Article

Experimental Evaluation of SAT Attack on Logic Locking

Milan Vojvoda *, Viliam Hromada, Filip Janikovský, Jozef Kučerák, Matúš Jókay

Posted Date: 7 March 2024

doi: 10.20944/preprints202403.0447.v1

Keywords: logic locking; piracy protection; SAT attack

Article

Experimental Evaluation of SAT Attack on Logic Locking

Milan Vojvoda *, Viliam Hromada, Filip Janikovský, Jozef Kučerák and Matúš Jókay

Institute of Computer Science and Mathematics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovičova 3, 841 04 Bratislava, Slovakia; viliam.hromada@stuba.sk (V.H.); filip.janikovsky@stuba.sk (F.J.); jozef.kucerak@stuba.sk (J.K.); matus.jokay@stuba.sk (M.J.)

* Correspondence: milan.vojvoda@stuba.sk

Abstract: To secure logic circuits against the illegal copying the insertion of the so-called key bit inputs using XOR/XNOR gates has been proposed in [3]. The inserted gates modify (lock) the behavior of the original circuit and only the correct choice of the secret key bits guarantees the equivalent behaviour of the locked circuit to the original circuit. To break this kind of security, i.e. to find the secret key bits, there was a SAT solver based attack proposed in [5]. The aim of this work is to experimentally verify this attack on several locked circuits with various number of primary inputs and outputs and with various number of gates. Except for the random placement of the key gates we evaluate also the positioning with chaining elimination. The obtained and presented results show that we were able to unlock 10 of 11 circuits within few minutes on a common PC.

Keywords: logic locking; piracy protection; SAT attack

1. Introduction

Integrated circuits (or logic circuits), constitute a fundamental aspect of the modernized world, catering to the daily needs of millions. The rapidly emerging era of Internet of Things (IoT) only accentuates this claim. As IoT devices increasingly permeate critical sectors, such as healthcare, automotive, and home automation, their security implications cannot be overstated. These devices often handle sensitive data and control systems integral to safety and privacy. The challenges confronting ICs revolve around the considerable expenses associated with circuit design and the initial acquisition of manufacturing equipment. The substantial costs incurred during the design phase render integrated circuit fabrication economically viable only in scenarios involving high-volume production. To address the second challenge, a strategic solution involves the bifurcation of IC design and fabrication processes. In this approach, fabrication is delegated to specialized foundries capable of producing circuits for multiple design companies. This decentralized model not only mitigates the financial burden of individual manufacturing setups but also fosters collaboration and efficiency in the production of integrated circuits. Ultimately, these intricate electronic components continue to drive technological advancements, underscoring their indispensability in contemporary society.

Outsourcing the fabrication of integrated circuits to external entities introduces a heightened susceptibility to forgery, piracy, and unauthorized reproduction, necessitating vigilant measures to mitigate associated risks. The absence of stringent monitoring and direct control over the fabrication process in external companies exposes the vulnerability of intellectual property to potential attacks. A malicious foundry, exploiting this vulnerability, may present a product with compromised IC security as its own, engaging in illegal copying and unauthorized sale under its brand. Such security risks, commonly referred to as subcontractor attacks, pose substantial economic threats to prominent IC design companies. Conversely, perpetrators stand to gain significant financial advantages by compromising designs, compelling designers to create circuits fortified against potential attacks. Striking a delicate balance between outsourcing efficiency and safeguarding intellectual property becomes imperative in navigating the intricate landscape of integrated circuit fabrication.

The paper is organized as follows. A brief overview of some chosen security mechanism for securing logic circuits is presented in Section 2. The used attack on logic locking using a SAT solver

 doi:10.20944/preprints202403.0447.v1

is stated in Section 3. The results obtained in the experiments are presented in Section 4. Finally, conclusions and possible way for future research are stated in Section 5.

2. Logic Circuits and Their Security

Because of these risks, different ways of securing integrated circuits began to emerge. A brief history of logic locking can be found in [14]. One of the first security methods was the passive protection, which was based on a unique identifier (ID) [6]. Each circuit contained its own identifier that was physically engraved on it. This protection facilitated the detection of fake products but did not interfere with its production process.

The first active scheme for protecting logic circuits was the generation of identifiers. Each integrated circuit generates a unique identifier during initialization. This identifier remains encrypted. Only the intellectual property holder can activate the circuit using a unique identifier to generate the unlocking key.

