On Preventing SAT Attack with Decoy Key-Inputs

Quang-Linh Nguyen, Marie-Lise Flottes, Sophie Dupuis, Bruno Rouzeyre

On Preventing SAT Attack with Decoy Key-Inputs

Quang-Linh Nguyen, Marie-Lise Flottes, Sophie Dupuis, and Bruno Rouzeyre
LIRMM, Université de Montpellier, CNRS
Montpellier, France
firstname.lastname@lirmm.fr

***Abstract*—The globalized supply chain in the semiconductor industry raises several security concerns such as IC overproduction, intellectual property piracy and design tampering. Logic locking has emerged as a Design-for-Trust countermeasure to address these issues. Original logic locking proposals provide a high degree of output corruption – i.e., errors on circuit outputs – unless it is unlocked with the correct key. This is a prerequisite for making a manufactured circuit unusable without the designer's intervention. Since the introduction of SAT-based attacks – highly efficient attacks for retrieving the correct key from an oracle and the corresponding locked design – resulting design-based countermeasures have compromised output corruption for the benefit of better resilience against such attacks. Our proposed logic locking scheme, referred to as SKG-Lock, aims to thwart SAT-based attacks while maintaining significant output corruption. The proposed provable SAT-resilience scheme is based on the novel concept of decoy key-inputs. Compared with recent related works, SKG-Lock provides higher output corruption, while having high resistance to evaluated attacks.**

***Index Terms*—Logic Locking, SAT Attack, Design-for-Trust, Hardware Security, IP Protection, Overproduction**

I. Introduction

The outsourcing business model currently dominates the semiconductor industry. As manufacturing costs have become prohibitive, outsourcing the fabrication process to offshore foundries has become a major trend. This leads to increasing exposure of hardware design Intellectual Property (IP) to external and possibly unreliable actors. Besides, hardware reverse-engineering techniques have become more advanced. Due to the loss of control over IP usage and the increasingly advanced adversaries in the supply chain, several threats, such as Integrated Circuit (IC) overproduction, counterfeiting, IP piracy and Hardware Trojan insertion, have become major sources of cybersecurity concern [1], [2].

Numerous recent Design-for-Trust approaches introduce preventive mechanisms at design time [3]. Among them, one of the most studied and most versatile approaches is logic locking [4]. *Logic locking* inserts logic controlled by additional key-inputs into a design, in order to lock it with a secret key [5]. Unless this secret key value is applied to the key-inputs, the circuit remains malfunctioning. Therefore, in following stages of the IC production flow, untrusted entities, without any knowledge about the key, are hindered from maliciously using the logic-locked IP. After production, the circuits are unlocked, by the design house or a trusted partner, before being released into the market.

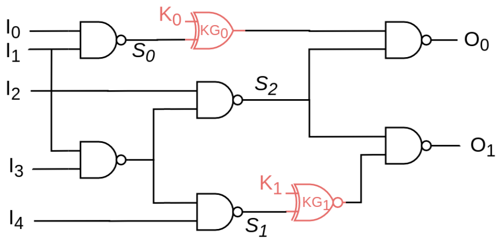


Fig. 1: Key-gate insertion based logic locking ($K_0K_1 = 01$).

Primitive logic locking techniques introduced the concept of inserting key-controlled gates, referred to as *key-gates* [6]–[9]. An example of XOR/XNOR key-gate insertion is shown in Fig. 1. However, such logic locking techniques are highly susceptible to the SAT attack [10], the most effective and the most studied attack on logic locking. This attack uses a Boolean satisfiability (SAT) solver, the locked netlist and an unlocked circuit to efficiently prune out wrong key values.

Provably secure logic locking against the SAT attack is based on exponentially increasing its computing time [11]–[14]. However, regarding this type of techniques, a fundamental trade-off has been identified between SAT resilience and output corruption [15]. Indeed, the output corruption is so limited that the circuit is mostly functional despite being supplied with a wrong key value [16].

In this paper, we present a novel logic locking technique, namely SKG-Lock, based on so-called switchable key-gates and additional decoy key-inputs. We theoretically validate that SKG-Lock achieves maximum resilience against the SAT attack, regardless of the number of switchable key-gates and their corruptibility. We also provide evaluations of attack resilience, output corruption and overhead of SKG-Lock, along with comparisons with state-of-the-art techniques.

