A *maxterm* is a sum including a literal for each variable, i.e., $j = n$. An *implicant* in POS form is a sum if and only if it evaluates f to 0.

The *SAT problem* is to find an assignment to the variables of a function φ in CNF that makes φ to be equal to 1 or to prove

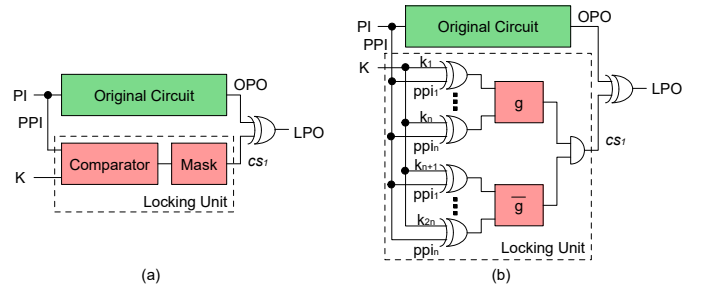


Fig. 3. (a) SARLock [4]; (b) AntiSAT [5].

that φ is equal to 0. The *QBF problem* is the generalization of the SAT problem, in which both existential (\exists) and universal (\forall) quantifiers can be applied to each variable.

B. Related Work

Before the SAT-based attack, earlier work focused on different types of key gates, such as look-up tables (LUTs), while considering the hardware complexity trade-offs [26]. After the SAT-based attack, among many other techniques, the point function has been used to generate SAT-resilient locked circuits [26]. Note that *one-point function* evaluates to 1 at exactly one input pattern. For example, consider a Boolean function with 3 variables and suppose that it is locked by 3 key inputs using a one-point function. Fig. 2 presents its behavior under all possible keys, where K^i stands for the assignment of the value i in binary to key inputs, i.e., $k_3k_2k_1 = (i)_{bin}$ with $0 \leq i \leq 2^3 - 1$, and the logic 0 (1) value under each key denotes that the locked function is (not) equal to the original one. Note that $k_3k_2k_1 = 100$ is the secret key for our example. Since the output under each wrong key differs for one input pattern, a DIP found by the SAT-based attack eliminates only one wrong key, forcing an exponential increase in the number of DIPs required to find the secret key.

Under the category of SFLT's, SARLock [4] adds a comparator and a masking circuit connected with the original netlist in a way that it generates corruption on a specific protected primary input as shown in Fig. 3(a). Anti-SAT [5] utilizes complementary functions, which are generally composed of an AND gate tree, whose output is merged with the original circuit as shown in Fig. 3(b). CAS-Lock [6] is based on the same concept of Anti-SAT, but uses a mix of AND and OR gates in the tree. Gen-Anti-SAT [7] uses non-complementary functions to increase output corruption.

Under the category of DFLTs, tenacious and traceless logic locking (TTLock) technique [8] initially corrupts an output based on a protected primary input in the perturb unit and then, corrects this output only when the secret key is applied in the restore unit as shown in Fig. 1(b). It is improved for output corruption and resiliency in [9], [10]. The corrupt and correct (CAC) technique [11] flips the original primary output for the protected primary input and flips it back when the primary input is equal to the protected primary input or the secret key. Techniques that hide the functionality of the restore unit in a read-proof hardware [27] are also given in [9], [12].

The OL attacks explore patterns in the structure of a locked netlist using statistical analysis [16]–[18]. For example, the

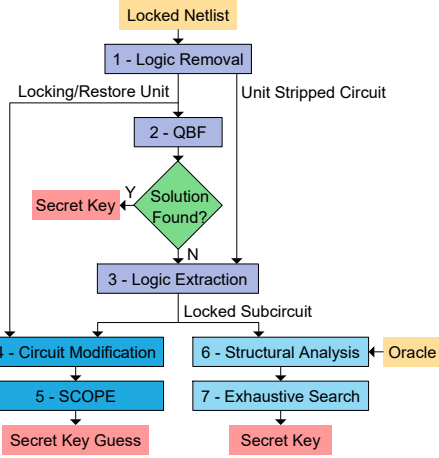


Fig. 4. Flow of the removal and structural analysis attack.

