Piercing Logic Locking Keys through Redundancy Identification

Leon Li and Alex Orailoglu
Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA 92093
xul065@ucsd.edu, alex@cs.ucsd.edu

***Abstract*—The globalization of the IC supply chain witnesses the emergence of hardware attacks such as reverse engineering, hardware Trojans, IP piracy and counterfeiting. The consequent losses sum to billions of dollars for the IC industry. One way to defend against these threats is to lock the circuit by inserting additional key-controlled logic such that correct outputs are produced only when the correct key is applied. The viability of logic locking techniques in precluding IP piracy has been tested by researchers who have identified extensive weaknesses when access to a functional IC is guaranteed.**

In this paper, we uncover weaknesses of logic locking techniques when the attacker has no access to an activated IC, thus exposing vulnerabilities at the earliest stage even for applications that seek refuge from attacks through functional opaqueness. We develop an attack algorithm that prunes out the incorrect value of each key bit when it introduces a significant level of logic redundancy. Throughout our experiments on ISCAS-85 and ISCAS-89 benchmark circuits, the attack deciphers more than half of the key bits on average with a high accuracy.

I. Introduction

Due to the ever-increasing complexity of sustaining an advanced fabrication facility, the integrated circuit (IC) industry is evolving into a fabless model. ICs are realized in a global supply chain with distributed vendors carrying out design, verification, fabrication, and testing. This paradigm shift has brought up various hardware-based security threats including hardware Trojans, Intellectual Property (IP) piracy, IC overbuilding, and reverse engineering, leading to a significant financial loss in the semiconductor industry [1] [2].

Logic locking, as a defense technique against the aforementioned threats, inserts locking circuitry such as XOR/XNOR key gates to obtain a locked netlist. The key gates are controlled by a secret key stored in a tamper-proof memory to prevent access by an attacker. A blueprint of the locked netlist is sent to a foundry that develops a costly mask and manufactures the ICs. A locked IC will be activated in a trustworthy environment to restore its correct functional behavior. An incorrect key, however, will force the IC to produce incorrect outputs.

Logic locking imposes significant area and performance overheads which must be justified by delivering a security guarantee verified under stringent scrutiny. Although certain mission-critical applications may be insensitive to overheads that typically run upwards of two-digit percentages, a run-of-the-mill consumer application has to utilize the technique judiciously lest its quest to preclude IP loss and piracy ends up making its IP products uncompetitive.

The research community has constantly been revealing vulnerabilities of logic locking and proposing countermeasures in turn. The overwhelming majority of existing work assumes the attack model of an activated IC being available to the malicious party [3] [4] [5]. Such an assumption may postpone the consideration of threats to when the IC becomes available in the open market. In a semiconductor world of rapid application dissemination, the advantages to an adversary of an early stage cracking of logic locking are dramatically accentuated. The necessity of examining its vulnerabilities on the manufacturing floor is even more pronounced for defense applications that may have been lulled into a false sense of security through their reliance on the presumed inaccessibility of an activated IC to attackers.

In this paper, we present an attack on logic locking that can be carried out solely on a locked netlist without an activated IC or a functional model. We observe that an incorrect key often results in a circuit with significantly more logic redundancy compared to the correct circuit. Our attack determines the likely value of key bits individually by comparing the levels of logic redundancy for each logic value. We demonstrate that the effectiveness can be further sharpened by targeting small sets of key inputs if computational resources are readily available. We evaluate the attack on 34 circuits from ISCAS-85 and ISCAS-89 benchmark suites encrypted with Random Logic Locking (RLL) [6] and Strong key-interference Logic Locking (SLL) [4]. We also shed light on the effectiveness of our attack on multi-layer defenses such as a SARLock [7] or Stripped Functionality Logic Locking (SFLL) [8] integrated with RLL or SLL.

The rest of this paper is organized as follows. In Section II, we present an overview of related work. Section III illustrates the existence of the vulnerability in two-level and multi-level circuits. Section IV describes the proposed attack strategy and optimization. Section V discusses the experimental results. Section VI concludes the paper.