A possible extension, which was described in the work [1], is the use of an identifier that would be integrated within the Finite State Machine (FSM). Only a circuit designer who knows the extended FSM structure can know the key to activate the circuit. Designing active protection of integrated circuits is challenging because different types of overheads (for security as well) increase the price of the product and make it less attractive to the consumer [2].

To secure logic circuits against the illegal copying the insertion of the so-called key bit inputs using XOR/XNOR gates has been proposed in [3]. The inserted gates modify (lock) the behavior of the original circuit and only the correct choice of the secret key bits guarantees the equivalent behaviour of the locked circuit to the original circuit.

The use of a set of lookup tables to the circuit such that every path from an input to an output goes through a lookup table was proposed as another locking scheme in [10]. The process of logic locking performed on a gate-level was automated in [7], where the authors developed a software in Python, that can be integrated with the existing digital synthesis tools. The method based on the insertion of key gates in the so-called interference mode, proposed in [8], is claimed to have less hardware overhead, higher speed, and to be more effective against SAT attacks than the existing conventional methods. A provable secure logic locking scheme that can thwart SAT-based attacks was proposed in [9]. The authors in [15] propose a generalized method for logic locking that is resistant against SAT attack. The locking scheme with low complexity resistant against the so-called removal attacks was proposed in [16]. The complexity analysis of the SAT attack on logic locking can be found in [20]. The logic locking schemes designed on lower levels were proposed in [11–13].

In the following we will omit the technical realization of ICs and discuss the logic circuit level only. Before the circuit is activated, it is necessary in the design to think about incorporating the key into the circuit and thus connecting the key bits to the existing logic gates of the circuit, while it is necessary to preserve the required functionality. There are several techniques that use the insertion of key bits into a circuit. They differ in the type of gates for connecting the key bits into the circuit and also in the way these gates are placed. The list of gates used in this work is shown in Figure 1. We will briefly describe some possible ways of inserting the key into the circuit in the next subsections.

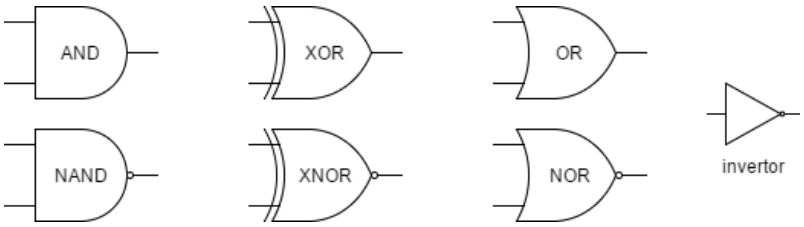

Figure 1. List of gates used in this work.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 March 2024 doi:10.20944/preprints202403.0447.v1

2.1. Inserting Key Bits into a Gate Scheme Using XOR and XNOR Gates

Combinational locking is performed in the most important IC modules by inserting XOR or XNOR gates at selected locations with added checks associated with the key. This technique was proposed in [3]. With the correct key, the locked circuit behaves equivalently to the unlocked one, i.e. $C'(\vec{x},\vec{k}) \equiv C(\vec{x})$, where $\vec{x}$ is the input, $\vec{k}$ is the key, $C(\vec{x})$ represents the output from the unlocked circuit and $C'(\vec{x},\vec{k})$ represents the output from the locked circuit. Otherwise, the circuit behaves differently. In Figure 2 (**b**) we can see the combinational locking of the integrated circuit using the XNOR gate.

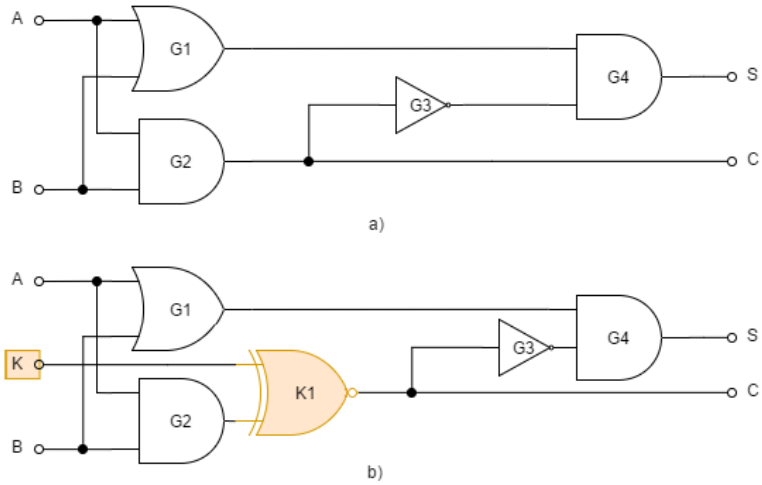

Figure 2. Combination lock. (**a**) Represents a half-adder circuit. (**b**) Shows a combination-locked half-adder (the half-adder circuit with an XNOR gate). A locked adder is equivalent to an unlocked one only if the key K with the value 1 is brought to the input.