SCOPE attack [18] is an unsupervised constant propagation technique, which analyzes each key bit of the locked design for critical features, such as area, power dissipation, and delay, which can reveal its correct value, after it is assigned to logic 0 and 1 value. Similar to the OG SAT-based attack, the technique of [13], called DDIP, eliminates at least 2 DIPs in a single iteration. The approximate attack of [14], called AppSAT, aims for approximate functional recovery. For SFLT, removal attacks that find the single critical point cs_1 and obtain the original circuit by removing the locking unit are proposed in [25]. Note that a DFLT is resilient to removal attacks since the original circuit is combined with the perturb unit, even though its restore unit can be easily removed. For DFLTs, efficient structural attacks are presented in [19]–[24].

III. REMOVAL AND STRUCTURAL ANALYSIS ATTACK

This section presents our removal and structural analysis attack KRATT. Its flow chart is given in Fig. 4 and its main steps are described in detail in the following subsections.

A. Logic Removal and QBF with Logic Extraction

For both SFLT and DFLT under both OL and OG threat models, KRATT takes the locked netlist as input and initially extracts its locking/restore unit with the protected primary inputs and associated key inputs from the locked netlist. To do so, it follows two steps: (i) it determines the critical signal, i.e., cs_1 in Fig. 1, in the locked netlist by finding the output of the first gate in the paths from key inputs to primary outputs, which all the key inputs pass through; (ii) it removes the logic cone of this critical signal and obtains the remaining of the locked netlist, called *unit stripped circuit* (USC). Note that the logic shared between the locking/restore unit and USC is preserved in both circuits and the critical signal becomes another primary input of USC. In the locking/restore unit, for each protected primary input, KRATT determines its associated key input. To do so, for each protected primary input ppi_j , $1 \leq j \leq n$, where n is the number of protected primary inputs, it finds a logic gate, whose inputs are ppi_j , its associated key input, or their complements. Note that a

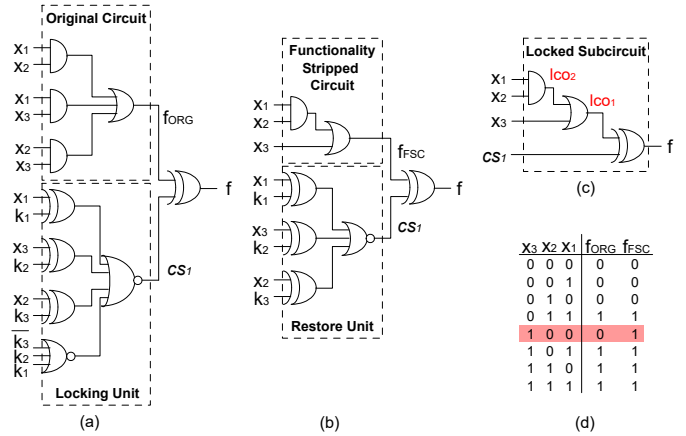


Fig. 5. Locked majority circuits: (a) SARLock; (b) TTLock; (c) locked subcircuit in TTLock; (d) truth table for original and functionality stripped circuits in TTLock.

protected primary input is associated with two key inputs in Anti-SAT and its variants as shown in Fig. 3(b).

Then, KRATT checks if there exist values of key inputs that set the output of the locking/restore unit, i.e., cs_1 in Fig. 1, to a constant logic value, i.e., 0 or 1, for all possible protected primary inputs. It formalizes this problem as a QBF problem in two steps: (i) it generates the CNF formula of this unit, $\varphi(PPI, K)$, as the conjunction of CNF formulas of each gate; (ii) it generates two QBF problems, $\exists K \forall PPI, \varphi(PPI, K)_{cs_1=0}$ and $\exists K \forall PPI, \varphi(PPI, K)_{cs_1=1}$. Then, it solves these problems using a QBF solver. If there exists a solution to one of these QBF problems, the values of key inputs are determined to be the secret key. Note that two QBF problems are generated in order to check all possible values for the critical signal cs_1 .