II. Related Work

Earlier logic locking techniques such as Random Logic Locking (RLL) [6] and Fault analysis-based Logic Locking (FLL) [3] select locations of key gates based on ad hoc heuristics. They are broken by the sensitization attack [4] which sensitizes individual key bits to the outputs by tailoring input patterns using ATPG tools. Strong Logic Locking (SLL) is proposed to insert key gates with complex interference that thwarts individual sensitization.

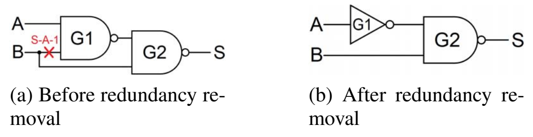
A powerful attack that has made significant inroads into breaking all the prior logic locking techniques is the Boolean Satisfiability (SAT) based attack [5]. The SAT attack identifies Distinguishing Input Patterns (DIPs) using SAT solvers, obtains the expected outputs from an oracle (activated IC), and prunes out a set of incorrect keys by each DIP. Subsequent to the publication of [5], research on logic locking has materially shifted to the task of identifying defenses against the SAT attack. Techniques such as Anti-SAT [9] and SARLock [7] thwart an SAT attack by minimizing the distinguishing capability of input patterns. The most recent defense that resists all known attacks is the Stripped Functionality Logic Locking (SFLL) [8]. All the existing SAT attack resilient logic locking techniques result in minimum output corruptions so they are often deployed in conjunction with gate selection-based locking techniques such as SLL for sufficient output entropy [7] [8].

Along with the continuous evolution of attacks that assumes functional IC availability, structural attacks are also beginning to be introduced. One group of structural attacks aims at certain SAT attack resilient locking circuitry and attempts to isolate the original logic cone, such as the Signal Probability Skew (SPS) attack [10], AppSAT Guided Removal (AGR) attack [11], and bypass attack [12]. A recent structural attack targeting gate selection-based locking techniques is the desynthesis attack [13]. The attack re-synthesizes the locked netlist with a random key and uses hill climbing search to find a key that yields maximum similarity between the locked netlist and the re-synthesized netlist in terms of the gate count. The attack presumes a sound knowledge of the synthesis tool and options, rendering its efficiency uncertain for large-scale designs where design decisions can hardly be inferred by an attacker.

III. Logic redundancy in a locked netlist

The single stuck-at fault where an individual signal is assumed to be stuck at a logic 0 or logic 1 is the most widely adopted model for test pattern generation. The presence of untestable stuck-at faults may increase power consumption and propagation delay, as well as prevent the detection of other faults. In combinational circuits or full-scan sequential circuits, untestable faults are associated with redundancy so they can be eliminated by simple rules [14].

As an example, consider the circuit shown in Fig. 1a. It defines the function *((AB)'B)'*. Now suppose that the B input of the NAND gate has a stuck-at-1 fault. This will deliver the function *(A'B)'* which is equivalent to the fault-free function, rendering the fault untestable. The circuit can be optimized by setting the untestable B input of the NAND gate to be a 1 and consequently replacing the NAND gate with a NOT gate, as shown in Fig. 1b.



(a) Before redundancy removal

(b) After redundancy removal

Fig. 1: An example of an untestable fault. The output function remains constant in the face of the stuck-at-1 fault at the *B* input of the NAND gate.

All commercially available synthesis tools include a redundancy removal engine which is often employed multiple times during optimization [15]. A synthesized netlist should have a near-perfect stuck-at fault coverage regardless of the specific algorithm implemented. In combinational logic locking, key gates are incorporated into a netlist after logic synthesis [13]. If the correct activation key is applied, the fault coverage of the unlocked circuit is equivalent to that of the original circuit [16]. Yet, an incorrect key modifies the circuit's structure which may hinder the testability of certain faults.