The security prerequisite of a locked circuit is the uniqueness of the key, i.e. the fulfillment of the following relationship:

$$\exists!\vec{k}\forall\vec{x}; C'(\vec{x},\vec{k}) \equiv C(\vec{x}) \tag{1}$$

The claim says that there must be exactly one key such that for each input of the locked circuit we get the same output as the unlocked one. If this statement does not hold, this expression becomes a Boolean function to find the combination of keys. The length of the key should be long enough to withstand a brute force attack. A key length of at least 80 bits is therefore recommended.

2.1.1. Random Placement of Key Gates

The easiest for inserting key bits into the logic circuit is to choose random location for key gates. The EPIC algorithm [3] uses the following simple approach to protect the circuit from piracy. Firstly, k random wires in the circuit marked as x_i and their corresponding key bits k_i are chosen. Wires should avoid critical areas and overloaded regions. For each selected wire x_i, its source is disconnected and either an XOR gate ($x'_i = x_i \oplus k_i$) or an XNOR gate ($x'_i = x_i \overline{\oplus} k_i$) is inserted. The choice of XOR or XNOR gate depends on the value of the bit k_i. If $k_i = 0$, an XOR gate is added, otherwise, an XNOR gate is added. Use of identity from Table 1, $x \oplus y = \overline{x} \overline{\oplus} y$, complicates reverse engineering because the designer or software can replace the selected XOR gate with an XNOR gate and inverter or vice versa.

Table 1. Truth table of the identity $x \oplus y = \overline{x} \overline{\oplus} y$.

x	y	$\overline{x}$	$x \oplus y$	$\overline{x} \overline{\oplus} y$
0	0	1	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	0	0

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 March 2024 doi:10.20944/preprints202403.0447.v1

However, random placement of gates is not in general a very secure solution for locking the circuit. The value of an unknown key bit can be determined if it is sensitive to the output without being masked by other key and input bits. If we assume that an attacker knows the layout of the circuit, he can specify an input pattern that will propagate the correct key value to the output. Consider the locked circuit in Figure 3. The key bit $K1$ will be sensitive on the output $O1$ if the second value on the gate $G8$ is 0 (non-controlling value on the OR gate). This can be achieved by setting the input pattern to $A = 1$, $B = 0$, and $C = 0$. If an attacker has access to the functional circuit, he can use this pattern to find the value of $K1$ at the output of $O1$. For example, if the value of $O1 = 0$ for this pattern then $K1 = 0$, otherwise $K1 = 1$.

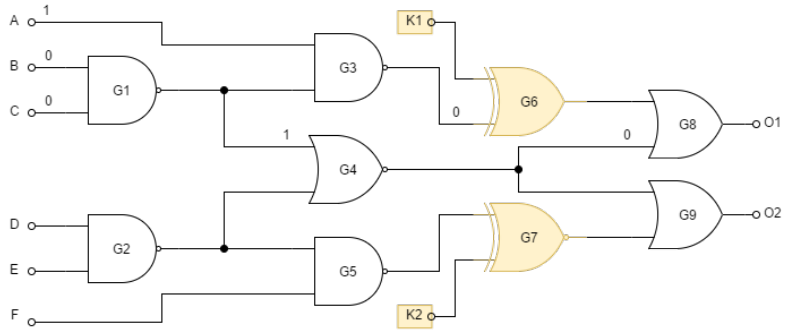

Figure 3. Vulnerable locked circuit. The value of the key $K1$ is sensitive to the output and is therefore propagated to the output $O1$ by setting a suitable pattern.

To prevent such attacks, the sensitivity of the keys must be ensured by placing the key gates in such places where the propagation of key bits is possible only if certain conditions are met. In Figure 4 is a circuit with the same functionality as in the Figure 3 but with differently positioned key gates. If the attacker wants to propagate the value of one of the keys, then a 0 must be forced on one of the inputs of $G4$. To reach this value, the attacker must control key inputs that are inaccessible. They are stored in a breach-indicating memory inside the structure to prevent attacker access. Thus, propagating the key value to the output is infeasible.