The rest of the paper is organized as follows. Section II provides background on logic locking and output corruption, then SAT-based attacks and existing countermeasures. Section III presents our proposed logic locking scheme, SKG-Lock, along with security analysis against the SAT attack. Experimental results for the evaluations of security, output corruption and overhead are shown in Section VI. Finally, Section V concludes the paper.

II. Background

A. Output Corruption Metrics for Logic Locking

A logic locking technique should ensure that the locked circuit behaves erroneously upon the application of a wrong key value. Thus, output corruption is an important criteria for logic locking. It is measured, not only by the frequency of observed corruption, but also by the breath of circuit outputs that are corrupted. Let us define two metrics of output corruption, corruption rate and corruption coverage.

Definition 1 (Corruptibility): Corruptibility[1]of a key-gate (or a key-controlled structure) is the probability of corruption at its insertion signal if a wrong key value is supplied.

For example, XOR/XNOR key-gates (cf. KG_0 and KG_1 in Fig. 1) have a 100% corruptibility whereas a point-function lock (cf. Fig. 2) has a corruptibility of $1/2^n$.

Definition 2 (Output corruption rate): Output corruption rate presents the probability of observing erroneous values on the outputs of a locked circuit. It is measured by the percentage of input patterns that lead to errors at circuit outputs when any incorrect key value is applied.

Referring back to the example in Fig. 1, the output corruption rate is 62.5%; even though each key-gate has high corruptibility, the corruption is masked if the signal S_2 is 0.

Note that a high output corruption rate may not be sufficient for preventing the usage of locked circuits. For instance, in the image processing domain, even if the least significant bit of the output is always wrong (100% output corruption rate), the circuit can still be useful.

Definition 3 (Output corruption coverage): Output corruption coverage presents the magnitude of corruption propagated to circuit outputs. It is characterized by the percentage of the circuit outputs that can be affected by all corrupted signals.

For example in Fig. 1, O_0 and O_1 are in the fanout of KG_0 and KG_1 and can be impacted by corruption at S_0 and S_1; thus, the output corruption coverage is 100%.

B. SAT Attack

The SAT attack [10] is an *oracle-guided attack*, with the following threat model. The attacker has the access to two fundamental assets: (i) the *locked netlist*, i.e., the netlist containing previously inserted logic locking structure, (ii) an *oracle*, i.e., an unlocked IC with accessible scan chains.

The SAT attack is an iterative process. In each iteration, the attack chooses two key values from two key equivalence classes and finds a so-called Distinguish Input Pattern (DIP) that results in different output values for the two key values. The detected DIP is then applied to the oracle to obtain the golden output. By adding the observed disagreement between the locked netlist and the oracle as constraints to the SAT solver, the attack is able to suppress at least one class of key values, which may contain multiple values. The key search space is reduced iteratively until no more DIP can be identified. Finally, the SAT solver deduces the correct key.

C. Provable SAT Countermeasures

The most notable countermeasures against SAT attack are SARLock [11] and Anti-SAT [12], [14]. Their SAT resilience is based on maximizing the number of iterations of the attack.

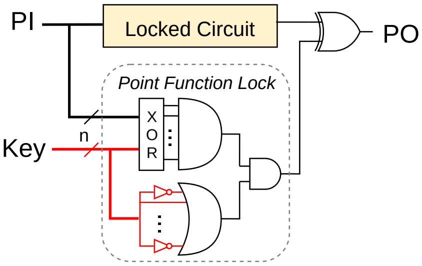


Fig. 2: Point-function based logic locking.

The general structure of these techniques is depicted in Fig. 2: a structure based on a point-function – a Boolean function that outputs the value 1 for exactly one input pattern – is inserted beside the locked circuit. It contains two comparators connected in parallel. The point-function lock only corrupts the circuit output for one corresponding key value per input pattern. The SAT attack picks such a pattern as a DIP and is able to rule out only one key value. This leads to a maximum number of iterations 2^n, n being the number of circuit inputs connected to the point-function lock; and the key size is n for SARLock or $2n$ for Anti-SAT.