Example: Such a vulnerability is illustrated in Fig. 2. The key gate is inserted at the output of G3 with the correct key being a 0. When we set K1=0, the circuit implements the Boolean function *f = (A + BC)(C' + D)* with no untestable faults. When we set K1=1, the circuit implements *g = (A + BC)(C' + D)'*. The stuck-at-1 fault at the C input of G1 is then untestable because it would convert the output function to *h = (A + B)(C' + D)'* which is equivalent to the fault-free function *g*. The existence of redundant hardware only when K1=1 justifies the deduction that the correct key is in all likelihood a 0.

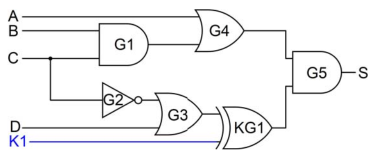
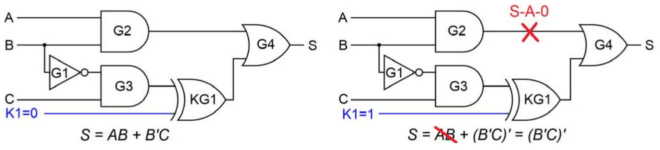
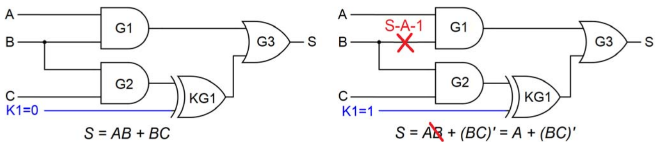


Fig. 2: A circuit locked with one XOR key gate. The circuit has redundant faults only when K1=1.

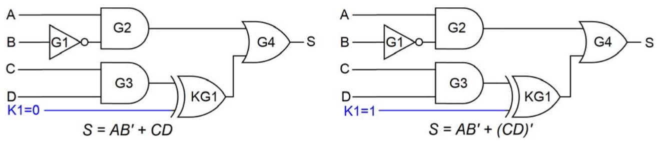
A rigorous substantiation of this insight can be garnered by examining a netlist topology. In a two-level single-output AND-OR circuit, the fundamental building block of logic design, suppose that we insert one key gate to protect a product term. The reader should note that if the protected product term shares any input variable with another product term, an incorrect key will without fail introduce redundancy. Specifically, if the shared input variable appears as the same literal in two terms, the redundancy for the incorrect key value can be verified through the application of the *absorption* law on its Boolean representation. Otherwise, if the shared input variable manifests as a complemented literal in one and uncomplemented in the other, the redundancy can be verified by applying the *elimination* law. The two conditions are demonstrated in Fig. 3a and Fig. 3b, respectively. Perhaps, interested readers may find it instructive to replicate this insight from the vantage point of Karnaugh Maps as well.



(a) A two-level circuit where an untestable fault exists on the stem of a product term when K1=1



(b) A two-level circuit where an untestable fault exists on the branch of B to G1 when K1=1.



(c) A two-level circuit without redundancy for either key value due to its tree structure

Fig. 3: Three illustrative functions showing the location of untestable faults for both logic values of the key bit.

Notice that in both Fig. 3a and Fig. 3b, signal B fans out and reconverges at the OR gate. This structure, known as reconvergent fanout, is a necessary condition for a redundant single stuck-at fault. Therefore, if the key gate is inserted after a product term with a disjunctive input cone, the key value would fail to affect the testability of the circuit, as illustrated in Fig. 3c. Empirically, most functions whether two-level or multi-level are not dominated by such tree-like structures. Especially, the state transition function of Finite State Machines (FSMs) which constitutes a major application arena of logic locking shall presumably have intricate fanout and convergence structures. A key gate anywhere in the circuit is likely to lie within multiple reconvergent fanouts. We can often observe a burst in the number of untestable faults for an incorrect key similar to what happens in a two-level circuit. An illustrative example of such a multi-level circuit is provided in Fig. 2.