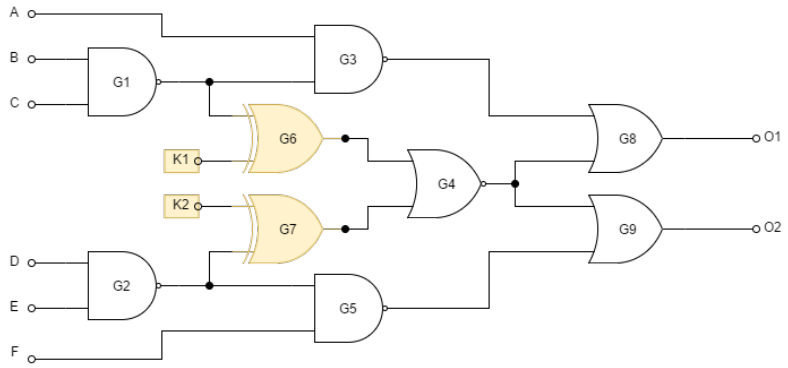

Figure 4. Invulnerable locked circuit. None of the key bits are output sensitive.

2.1.2. Chained Key Gates

In the case of connecting several key gates one after the other, we speak of chained key gates. Chained key gates reduce the effort for an attacker because they increase the space of valid keys. In the case of n chained gates, the space of valid keys increases from the only one (optimal) to 2^{n-1}.

In Figure 5 the attacker replaces the two sequentially running key gates $G2$ and $G3$ in a) with one marked as $G2 + 3$ in b). If we succeed in finding the value of the key K, we would have two suitable solutions for the pair $K1$ and $K2$. In that case, the equation (1) does not hold and the difficulty of the attack is reduced.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 March 2024 doi:10.20944/preprints202403.0447.v1

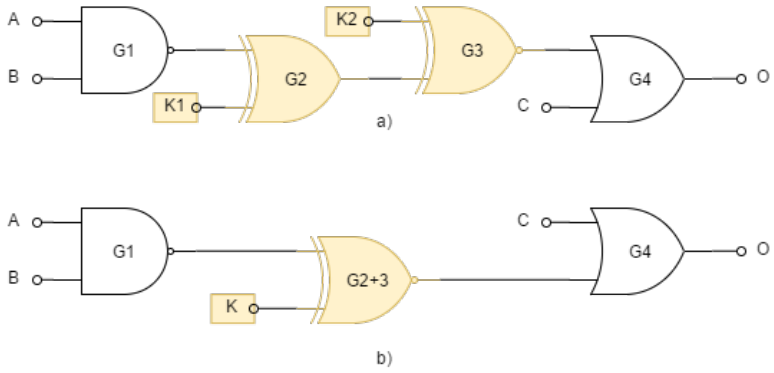

Figure 5. Chained key circuit. Two key gates from (**a**) can be replaced by one from (**b**), thereby increasing the space of valid keys.

There are of course several other techniques for locating key gates. Depending on these techniques, attackers have also developed different approaches to reveal the key bits [4].

3. Description of the Attack

The locked circuit can be described using the relation among the input bits, key bits, and the output bits $C(\vec{X}, \vec{K}, \vec{Y}) \subseteq B^{M+L+N}$, where $B = \{0,1\}$ is the set of Boolean values, vector $\vec{X} \in B^M$ represents M primary inputs of the circuits, vector $\vec{K} \in B^L$ represents the key bits, and vector $\vec{Y} \in B^N$ denotes the N primary output bits. Relation $C(\vec{X}, \vec{K}, \vec{Y})$ is equivalent to the input/output relation of the unlocked circuit if the correct key value $\vec{K}$ is set.

Relation $C(\vec{X}, \vec{K}, \vec{Y})$, i.e. the circuit layout including the key bits, can also be represented by a conjunctive normal form (CNF) with Boolean variables $\vec{X}, \vec{K}, \vec{Y}$. This representation forms the basis of the attack described below.

It is assumed that the attacker has the layout of the circuit (so-called netlist) and has its unlocked version, so he can observe the correct behavior of the circuit for arbitrary inputs. We will model this observation using the function $eval : B^M \rightarrow B^N$.

The goal of the attacker is to find such a key $\vec{K} = \vec{K}_c$, which would satisfy the following equivalence $C(\vec{X}, \vec{K}_c, \vec{Y}) \iff eval(\vec{X}) = \vec{Y}$ for all input patterns $\vec{X}$.

In this work, we consider the type of attack in which the attacker has vectors of input values $\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_p$ and the corresponding vectors of output values $\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_p$. The behavior of the locked circuit, in other words the relationship between known input and output bits and unknown key bits can be described using a Boolean formula of the form $\wedge_{j=1}^{p} C(\vec{X}_j, \vec{K}_c, \vec{Y}_j)$.