Definition 4 (SAT resilience level): SAT resilience level of a logic locking technique is n-secure if the number of iterations returned by the SAT attack on its locked circuits is 2^n [13].

Note that the corruptibility of point-function lock is $1/2^n$. Due to a low number of interconnections with the locked circuit, its output corruption coverage is inherently insufficient.

D. Advanced Defenses and Attacks

A number of secure logic locking techniques has been recently proposed. Stripped Functionality Logic Locking (SFLL) [13] provides a solution for measurable trade-off between attack resilience and output corruption. In SFLL-HD^h, the parameter h, the Hamming distance between the input and the key, can be configured so that the output corruption rate is increased, however, at the cost of decreasing SAT-attack resilience. More recently, CAS-Lock [18] was proposed to avoid conforming to the security-corruptibility trade-off. By improving the Anti-SAT structure, it is able to achieve high SAT resilience with sufficient corruption rate.

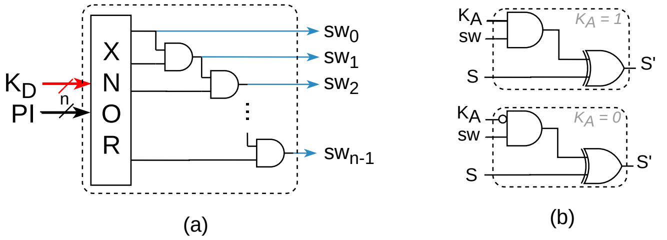
Several variants of the SAT attack have also been proposed [19]. The AppSAT attack [17] is an *approximate* version of the SAT attack. Instead of finding the exact secret key, AppSAT returns a key value that renders the circuit "approximately" functional. Compared to the baseline SAT attack, the AppSAT attack has additional capabilities of error estimation and random query reinforcement. Therefore, it can avoid being trapped into solving the key for the point-function lock. Hence, the resulting key is expected to reduce the output corruption rate as low as that of a point-function lock.

III. Proposed Logic Locking Scheme

A. Basic Components

The two fundamental components of SKG-Lock are Switchable Key-Gates (SKGs) and a Switch Controller (SWC), as

[1] It is also referred to as error rate in the literature [13], [17].



(a) (b)

Fig. 3: Basic structure of SKG-Lock components. (a) Switch controller. (b) Switchable key-gates.

shown in Fig. 3.

We introduce two sets of key-inputs, the *Activation Key* (K_A) and the *Decoy Key* (K_D):

- K_A is connected to the SKGs. Inserting the correct value of K_A nullifies all SKGs and unlocks the circuit. This K_A value is set by the designer during the design phase.
- K_D is connected to the SWC. It sets the input patterns that trigger the SKGs (through the switch-signals, denoted as sw in Fig. 3). K_D is not involved in the circuit unlocking, i.e., the circuit can be unlocked irrespective of the K_D value.

Note that K_D is treated as a key-input in the same way as K_A. They both come from a protected memory and are physically indistinguishable. They are controllable inputs in the locked netlist used in oracle-guided attacks.

An SKG has three inputs, two control signals — K_A and switch-signal sw — and one signal *S* from the locked circuit (cf. Fig. 3.b). Hence, compared to a traditional XOR key-gate, in an SKG, the additional *switch-signal* enables the control of its corruptibility. To make an SKG corrupt the signal *S*, both of its control signals have to be asserted. Therefore, corruption only happens when an incorrect value is inserted at the K_A key-input and 1 is set on the switch-signal (in case of SKGs with positive switch[2]).

The SWC determines the activity of SKGs by controlling their switch-signals. Its inputs are K_D and an equal number of circuit inputs[3]. A general design for an SWC is a comparator, constructed with a row of XNOR gates and a cascade of AND gates (cf. Fig. 3.a). Multiple switch-signals can be outputted from the SWC, one from each node in the AND cascade. The output at the end of the cascade sw_{n-1} is essentially the output of a point-function between K_D and PI. Therefore, sw_{n-1} presents low corruptibility ($1/2^n$) but maximal complexity for the SAT attack, as shown in the following section. Other switch-signals from the upstream of the cascade have higher activity. Thus, SKGs driven by them have higher corruptibility.