IV. Description of Attack Algorithm

A. Attack Model

We assume that an attacker has access to a locked netlist but not necessarily an activated IC. The locked netlist can be obtained from reverse engineering the layout or mask. The attacker can distinguish key inputs and primary inputs, and consequently identify the locations of the key gates. The attacker's goal is to correctly determine as many key bits as possible.

B. Attack Strategy

By virtue of a structural analysis, our redundancy attack can decipher key bits individually. All the key bits are initialized as unspecified and are modeled as XOR control points in the course of redundancy identification until their values are determined. The logic value of one key bit is enumerated, and a redundancy identification tool will extract untestable faults for each logic value assignment. Note that the testability of a fault is determined assuming an arbitrary assignment to unspecified key bits, and therefore once a fault is identified to be untestable, it will remain untestable regardless of further key value assignments. We assess the likelihood of a key bit value based on the number of untestable stuck-at faults that result from a key bit assignment.

The sensible value of a key bit is determined if it is associated with significantly fewer untestable faults compared to its counterpart. If the original circuit (before logic locking) does not contain any logic redundancy, a logic value associated with an untestable fault is bound to be the wrong key value. In the situation where the original circuit contains logic redundancy, redundant faults may exist not only for a wrong key but also for the correct key. Suppose that the original circuit contains untestable faults $U_{original}$. Due to the increased testability provided by XOR control points, the set of untestable faults in the locked circuit U_{locked} satisfies the following relationship $U_{locked} \subseteq U_{original}$. When a single correct key bit value is assigned, a small portion of faults in $U_{original} - U_{locked}$ may be proven to be untestable due to decreased controllability. Upon the application of the incorrect key bit value, on the contrary, the circuit is likely to exhibit a significant number of untestable faults as explicated in the last section. We leave the assignment to the key bit fluid if a comparable number of untestable faults exists for both settings.

There are situations where multiple key bits cannot be deciphered independently. The existence of untestable stuck-at faults is a precondition for deciphering a key bit. A fault may have multiple sensitized paths that depend on unspecified key bits. It cannot be declared to be untestable until all the paths are shown to fail in propagating the fault effect to a primary output. As the attack determines more key bits, unspecified key bits are muted into constant values which prod more untestable faults to emerge with another tentative assignment. We update the netlist with determined key bits and repeat the procedure on unspecified key bits individually until no further progress can be made. In the iterative process, untestable faults in each intermediate determined circuit, including U_{locked}, are dismissed from further comparisons to minimize the adverse impact of inherent redundancy.

Example: Consider a circuit locked with two key gates as shown in Fig. 4. The correct key value is supposed to be 00. When K1 is enumerated, the circuit contains no untestable fault for either logic value. When K2 is enumerated, untestable faults are revealed only when K2=1. We update the netlist with the determined value of K2 (which is a 0) and attempt the examination of K1 again. There is no untestable fault when K1=0 and yet when K1=1, the stuck-at-1 fault at the B input of G2 is untestable because neither of its two sensitized paths through G5 or G6 can be justified. Therefore, we determine the value of K1 to be a 0.

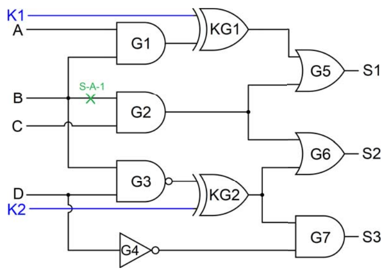


Fig. 4: A circuit locked with 2 key gates. K1 can be deciphered only after determining K2.

The approach can be further sharpened by brute forcing groups of key bits. When multiple key bits are back-to-back or they dominate parallel reconvergent paths, it is possible that none of them can be determined individually while their consideration in unison may yield a relationship between them. When we enumerate a set of two key bits, for instance, we make different deductions depending on the amount of logic redundancy for 4 logic value combinations. The deductions always conform to the lowest number of untestable faults and are summarized in Table I. *Low* (in most cases zero) and *High* stands for the comparative number of untestable faults. Absent table entries indicate a failure to decipher a key bit forthrightly. When two key bits are enumerated, logic redundancy shall only be identified within the faults that converge with both of them. The testability of other faults is never affected by both key bits concurrently. That is to say, a pair of key bits does not have to be examined if they belong to disjoint logic cones.