To determine the value of the key K_C, it is theoretically sufficient to use a SAT solver to find the key assignment $\vec{K}_c$ such that the formula is satisfiable. The problem is the possible existence of false keys, for which the given formula is also satisfiable. The authors in [5] therefore proposed an algorithm, shown in Algorithm 1, that tries to eliminate false keys in two steps. In the first step, there is a search for so-called distinctive input pattern $\vec{X}^d$, that for two different keys $\vec{K}_1$ and $\vec{K}_2$ outputs two different output patterns $\vec{Y}_1^d$ and $\vec{Y}_2^d$. $\vec{X}^d$ is then the distinctive input pattern if $C(\vec{X}^d, \vec{K}_1, \vec{Y}_1) \wedge C(\vec{X}^d, \vec{K}_2, \vec{Y}_2) \wedge (\vec{Y}_1 \neq \vec{Y}_2)$.

In the second step, the activated circuit is used with the distinctive input pattern $\vec{X}^d$ to obtain the distinctive output pattern $\vec{Y}^d$. Based on this observation, we can exclude one or both keys $\vec{K}_1$ and $\vec{K}_2$ from the class of possible correct keys. The database of distinctive patterns G_i is iteratively built and the SAT solver finally determines the resulting secret key K_C using it.

Algorithm 1 uses two subroutines, $sat(F_i)$ and $sat\_assignment_{\boldsymbol{A}}(F_i)$. The routine $sat(F_i)$ returns True or False depending on whether F_i is a satisfiable or unsatisfiable Boolean formula. The routine $sat\_assignment_{\boldsymbol{A}}(F_i)$ returns an assignment of logical values for the vector $\boldsymbol{A}$ such that the Boolean formula F_i is satisfiable.

 doi:10.20944/preprints202403.0447.v1

Algorithm 1 SAT attack on a locked circuit [5].

Input: C and $eval$
Output: $\boldsymbol{K}_C$
1: $i \leftarrow 1$
2: $G_i \leftarrow True$
3: $F_i \leftarrow C(\boldsymbol{X}, \boldsymbol{K}_1, \boldsymbol{Y}_1) \wedge C(\boldsymbol{X}, \boldsymbol{K}_2, \boldsymbol{Y}_2) \wedge (\boldsymbol{Y}_1 \neq \boldsymbol{Y}_2)$
4: **while** $sat(F_i)$ **do**
5: $\quad \boldsymbol{X}_i^d \leftarrow sat\_assignment_{\boldsymbol{X}}(F_i)$
6: $\quad \boldsymbol{Y}_i^d \leftarrow eval(\boldsymbol{X}_i^d)$
7: $\quad G_{i+1} \leftarrow G_i \wedge C(\boldsymbol{X}_i^d, \boldsymbol{K}, \boldsymbol{Y}_i^d)$
8: $\quad F_{i+1} \leftarrow F_i \wedge C(\boldsymbol{X}_i^d, \boldsymbol{K}_1, \boldsymbol{Y}_i^d) \wedge C(\boldsymbol{X}_i^d, \boldsymbol{K}_2, \boldsymbol{Y}_i^d)$
9: $\quad i \leftarrow i + 1$
10: **end while**
11: $\boldsymbol{K}_C \leftarrow sat\_assignment_{\boldsymbol{K}}(G_i)$

4. Results

We tested the described attack from [5], described in Section 3 on 11 combinatorial circuits from the ISACS '85 class. Basic information about these circuits can be seen in the Table 2). We inserted key gates that represent 5%, 10%, 15%, and 20% of the total number of gates using two approaches: insertion into random wires (rnd lines in the Figures 6 and 7) and insertion into random wires with chaining elimination (nochain lines in the Figures 6 and 7). The exact number of inserted key bits is stated in the Table 2 in the four right-most columns.

Table 2. The ISACS '85 class circuits used for evaluation.

name	# of inputs	# of outputs	# of gates	5%	10%	15%	20%
c17	5	2	6	1	1	1	1
c432	36	7	160	8	16	24	32
c499	41	32	202	10	20	30	40
c880	60	26	383	19	38	57	76
c1355	41	32	546	27	54	82	109
c1908	33	25	880	44	88	132	176
c2670	233	140	1193	63	126	189	252
c3540	50	22	1669	83	167	250	334
c5315	178	123	2307	115	231	346	460
c6288	32	32	2416	121	242	363	484
c7552	207	108	3512	176	351	528	704

The locked circuits were subsequently attacked using the attack in the Algorithm 1. The CaDiCaL SAT solver [21] was used for solving the SAT problem in the Algorithm 1. The choice for the CaDiCaL SAT solver was motivated by the fact that CaDiCaL was the winner of the SAT solver competition in the time of the research. Several other SAT solvers can be found on the international SAT competition webpage [22]. Experiments were performed on an Intel Core i5-5200U CPU with 8GB RAM. The maximal attack running time was set to 1000 seconds.