If there exist no solutions to both QBF problems, for classification purposes, KRATT checks if the locking/restore unit realizes a comparator logic or its complement between the protected primary inputs and their associated key inputs using a SAT formulation to ensure that this unit is actually the restore unit of DFLTs. Then, it applies the logic extraction method, which takes the USC as an input and generates the *locked subcircuit* including only the locked primary outputs. To do so, it finds the primary outputs reached by the critical signal cs_1 in USC and generates their logic cones.

As an example, consider the majority circuit locked by SARLock and TTLock using 3 key inputs as shown in Fig 5. Note that protected primary inputs x_1 , x_2 , and x_3 are associated with key inputs k_1 , k_3 , and k_2 , respectively. The locking unit in Fig. 5(a) always generates logic 0 for all possible x_1 , x_2 , and x_3 values due to the 3-input NOR gate when $k_3k_2k_1 = 100$, which is the secret key found by the QBF formulation. However, since the restore unit in Fig. 5(b) only compares protected primary inputs with associated key inputs, there exist no solutions to the QBF problems. The subcircuit locked by TTLock is shown in Fig. 5(c). For the locked netlists, whose secret key cannot be found by the QBF formulation, the steps of KRATT under the OL and OG threat models are described in Sections III-B and III-C, respectively.

B. OL Attack: Circuit Modification and SCOPE

Under the OL threat model, KRATT modifies the locked netlist to enable SCOPE [18], which may fail to make a guess or make a random guess as a standalone attack as shown in Section IV, to make a strong guess. For SFLT, it focuses on the locking unit that includes key inputs. For Anti-SAT and its variants, where each protected primary input is associated with two key inputs, it removes all the protected primary inputs from the locking unit by setting them to a constant logic value since these inputs are not relevant to the complementary/non-complementary functions. For DFLT, it focuses on the locked subcircuit and replaces the protected primary inputs with their associated key inputs since the information on the values of the protected primary input, although not complete, is inside the locked subcircuit. Then, it runs SCOPE on these circuits. Note that Steps 1-5 in Fig. 4 are the steps of KRATT under the OL threat model.

C. OG Attack: Structural Analysis and Exhaustive Search

The *functionality stripped circuit* (FSC) of DFLT as shown in Fig. 1(b) is obtained after corrupting the original circuit on the protected primary input(s). For our example in Fig. 5(b), it is obtained by changing a maxterm of the original function into a minterm as shown in Fig. 5(d). The FSC includes implicants consisting of protected primary inputs and thus, their values, i.e., the secret key, can be found when exercised with oracle.

Under the OG threat model, KRATT initially finds all the logic cones of the locked subcircuit, where their inputs are the protected primary inputs. The output of such a cone is denoted as lco as shown in Fig. 5(c). Then, for each logic cone, it generates two sets of values of all protected primary inputs, which initially have unspecified values denoted as X . It determines their values by finding the input values of the logic cone when its output, i.e., lco_i , $1 \leq i \leq k$, where k is the number of such logic cones, is set to logic 0 and 1. It formalizes this problem as a SAT problem. The reason behind setting the output of the logic cone to 0 and 1 is to try both a maxterm and minterm in the logic cone and the reason behind finding only two sets rather than all possible implicants of the logic cone is to try a small number of promising ones, as other gates in the logic cone will also be considered. For our example in Fig. 5(c), there exist 4 possible sets, $x_3x_2x_1 = 000$ and $x_3x_2x_1 = 100$ found for lco_1 when it is set to 0 and 1, respectively and $x_3x_2x_1 = X00$ and $x_3x_2x_1 = X11$ found for lco_2 when it is set to 0 and 1, respectively. These sets are augmented by those, if not available, where a single protected primary input is set to a constant value and all others are set to X to cover all possible values, e.g., $x_3x_2x_1 = 0XX$.

Then, all these sets of values for the protected primary inputs are sorted based on the number of unspecified values in ascending order. Starting with the one including the maximum number of specified protected primary input values, for each set, KRATT generates all possible protected primary input values by exercising logic 0 and 1 values on the unspecified entries, applies it to the oracle while other primary inputs are set to logic 0, and obtains the oracle output. Then, it applies

TABLE I
DETAILS OF THE ISCAS'85 AND ITC'99 CIRCUITS.