Example: Consider a circuit locked with two key gates in Fig. 5. The correct key is K1=K2=0. There is no redundancy whatsoever if we enumerate a single key bit. We then attempt enumerations of the pair of key bits. The circuit contains no untestable fault when K1 and K2 are equivalent, i.e. {K1=0, K2=0} or {K1=1, K2=1}. On the other hand, when K1 and K2 are not equivalent, there exist untestable faults in the circuit. We, therefore, conclude that K1=K2 and merge the two key bits into one key bit.

The best discrimination capability is achieved when the attack examines individual key bits or pairs of two. Under such circumstances, an imbalance in the number of untestable faults can often reduce the key size by fixing a key bit or correlating two equivalent key bits. When it comes to groups of three and more key bits, although we are more likely to find untestable faults in each assignment, the probability of immediately reducing the key size is lowered. For example, a deduction table with 256 rows has to be created for 3 bits but only 40 rows[1] enable deductions to reduce the key size. Furthermore, the number of sets grows polynomially when the cardinality increases, with an increasing number of trials needing to be made in each set as well. As a result, we constrain our analysis to groups of two key bits for efficiency considerations.

TABLE I: Logical deductions for two key bits.

Untestable faults for (K1,K2)				Deduction
(0,0)	(0,1)	(1,0)	(1,1)	
Low	High	High	High	k1=0, k2=0
High	Low	High	High	k1=0, k2=1
High	High	Low	High	k1=1, k2=0
High	High	High	Low	k1=1, k2=1
Low	High	High	Low	k1=k2
High	Low	Low	High	k1=k2'
Low	Low	High	High	k1=0
Low	High	Low	High	k2=0
High	High	Low	Low	k1=1
High	Low	High	Low	k2=1

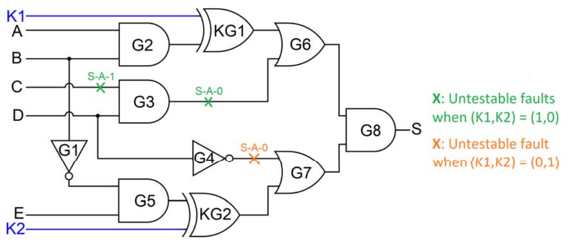

K1
A
B
C
D
E
K2
G1
G2
KG1
G6
S-A-1
G3
S-A-0
G4
S-A-0
G5
KG2
G7
G8
S
X: Untestable faults when (K1,K2) = (1,0)
X: Untestable fault when (K1,K2) = (0,1)

Fig. 5: A circuit locked with 2 key gates. Neither key bit can be determined individually, but they are identified to be equivalent upon pair-wise examination.

V. Evaluation

A. Methodology

The effectiveness of the proposed technique is analyzed on 34 circuits from the ISCAS-85 combinational and the ISCAS-89 sequential benchmark suites locked by RLL and SLL. The key size is selected to be 5% of the number of gates in the circuit to ensure a realistic overhead rate. We implemented the algorithm in Java and integrated the ATALANTA [17] ATPG tool for identifying untestable stuck-at faults. Experiments were run on an Intel Core i5-8400 CPU with 16GB of RAM. The number of maximum backtrackings for the FAN algorithm employed in ATALANTA is set to 200.

B. Results

The summary of results on RLL is presented in Table II. To ensure elevated statistical significance, we randomly encrypt each circuit 10 times and tally the average number of deciphered key bits. The attack algorithm deciphers 52.01% of the key bits on average, among which 96.22% are correct and 3.78% are incorrect[2]. It is interesting to note that in 82% of the netlists we examined, the full set of the deciphered key bits with no exception are correct.