The final results from the evaluation can be seen in Figures 6, 7 and 8.

We unlocked 10 of the 11 locked circuits with 5% of the key gates as can be seen on blue lines rnd5 and nochain5 in the Figure 6. To use 5% of the key gates can be considered the most realistic scenario. One should realize that it is desired to add a minimum number of gates due to speed, power consumption and cost of the circuit as well.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 March 2024 doi:10.20944/preprints202403.0447.v1

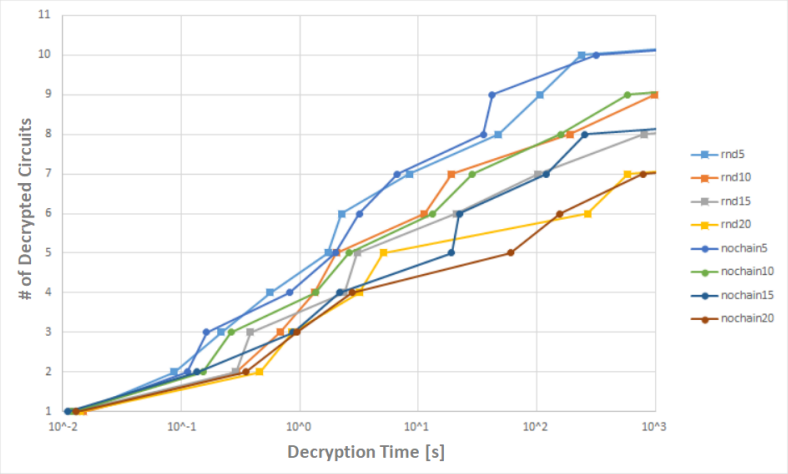

Figure 6. Time versus number of decrypted circuits.

The dependency of the number of unlocked circuits on the number of the decryption rounds (rounds necessary for unlocking), in other words also on the number of input-output pairs $(\vec{X}^d, \vec{Y}^d)$ is shown in the Figure 7. As one can see, 100 pairs were enough to unlock 10 of the 11 locked circuits with 5% of the key gates.

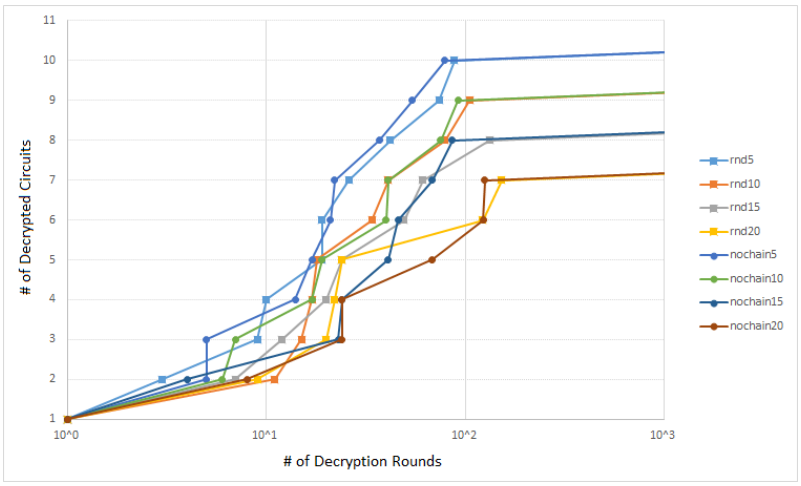

Figure 7. Number of used input-output pairs $(\vec{X}^d, \vec{Y}^d)$versus number of decrypted circuits.

Figure 8 shows the graph of the distribution of the time required to unlock the 11 circuits that were locked using various approaches. The circuits are ordered in the graph according to the total number of gates in the ascending manner. The red mark denotes the median for the given circuit. We assumed that the higher number of gates will be result in the greater attack complexity. However, attacking the circuit $c2670$ was more difficult than the next two bigger circuits. The circuit $c2670$ however has more primary inputs and outputs.