Circuit	#inputs	#outputs	#gates	#key inputs
c2670	157	64	1193	64
c5315	178	123	2307	64
c6288	32	32	2416	32
b14_C	277	299	9768	128
b15_C	485	519	8367	128
b20_C	522	512	19683	128

these values of primary inputs of the original design to primary inputs of the locked netlist while the values of key inputs are set to those of associated protected primary inputs and obtains the locked netlist output. If outputs of the oracle and locked netlist match as shown in Fig. 2, it determines the secret key based on the association between the protected primary inputs and key inputs. For our example in Fig. 5(b), the secret key is found as $k_3k_2k_1 = 010$ when it is observed that the original design and locked netlist generate the same output at $x_3x_2x_1 = 100$, which was deduced from the logic cone with the output lco_1 . Note that Steps 1-3 and 6-7 in Fig. 4 are the steps of KRATT under the OG threat model.

KRATT is developed in Perl and is freely available [28]. It is only equipped with the QBF solver DepQBF [29] and the SAT solver cryptominisat [30].

IV. EXPERIMENTAL RESULTS

As the first experiment set, we used a total of six circuits from ISCAS'85 and ITC'99 benchmarks in a wide range of the number of gates. Table I presents the details taken from their bench files. We locked them using our implementations of Anti-SAT [5], SARLock [4], CAC [11], and TTLock [8] at register transfer level (RTL) with the number of key inputs given in Table I. We also *synthesized* the locked circuit using the Cadence Genus logic synthesis tool to break the regular structure of the locking scheme, making it harder to find the secret key, especially for removal and structural analysis attacks including KRATT. We used OL attack SCOPE [18] and OG attacks, SAT-based [3], DDIP [13], and AppSAT [14], all of which are available to the public. The FALL attack of [19], which targets only stripped functionality logic locking (SFL) techniques, was also run on circuits locked by TTLock, but without success. Since DDIP and AppSAT may return a wrong key in a single run, they were run multiple times with different settings. These attacks were run on a computing server including 32 Intel Xeon processing units at 3.9 GHz with 128 GB memory. The time limit for the QBF solver was set to 1 minute since finding a satisfiable solution, if exists, is generally trivial.

Tables II and III present the results of OL and OG attacks, respectively. In these tables, the run-time of attacks is given in seconds. In Table II, cdk and dk are the number of correctly deciphered key inputs and deciphered key inputs, respectively. In Table III, OoT indicates that a solution could not be found due to the given time limit set to 2 days.

Observe from Table II that while the SCOPE attack cannot decipher all key inputs, except the circuits locked by SARLock, KRATT can decipher all the key inputs of the

TABLE II
RESULTS OF OL ATTACKS ON LOCKED ISCAS'85 AND ITC'99 CIRCUITS.

Circuit	SFLT								DFLT							
	Anti-SAT				SARLock				CAC				TTLock			
	SCOPE		KRATT		SCOPE		KRATT		SCOPE		KRATT		SCOPE		KRATT	
	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU	cdk/dk	CPU
c2670	13/23	3.12	64/64	0.39	64/64	3.3	64/64	0.34	17/26	3.22	33/64	64.48	14/26	3.19	34/64	64.37
c5315	13/22	3.95	64/64	0.68	64/64	3.94	64/64	0.50	12/19	3.93	33/64	64.61	16/31	4.03	34/64	64.53
c6288	7/12	2.25	32/32	0.67	32/32	2.48	32/32	0.74	11/18	2.29	18/32	64.09	9/14	2.23	20/32	63.07
b14_C	32/55	15.15	128/128	4.61	128/128	15.52	128/128	10.11	39/71	15.08	67/128	74.65	35/59	14.89	70/128	74.42
b15_C	22/38	20.04	128/128	9.01	128/128	20.42	128/128	11.91	18/35	21.41	64/128	79.57	43/70	20.22	68/128	78.70
b20_C	24/46	25.82	128/128	13.60	128/128	26.25	128/128	16.98	30/54	26.11	58/102	79.35	24/46	26.16	68/128	82.34

TABLE III
RESULTS OF OG ATTACKS ON LOCKED ISCAS'85 AND ITC'99 CIRCUITS.