B. Light-weight SAT-Proof Lock

We present in Fig. 4 a lightweight point-function lock based on basic components of SKG-Lock, referred to as SKG-Lock$^{\text{lw}}$. It contains an SWC and an SKG. The only

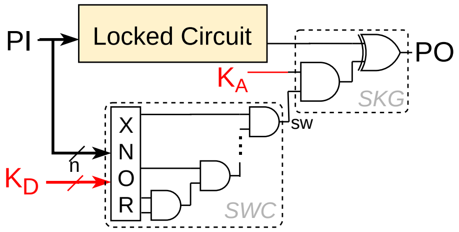


Fig. 4: Light-weight SAT-proof lock constructed by an SWC and an SKG.

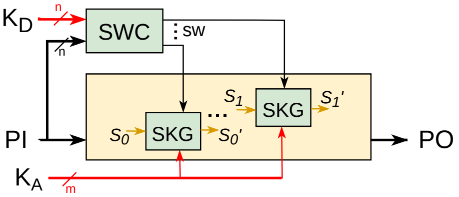


Fig. 5: General architecture of SKG-Lock.

switch-signal sw is the output of the point-function in the SWC. We show subsequently that this structure achieves an n-secure SAT resilience level.

Notations & Assumptions: Without loss of generality, we can assume that the size of K_D and circuit PI is n; the correct value of each bit of K_A is 1 for every SKG; and each SKG is inserted at a different circuit output so that any corruption is observed at an output. A DIP produced at *i*-th iteration by the SAT attack is denoted as X_i. The estimated number of iterations of SAT attack is N.

Proof for SAT resilience: Wrong key values that can be ruled out by a DIP X_i satisfy the following condition:

$$(K_A = 0) \wedge (\vec{K_D} = \vec{X_i}) \qquad (1)$$

For any given X_i, there is one way to select K_A and one way to select K_D to satisfy the condition in (1). Thus, each iteration identifies only one wrong key value. Hence, the number of iterations required by the SAT attack to eliminate all (2^n) wrong key values is $N = 2^n$. The circuit is n-secure against SAT attack.

SKG-Lock$^{\text{lw}}$ achieves the same SAT resilience as SAR-Lock and Anti-SAT. Moreover, it only requires half of the hardware as it contains one comparator instead of two in other techniques. Using SKG-Lock$^{\text{lw}}$ standalone, however, is not practical: as K_A is only 1 bit, the ratio of correct keys over key space is $1/2$; the low output corruptibility ($1/2^n$) issue remains; its structure is isolated from the locked circuit structure.

C. SKG-Lock Architecture

The general version of SKG-Lock comprises of several SKGs with different corruptibility. Its architecture is illustrated in Fig. 5. m SKGs (hence the size of K_A) are inserted in the locked circuit. An SWC, with n circuit inputs and n-bit K_D

[2] SKGs with negative switch can be constructed with an OR gate and an XNOR gate.

[3] Circuit inputs can include primary inputs and pseudo primary inputs (outputs of flip-flops in a sequential circuit).

as its inputs, produces n switch-signals used for controlling the inserted SKGs. Note that, in the case where $n \neq m$, several SKGs may be driven by the same switch-signal or certain switch-signals may be unconnected. The ratio of correct keys over key space of SKG-Lock is $1/2^m$.

Any existing key-gate insertion strategy for logic locking [6]–[9] can be used for SKG insertion in SKG-Lock. Fault-based strategy [8] selects signals with the highest fault impact in the circuit so that key-gates inserted at chosen signals have high output corruption rate and coverage. For this proposal, we therefore choose to insert SKGs using the fault-based strategy, to maximize output corruption for SKG-Lock.

Note that the use of several switch-signals and SKGs creates multiple connections between the SWC and the locked circuit. Thus, our SKG-Lock architecture, in its nature, achieves structural entanglement without requiring any compound structural obfuscation technique, such as wire entanglement used in Anti-SAT [14].