It is also very interesting to take a bit closer look on the number of key bits (key gates). If we insert n key bits into the circuit, it would take $\mathcal{O}(2^n)$ time to try all possible keys, i.e. to use the so-called brute force attack. However, brute force attack is computationally infeasible for enough large values of n. Nowadays secure value is $n \geq 128$. As one can see in the Figure 8, we were able to unlock all the circuits from the $c3540$ with keys of lengths from 83 to 334 bits (see Table 2).

 doi:10.20944/preprints202403.0447.v1

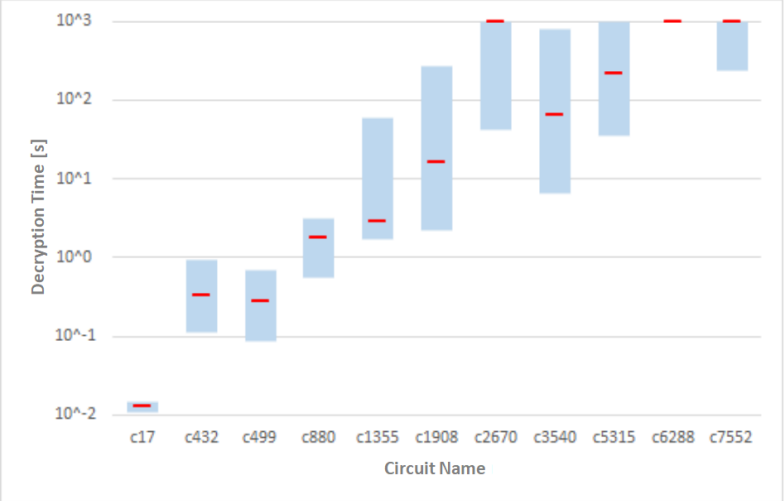

Figure 8. The distribution of time required to attack the individual circuits.

5. Conclusions

To break the logic locking based on the insertion of key bits using XOR/XNOR gates there was an SAT solver-based attack proposed in [5]. The aim of this work was to experimentally verify this attack on several locked circuits with various numbers of primary inputs and outputs and with various numbers of gates. Except for the random placement of the key gates, we evaluate also the positioning with chaining elimination. The obtained and presented results show that we were able to unlock 10 of 11 circuits within a few minutes on a common PC. It follows from the experiments that a higher number of primary inputs and outputs results in more input-output observations needed to unlock the circuit thus resulting in an attack time increase. The results presented in Section 4 show the power of SAT solvers where it was possible to find keys of lengths exceeding the limits suggested nowadays. On the other hand, SAT solvers are struggling with limits concerning the number of clauses for the Boolean formula to be solved as well as the number of used Boolean variables. Thus only logical circuits of limited complexity can be attacked.

An interesting idea for future research is to use the so-called multiple right-hand sides (MRHS) equations, proposed in [17], for attacks on logic locking. MRHS equations were also used in the cryptanalysis of block ciphers [18]. This research way supports also the existence of the MRHS solver [19].

Author Contributions: Conceptualization, F.J. and M.V.; methodology, F.J. and M.V.; software, F.J., J.K., and M.J. ; validation, J.K. and M.J.; formal analysis, F.J. and J.K.; writing—original draft preparation, F.J., J.K., M.V., and V.H.; writing—review and editing, V.H., M.V., and M.J.; visualization, F.J. and J.K.; supervision, M.V. and V.H.; funding acquisition, V.H., M.V., and M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by VEGA grant No. 1/0105/23 and by NATO Science for Peace and Security Programme under Grant G5985.

Acknowledgments: We would like to thank the anonymous reviewers for their comments and suggestions that improved this paper's readability and content.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202403.0447.v1

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
IC	Integrated Circuit
ID	Unique Identifier
FSM	Finite State Machine
CPU	Central Processor Unit
RAM	Random Access Memory
GB	Giga Byte
SAT	Satisfiability