Circuit	SFLT								DFLT							
	Anti-SAT				SARLock				CAC				TTLock			
	SAT	DDIP	AppSAT	KRATT	SAT	DDIP	AppSAT	KRATT	SAT	DDIP	AppSAT	KRATT	SAT	DDIP	AppSAT	KRATT
c2670	OoT	OoT	OoT	0.32	OoT	OoT	OoT	0.33	OoT	OoT	OoT	70.79	OoT	OoT	OoT	70.50
c5315	OoT	OoT	OoT	0.65	OoT	OoT	OoT	0.47	OoT	OoT	OoT	76.37	OoT	OoT	OoT	75.94
c6288	OoT	OoT	OoT	0.63	OoT	OoT	OoT	0.70	OoT	OoT	OoT	163.19	OoT	OoT	OoT	161.21
b14_C	OoT	OoT	OoT	4.58	OoT	OoT	OoT	10.74	OoT	OoT	OoT	114.97	OoT	OoT	OoT	112.89
b15_C	OoT	OoT	OoT	9.14	OoT	OoT	OoT	11.93	OoT	OoT	OoT	133.30	OoT	OoT	OoT	131.60
b20_C	OoT	OoT	OoT	13.72	OoT	OoT	OoT	16.94	OoT	OoT	OoT	128.06	OoT	OoT	OoT	138.70

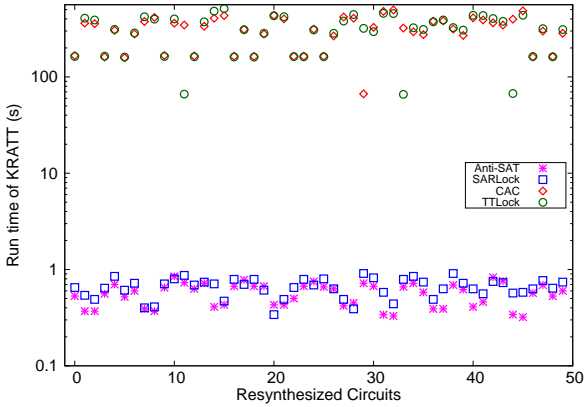


Fig. 6. Impact of resynthesis on the run-time of KRATT.

locked designs, except the *b20_C* circuit locked by CAC, and can guarantee the secret key on both SFLT. Note that KRATT finds a solution in less time than SCOPE on SFLT since it focuses on the locking unit rather than the entire locked design and uses a QBF formulation. Observe from Table III that the SAT-based attack and its variants cannot find a solution in the given time limit. Note that AppSAT and DDIP, which were previously shown to break AntiSAT and SARLock, respectively, fail simply because of the synthesis of locked designs and their large hardware complexity. However, KRATT can break these locked designs using a little computational effort. Note also that all the SFLT were broken through the QBF formulation. As the complexity of circuits and the number of key inputs increase, the run-time of KRATT also increases. Its run-time on DFLT is larger than that on SFLT since it explores the values of possible protected primary inputs exhaustively in circuits locked by DFLT after running the QBF solver.

In order to explore the impact of different circuit structures on the run-time of KRATT, we resynthesized the locked *c6288* circuit using different design efforts and delay constraints and generated 50 functionally equivalent but structurally different circuits. This circuit was chosen because its netlists locked

TABLE IV
RESULTS OF OL ATTACKS ON CIRCUITS LOCKED BY GEN-ANTI-SAT.

Circuit	SCOPE		KRATT	
	cdk/dk	CPU	cdk/dk	CPU
b14_C	9/12	14.38	127/127	106.32
b15_C	0/0	19.53	128/128	137.20
b17_C	0/0	51.54	128/128	533.35
b20_C	4/4	25.07	128/128	170.28
b21_C	0/0	24.80	128/128	173.41
b22_C	4/4	34.49	128/128	261.95

by DFLT require the longest run-time under the OG threat model. Fig. 6 presents the run-time of KRATT on these resynthesized circuits under the OG threat model.