Proof for SAT resilience: We consider the case where the circuit is locked with two SKGs: one SKG is driven by sw_{n-1} and another SKG is driven by sw_{n-2} (cf. Fig. 3).

The condition for any wrong key value identified by a given X_i is:

$$\begin{aligned}&\big[(\vec{K_A}[0] = 0, \vec{K_A}[1] \in \mathbb{B}) \wedge (\vec{K_D} = \vec{X_i})\big] \vee \\ &\big[(\vec{K_A}[0:1] = \vec{10}) \wedge (\vec{K_D}[0:n-2] = \vec{X_i}[0:n-2])\big] \quad (2)\end{aligned}$$

Thus, when $\vec{K_A}[0] = 0$, the set of wrong key eliminated by X_i has the following form:

$$(\vec{K_A}[0] = 0, \vec{K_A}[1] \in \mathbb{B}, \vec{K_D} = \vec{X_i}) \quad (3)$$

There is a one-to-one matching between K_D and X_i. Thus, any input pattern can be selected as a DIP to identify a unique set of wrong keys in the form of (3). Therefore, the total number of SAT iterations is $N = 2^n$. The circuit is n-secure against SAT attack.

This proof holds for the case where there are more than two SKGs. The same SAT-resilience level is achieved as long as there is at least one SKG connected with sw_{n-1}.

IV. Results

We implemented SKG-Lock on ISCAS'85 and MCNC benchmarks. We set in each benchmark an equal size of K_A and K_D, $m = n$ (hence the total key size is $2n$). The n circuit inputs connected to the SWC were randomly selected. n SKGs were inserted using fault-based strategy and n switch-signals were used, each of which was driving each SKG. The experiments were executed on an 8-core Intel processor running at 1.90GHz with 16 GB RAM. ModelSim was used for simulation and measuring output corruption. To evaluate output corruption rate, we calculated the percent of applied patterns that produced errors at circuit outputs. Output corruption coverage was calculated as the percent of accumulated number of corrupted circuit outputs throughout the simulation. Synopsys Design Compiler, with a 65nm technology library, was used for estimating overheads.

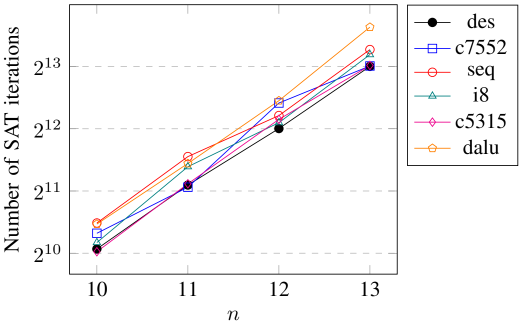


Fig. 6: Evaluation of SAT resilience vs. key size n.

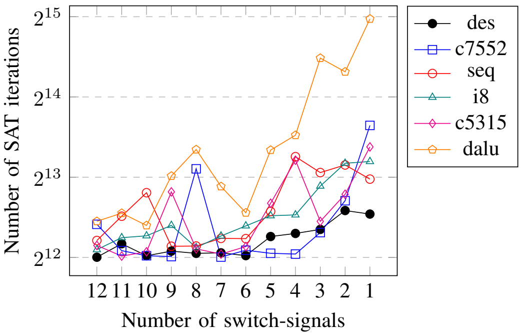


Fig. 7: Evaluation of SAT resilience vs. output corruption ($n = 12$). The fewer switch-signals, the lower output corruption (only 1 switch-signal is equivalent to a point-function).

A. Security Evaluation

We first show the evaluation of SKG-Lock against the SAT attack [10] (using the provided tool) to validate the proof in Section III-C.

The evaluation of SAT resilience of SKG-Lock with increasing key size[4](by increasing n) is shown in Fig. 6. The expected number of iterations for each case is 2^n in order to be n-secure against the SAT attack. The observed numbers of SAT iterations[5]are bigger than the expected numbers.

We further investigated the relation between SAT resilience and output corruption. To create lower output-corruption configurations of SKG-Lock, high-corruptibility switch-signals can be left unused, and low-corruptibility ones can be connected to several SKGs. Thus, we restricted the number of switch-signals and the selection was made according to the decreasing corruptibility order. The number of used switch-signals ranges from n down to 1, where n stands for the base configuration (all switch-signals are used and each is connected to an SKG) and 1 stands for the point-function configuration (only sw_{n-1} is used for all SKGs).