It turns out that the bulk of the results are delivered from targeting singletons, i.e, most of the deciphered key bits are derived independently or sequentially. We do observe improvements by targeting pairs of key bits. Throughout the experiments, doubleton key assignment examination is capable of correctly breaking 4 more key bits in the best case. The success of our attack varies across circuits and the locations of the key gates as illustrated in Fig. 6. The box shows

[1]The successful rows either result in a declaration regarding the value of some key bits such as "K1=0 and K2=1", or represent some key bits as a function of others such as "K1=K2⊕K3".

[2]In the interest of time, circuits marked with * are only attacked through singleton key assignment examination.

TABLE II: Performance of redundancy identification attack on ISCAS 85 and ISCAS 89 benchmark circuits locked with RLL.

Circuit	Number of Gates	Number of Key Gates	Average Deciphered Bits from Singleton	Average Deciphered Bits from Doubleton	Average Deciphered Bits in Total	**Average Deciphered Percentage**	Average Correct Bits	Average Wrong Bits	**Average Correct Bits Percentage**	Average Runtime (s)
c432	160	8	3.9	0.4	4.3	**53.75%**	4.3	0	**100.00%**	0.9
c499	202	10	1.3	0.1	1.4	**14.00%**	1.1	0.3	**78.57%**	9.4
c880	383	19	7.6	0.1	7.7	**40.53%**	7.7	0	**100.00%**	12.7
c1355	546	27	16.1	0.7	16.8	**62.22%**	16.4	0.4	**97.62%**	26.7
c1908	880	44	26.2	0.2	26.4	**60.00%**	26.1	0.3	**98.86%**	125.3
c2670	1193	59	46.2	0.7	46.9	**79.49%**	41.1	5.8	**87.63%**	588.1
c3540	1669	83	40.8	1.2	42.0	**50.60%**	40.9	1.1	**97.38%**	3010.2
c5315*	2307	115	5.9	0.0	5.9	**5.13%**	5.9	0	**100.00%**	255.4
c6288*	2416	120	70.2	0.0	70.2	**58.50%**	68.2	2	**97.15%**	1245.0
c7552*	3512	175	124.6	0.0	124.6	**71.20%**	117.4	7.2	**94.22%**	7056.4
s208	96	5	2.9	0.0	2.9	**58.00%**	2.9	0	**100.00%**	0.2
s298	119	6	3.0	0.1	3.1	**51.67%**	3.1	0	**100.00%**	0.3
s344	160	8	4.1	0.0	4.1	**51.25%**	4.1	0	**100.00%**	0.5
s349	161	8	3.9	0.0	3.9	**48.75%**	3.9	0	**100.00%**	0.6
s382	158	8	4.4	0.2	4.6	**57.50%**	4.6	0	**100.00%**	0.6
s386	159	8	4.0	0.0	4.0	**50.00%**	4.0	0	**100.00%**	0.7
s400	162	8	4.2	0.0	4.2	**52.50%**	3.7	0.5	**88.10%**	0.7
s420	196	10	4.4	0.1	4.5	**45.00%**	4.5	0	**100.00%**	1.4
s444	181	9	4.6	0.0	4.6	**51.11%**	4.3	0.3	**93.48%**	0.9
s510	211	11	7.9	0.1	8.0	**72.73%**	8.0	0	**100.00%**	0.8
s526	193	10	6.8	0.1	6.9	**69.00%**	6.8	0.1	**98.55%**	0.8
s641	379	19	3.1	0.1	3.2	**16.84%**	3.2	0	**100.00%**	11.6
s713	393	20	4.9	0.0	4.9	**24.50%**	4.7	0.2	**95.92%**	15.0
s820	289	14	8.9	0.3	9.2	**65.71%**	9.2	0	**100.00%**	4.5
s832	287	14	7.9	0.3	8.2	**58.57%**	8.2	0	**100.00%**	6.2
s838	390	19	8.6	0.3	8.9	**46.84%**	8.9	0	**100.00%**	14.9
s953	395	20	12.7	0.6	13.3	**66.50%**	13.3	0	**100.00%**	8.9
s1196	529	26	16.3	0.3	16.6	**63.85%**	16.6	0	**100.00%**	26.4
s1238	508	25	16.6	0.3	16.9	**67.60%**	13.4	3.5	**79.29%**	22.5
s1423	657	33	14.2	0.1	14.3	**43.33%**	13.4	0.9	**93.71%**	90.1
s1488	653	33	22.2	0.7	22.9	**69.39%**	22.9	0	**100.00%**	37.9
s1494	647	32	23.1	0.6	23.7	**74.06%**	23.0	0.7	**97.05%**	27.0
s5378	2779	139	56.4	0.0	56.4	**40.58%**	55.6	0.8	**98.58%**	4142.9
s9234*	5597	280	57.4	0.0	89.6	**32.00%**	79.4	10.2	**88.62%**	2726.6