Note that all the SFLT and DFLT were again broken through the QBF formulation and structural analysis, respectively. Observe from Fig. 6 that the impact of resynthesis on the run-time of KRATT on the circuits locked by SFLT is less than that on circuits locked by DFLT. The average (standard deviation) of these run-time results on resynthesized circuits locked by Anti-SAT, SARLock, CAC, and TTLock are computed as 0.56 (0.14), 0.65 (0.14), 306.85 (106.13), and 305.17 (121.57) in seconds and the ratio between the maximum and minimum run-time values on these locking techniques are found as 2.65, 2.67, 7.44, and 7.76, respectively.

As the second experiment set, we used all the locked circuits from the Valkyrie repository [31], including six ITC'99 benchmarks locked by Anti-SAT, CAS-Lock, Gen-Anti-SAT [7], and SARLock of SFLT and CAC and TTLock of DFLT using two different numbers of key inputs and 10 *synthesized* circuits for each benchmark. Thus, there exist a total of 720 locked circuits. Note that the Valkyrie tool in the given repository works as a security diagnostic tool, providing the critical signals as shown in Fig. 1 rather than an attack finding the secret key. However, KRATT was able to find the secret key of all circuits locked by SFLT and DFLT under the OL and OG threat models, respectively. The QBF formulation led to the secret key of all the 120 circuits locked by CAS-Lock and 112 out of 120 circuits locked by SARLock, i.e., a total of

TABLE V
 DETAILS OF LOCKED HELLO: CTF'22 CIRCUITS AND RESULTS OF OL AND OG ATTACKS.

Circuit	Locked Circuit Details				OL Attacks				OG Attacks	
	#inputs	#outputs	#gates	#key inputs	SCOPE		KRATT		SAT	KRATT
					cdk/dk	CPU	cdk/dk	CPU		
final_v1	767	757	17144	87	0/0	261.19	73/87	194.61	1117.05	350.22
final_v2	1452	1445	27440	47	0/0	39.73	34/46	99.45	OoT	2186.56
final_v3	522	1	93	29	0/0	1.94	25/29	62.26	20448.65	63.97

232 locked circuits. As an interesting result, Table IV presents solutions of attacks under the OL threat model on a single circuit for each benchmark locked by Gen-Anti-SAT using 128 key inputs, whose secret key could not be found through the QBF formulation due to the non-complementary function.

Observe from Table IV that while SCOPE can decipher a small number of key inputs on the entire locked circuit, KRATT can correctly decipher all the key inputs on the modified locking unit. Note that on *b14_C* circuit, it was proved that the secret key was found when the value of the missing key input was set to logic 0 or 1.

As the third experiment set, we used the circuits locked by SFL from the HeLLO: CTF'22 competition. Table V presents the details of the locked circuits taken from their bench files and the solutions of OL and OG attacks. We highlight that the FALL attack of [19] was not successful on these circuits.

Observe from Table V that KRATT can decipher all the key inputs, except *final_v2*, with high accuracy under the OL threat model, where no secret key was reported to be found by the competition participants. Also, it can find the secret key of all locked circuits using less run-time than the SAT-based attack under the OG threat model. The reason KRATT takes a longer time to break the *final_v2* circuit than others is due to a large number of promising protected primary input candidates.

V. DISCUSSION

In row/column-activated-LUT [12] and SFL-Flex [9] techniques, the original circuit corrupted on a number of protected primary inputs is corrected by the restore unit implemented in a read-proof hardware [27]. Since the restore unit is hidden from the adversary, and thus, the association of protected primary inputs with key inputs, KRATT cannot find the secret key as any other attack. However, for row-activated-LUT of [12] or SFL-Flex techniques, the structural analysis and exhaustive search of KRATT described in Section III-C can be used to find all the protected primary inputs and thus, the original circuit can be constructed after adding these values into the FSC using a comparator and XOR logic.