References

1. Suh, G.E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 04–08 June 2007; pp. 9–14. https://doi.org/10.1145/1278480.1278484
2. Koushanfar, F.; Qu, G.; Potkonjak, M. Intellectual Property Metering. *IH 2001, LNCS* **2001**, *2137*, 81–95. https://doi.org/10.1007/3-540-45496-9_7
3. Roy, J.A.; Koushanfar, F.; Markov, I.L. EPIC: Ending Piracy of Integrated Circuits. In Proceedings of the 2008 Design, Automation and Test in Europe, Munich, Germany, 10–14 March 2008; pp. 1069–1074. https://doi.org/10.1109/DATE.2008.4484823
4. Rajendran, J.; Pino, Y.; Sinanoglu, O.; Karri, R. Security Analysis of Logic Obfuscation. In Proceedings of the DAC Design Automation Conference 2012, San Francisco, CA, USA, 03–07 June 2012; pp. 83–89. https://doi.org/10.1145/2228360.2228377
5. Subramanyan, P.; Ray, S.; Malik, S. Evaluating the security of logic encryption algorithms. In Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 05–07 May 2015; pp. 137–143. https://doi.org/10.1109/HST.2015.7140252
6. Koushanfar, F.; Gang, Q. Hardware metering. In Proceedings of the 38th annual design automation conference 2001. pp. 490–493. https://doi.org/10.1145/378239.378568
7. Kajtez, N.; Zhang, Y.; Halak, B. Lockit: A Logic Locking Automation Software. *Electronics* **2021**, *10*, 2817. https://doi.org/10.3390/electronics10222817
8. Mirmohammadi, Z.; Borujeni, S.E. A New Optimal Method for the Secure Design of Combinational Circuits against Hardware Trojans Using Interference Logic Locking. *Electronics* **2023**, *12*, 1107. https://doi.org/10.3390/electronics12051107
9. Nguyen, Q.-L.; Dupuis, S.; Flottes, M.-L.; Rouzeyre, B. SKG-Lock+: A Provably Secure Logic Locking Scheme-Creating Significant Output Corruption. *Electronics* **2022**, *11*, 3906. https://doi.org/10.3390/electronics11233906
10. Baumgarten, A.; Tyagi, A.; Zambreno, J. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design & Test of Computers* **2010**, *27*, 66–75. https://doi.org/10.1109/MDT.2010.24
11. Divyanshu, D.; Kumar, R.; Khan, D.; Amara, S.; Massoud, Y. An Approach towards Designing Logic Locking Using Shape-Perpendicular Magnetic Anisotropy-Double Layer MTJ. *Electronics* **2023**, *12*, 479. https://doi.org/10.3390/electronics12030479
12. Divyanshu, D.; Kumar, R.; Khan, D.; Amara, S.; Massoud, Y. Design of VGSOT-MTJ-Based Logic Locking for High-Speed Digital Circuits. *Electronics* **2022**, *11*, 3537. https://doi.org/10.3390/electronics11213537
13. Alasad, Q.; Yuan, J.-S.; Bi, Y. Logic Locking Using Hybrid CMOS and Emerging SiNW FETs. *Electronics* **2017**, *6*, 69. https://doi.org/10.3390/electronics6030069
14. Yasin, M.; Rajendran, J.; Sinanoglu, O. A Brief History of Logic Locking. In *Trustworthy Hardware Design: Combinational Logic Locking Techniques, Analog Circuits and Signal Processing*; Springer International Publishing: Cham, Switzerland, 2020; pp. 17–31 http://dx.doi.org/10.1007/978-3-030-15334-2_2
15. Zhou, J; Zhang, X. Generalized SAT-attack-resistant logic locking. *IEEE Transactions on Information Forensics and Security* **2021**, *16*, 2581–2592. https://doi.org/10.1109/TIFS.2021.3059271

 doi:10.20944/preprints202403.0447.v1

16. Zhou, J.; Zhang, X. A Low-Complexity Flexible Logic-Locking Scheme Resisting Removal Attacks. In IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 09–11 August 2021, pp. 869–873. https://doi.org/10.1109/MWSCAS47672.2021.9531796
17. Zajac, P. MRHS equation systems that can be solved in polynomial time. *Tatra Mountains Mathematical Publications* **2016**, *67*, 205–219. https://doi.org/10.2478/tatra.v67i0.449
18. Zajac, P. Algebraic Cryptanalysis with MRHS Equations *Cryptography* **2023**, *7*, 19. https://doi.org/10.3390/cryptography7020019
19. Raddum, H.; Zajac, P. MRHS solver based on linear algebra and exhaustive search *Journal of Mathematical Cryptology* **2018**, *12*, pp. 143–157. https://doi.org/10.1515/jmc-2017-0005
20. Zhong, Y.; Guin, U. Complexity Analysis of the SAT Attack on Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2023**, *42*, pp. 3143–3156. https://doi.org/10.1109/TCAD.2023.3240933
21. CaDiCaL. Simplified Satisfiability Solver. Available online: https://fmv.jku.at/cadical/ (accessed on March 3, 2024).
22. The International SAT Competition Web Page. Available online: http://www.satcompetition.org/ (accessed on March 3, 2024)