the 25%-75% percentile distribution of the percentage of correctly deciphered key bits, the red horizontal line shows the median value, and the two black lines show the minimum and maximum values. The inefficiency of our attack in tree-heavy circuits such as c499, c5315, and s641 is to be expected because faulty inverters do not necessarily leave any structural traces in terms of redundant hardware.

Even though previous attacks can usually break all the key bits correctly, we invite readers to note that our attack does not require a functional IC yet significantly compromises the obfuscation offered by logic locking. Compared to the desynthesis attack [13] which assumes a knowledge of the synthesis process and recovers a complete but faulty key vector, our attack makes a minimal assumption on an attacker and achieves a false positive rate that is smaller by more than one order of magnitude. The significant reduction of the effective key size may also facilitate brute force attacks and SAT-based algorithmic attacks even for large circuits where they would traditionally run into scalability issues.

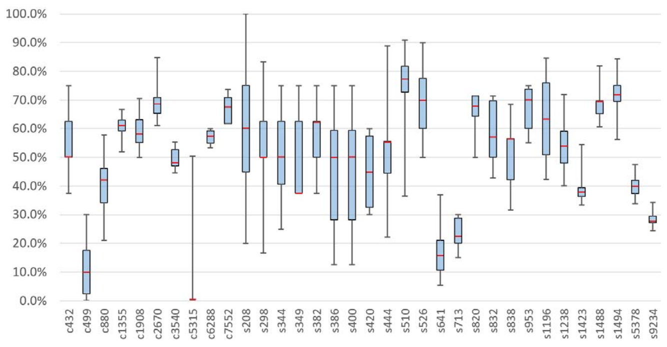


Fig. 6: Correct deciphering box plot percentage of RLL-locked ISCAS 85 and ISCAS 89 benchmark circuits.

We examine the impact of inherent redundancy from the original netlist on our attack. s344, s382, s641, s820, s1196, s1488 are resynthesized from s349, s400, s713, s832, s1238, s1494 respectively after removing all redundancies in full-scan mode [18]. The experimental results confirm our analysis that the attack will return no incorrect key bits whatsoever if the original circuit is free of logic redundancy. It turns out that our attack is consistently effective even if the original netlist contains redundant stuck-at faults, although a small percentage of the deciphered key bits may be incorrect.

The summary of results on SLL is presented in Table III based on one copy provided by [5]. Within the samples, we observed an average effectiveness slightly lower than that of RLL. c1355 exhibits the most significant degradation because it implements the same function as c499 by expanding each XOR gate into its 4 NAND gate equivalent with the key gates

TABLE III: Performance of redundancy identification attack on ISCAS 85 circuits locked with SLL.