VI. CONCLUSIONS

This paper presented a novel removal and structural analysis attack, called KRATT, which can break a large number of state-of-the-art SAT-resilient logic locking techniques. KRATT utilizes the SAT and, most importantly, QBF formulations to find the secret key of locked circuits under both OL and OG threat models. It was shown that it can find the secret key of circuits locked by SFLT using the QBF formulation, can decipher values of key inputs with higher accuracy under the OL threat model with respect to a prominent OL attack,

and can break DFLT under the OG threat model, where well-known logic locking attacks cannot find a solution. It was also shown that hardware complexity, resynthesis, and the number of key inputs have a moderate impact on its run-time.

VII. ACKNOWLEDGMENT

This work was supported by the EU through the European Social Fund in the context of the project "ICT programme".

REFERENCES

- [1] Defence Science Board Task Force. (2015) On High Performance Microchip Supply Chain. [Online]. Available: <https://dsb.cto.mil/reports/2000s/ADA435563.pdf>
- [2] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *DATE*, 2008, pp. 1069–1074.
- [3] P. Subramanian, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *HOST*, 2015, pp. 137–143.
- [4] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *HOST*, 2016, pp. 236–241.
- [5] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE TCAD*, vol. 38, no. 2, pp. 199–207, 2019.
- [6] 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.
- [7] J. Zhou and X. Zhang, "Generalized SAT-Attack-Resistant Logic Locking," *IEEE TIFS*, vol. 16, pp. 2581–2592, 2021.
- [8] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, "What to Lock? Functional and Parametric Locking," in *GLSVLSI*, 2017, p. 351–356.
- [9] 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.
- [10] 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.
- [11] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the Approximation Resiliency of Logic Locking and IC Camouflaging Schemes," *IEEE TIFS*, vol. 14, no. 2, pp. 347–359, 2019.
- [12] K. Shamsi and Y. Jin, "In Praise of Exact-Functional-Secrecy in Circuit Locking," *IEEE TIFS*, vol. 16, no. 1, pp. 5225–5238, 2021.
- [13] Y. Shen and H. Zhou, "Double DIP: Re-Evaluating Security of Logic Encryption Algorithms," in *GLSVLSI*, 2017, pp. 179–184.
- [14] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *HOST*, 2017, pp. 95–100.
- [15] 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 *CHES*, vol. 10529, 2017, pp. 189–210.
- [16] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *DATE*, 2019, pp. 540–545.
- [17] 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.
- [18] 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.
- [19] D. Sirone and P. Subramanian, "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 (SFLH-d) - Unlocked," *IEEE TIFS*, vol. 14, no. 10, pp. 2778–2786, 2019.

- [21] Z. Han, M. Yasin, and J. J. Rajendran, "Does Logic Locking Work with EDA Tools?" in *USENIX Security Symposium*, 2021, pp. 1055–1072.
- [22] 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, pp. 418–440, Jul. 2021.
- [23] S. Patnaik, N. Limaye, and O. Sinanoglu, "Hide and Seek: Seeking the (Un)-Hidden Key in Provably-Secure Logic Locking Techniques," *IEEE TIFS*, vol. 17, pp. 3290–3305, 2022.
- [24] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability Assessment Tool and Attack for Provably-Secure Logic Locking Techniques," *IEEE TIFS*, vol. 17, pp. 744–759, 2022.
- [25] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE TETC*, vol. 8, no. 2, pp. 517–532, 2020.
- [26] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *ACM Journal of Electronic Testing*, vol. 35, pp. 273–291, 2019.
- [27] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-Proof Hardware from Protective Coatings," in *CHES*, 2006, pp. 369–383.
- [28] L. Aksoy. KRATT: A Removal and Structural Analysis Attack. [Online]. Available: <https://github.com/leventaksoy/kratt>
- [29] F. Lonsing. DepQBF Solver. [Online]. Available: <https://github.com/lonsing/depqbf>
- [30] M. Soos. Cryptominisat SAT Solver. [Online]. Available: <https://github.com/msoos/cryptominisat>
- [31] S. Patnaik. Valkyrie. [Online]. Available: <https://github.com/LL-Tools/Valkyrie>