	Average Deciphered Bits	**Average Deciphered Percentage**	Average Correct Bits	**Average Correct Percentage**
c432	8	**100%**	8	**100%**
c499	0	**0%**	0	**-**
c880	4	**21.05%**	4	**100%**
c1355	0	**0%**	0	**-**
c1908	12	**27.27%**	12	**100%**
c2670	52	**64.41%**	38	**73.08%**
c3540	18	**19.28%**	16	**88.89%**
c5315	49	**42.61%**	49	**100%**
c7552	113	**64.20%**	92	**81.42%**

consistently being inserted outside the group of NAND gates. The performance is therefore brought down to the same level as that of c499. At the same time, we observe an exceptional success in c432 and c5315. Therefore, the interference-based locking continues to be susceptible to our attack as it takes no notice of the irrational redundancy under incorrect key bits.

Essentially, the complexity of our algorithm is dominated by the complexity of the redundancy identification algorithm being employed on specific circuits. We use ATALANTA [17], an academic ATPG tool, to perform the analysis. A resourceful attacker will in all likelihood employ specialized redundancy removal engines and more advanced ATPG algorithms to shorten the run time on larger circuits. With more computational resources, attackers can further exploit the vulnerability to its full potential by examining combinations of more key bits.

SAT attack resilient techniques are often deployed in conjunction with gate selection-based locking techniques for a sufficient output entropy, as shown in Fig. 7. The logic cone, whether it is unmodified or functionally stripped, is locked with RLL or SLL controlled by $|K1|$ key bits and the SARLock or the Hamming distance calculator in SFLL is controlled by another $|K2|$ key bits. Our attack is still able to break the set of K1 bits because both SARLock and SFLL have a payload XOR gate at the primary output which does not affect the testability of any faults.

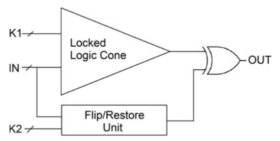


Fig. 7: Two layer logic locking with gate selection based locking controlled by K1 and SAT-resilient locking controlled by K2.

VI. Conclusion

In this paper, we reveal a novel vulnerability of logic locking based on the logic redundancy introduced by incorrect key bits. Contrary to previous work that relies on the availability of an activated IC to jump-start the attack, we present an attack that analyzes the logic redundancy for different key assignments to individual or pairs of key bits. We demonstrate the existence of such vulnerability in two-level and multi-level circuits. We develop a variety of strategies to improve the attack performance for circuits that contain inherent redundancy and for circuits whose redundancy manifests due to interference of key gates. Throughout the experiments, we can recover approximately half of the key bits with a high accuracy in both RLL and SLL, questioning the viability of logic encryption techniques for consumer applications wherein the already elevated area and performance overhead does not lend itself to ready duplication without loss of competitiveness. The outlined attack should be of concern furthermore to security sensitive applications that rely on function IC inaccessibility as it can be mounted through a netlist that is easily leaked on the manufacturing floor. It remains imperative to develop a gate selection-based locking technique that is resilient to the proposed redundancy identification attack.

References

[1] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *IEEE Computer*, vol. 102, no. 8, pp. 1283–1295, 2014.

[2] M. Pecht and S. Tiku, "Bogus! electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum*, pp. 37–46, 2006.

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

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

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

[6] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *IEEE Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[7] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, IEEE, 2016.

[8] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, "What to lock?: Functional and parametric locking," in *Proceedings of the Great Lakes Symposium on VLSI*, pp. 351–356, ACM, 2017.

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

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

[11] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, no. 99, 2017.

[12] 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 *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 189–210, Springer, 2017.

[13] M. E. Massad, J. Zhang, S. Garg, and M. V. Tripunitara, "Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism," *arXiv preprint arXiv:1703.10187*, 2017.

[14] K.-T. Cheng, "On removing redundancy in sequential circuits," in *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 164–169, ACM, 1991.

[15] J.-H. R. Jiang and S. Devadas, "Logic synthesis in a nutshell," in *Electronic Design Automation*, pp. 299–404, Elsevier, 2009.

[16] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of logic encrypted chips: Pre-test or post-test?," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 139–144, IEEE, 2016.

[17] H. Lee and D. Ha, "Atalanta: An efficient ATPG for combinational circuits." Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.

[18] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934, IEEE, 1989.