We report in Fig. 7 the result of circuits with $n = 12$. We expect 12-secure against SAT in every case. Once again, each

[4] To better observe the trends, we experiment with small key sizes.

[5] The SAT computation time is proportional to the number of iterations.

TABLE I: COMPARISON OF SAT RESILIENCE LEVEL

Techniques	SARLock	Anti-SAT	SFLL-HD^h	CAS-Lock	SKG-Lock^{lw}	SKG-Lock
SAT resilience level	n	n	$n - \lceil log_2 \binom{n}{h} \rceil$	n	n	$\geq n$ †

† Higher level can be achieved with lower-corruption configurations.

obtained number of SAT iterations is higher than expected. A remarkable point is that lower output-corruption configurations are inclined to have higher gain in iterations. Several of the evaluated circuits achieve 13-secure up to 15-secure.

The cause of the extra iterations (up to several times the expected number) could stem from the locations of inserted SKGs. From the proofs in Section III, the bits in a DIP that differentiate it from another belong to the part of circuit inputs connected to the SWC. However, propagating SKGs corruption to circuit outputs involves controlling several circuit inputs, not only the ones connected to the SWC. Moreover, with a restricted number of switch-signals, more SKGs have the same corruptibility such that there could be masking or interference among corruptions from SKGs. Thus, inputs that are not connected to the SWC may also be taken into account (in addition to the connected ones) when the attack identifies DIPs. Hence, there is a significant gain in SAT iterations.

The comparison of security level against SAT attack of SKG-Lock with other SAT-resilient techniques is presented in Table I (n is the number of circuit inputs connected to the key-controlled block). Whereas SARLock, Anti-SAT and LW-SKG are n-secure but with exponentially reduced corruptibility of $1/2^n$, SFLL-HD enables trading resilience for better corruptibility ($\binom{n}{h}/2^n$). SKG-Lock achieves n-secure level while having significant output corruption, as shown in the following Section.

We also evaluated SKG-Lock against the AppSAT attack (using the tool from [20]). We used the same attack configuration as in [17]: 50 random queries were applied after every 12 SAT iterations and the settlement threshold was 5. We applied AppSAT on SKG-Lock with $n = 32$ and ran the attack ten times on each benchmark. The accuracy of the approximate keys found by the attack — directly related to the output corruption generated by this key — is of interest.

The results are shown in Table II. With the benchmark *seq*, AppSAT cannot converge and, hence, did not return any result in ten runs. With other benchmarks, AppSAT was able to converge; however, it always produced an inaccurate key. The K_A part of the obtained keys are different from our predefined K_A values for several bits (the average number of different bits of 10 key values is reported in the column "Accuracy #DiffBits"). We measured the output corruption with the produced keys: for each key value, we applied 1,000,000 random input patterns and compared the outputs observed from the locked circuit to the golden outputs. Columns "#Patterns" (the number of input patterns that produce corrupted outputs), "Rate" (output corruption rate) and "Coverage" (output corruption coverage) in Table II show the average results of 10 obtained key values. It is apparent that AppSAT failed to reduce the output corruption of SKG-Lock to a point-function corruptibility ($1/2^{32}$ in this case). The observed corruption rates range from 0.1% to 4%, indicating that the circuit, if run at 1MHz, may produce several thousands of errors each second. Moreover, we also observed sufficient output corruption coverage on several benchmarks. This result is indeed in line with the result in [21] concluding that the AppSAT attack is not effective against logic locking techniques where different key values correspond to different amounts of output corruption.

TABLE II: APPSAT ATTACK EVALUATION ($n = 32$)

Bench	Converged	Accuracy #DiffBits /32	Output Corruption		
			#Patterns /1000000	Rate (%)	Coverage (%)
des	Yes	2.5	1527	0.153	4.8
c7552	Yes	4.3	7274	0.73	18.6
seq	No	-	-	-	-
i8	Yes	5.9	1025	0.103	71.5
c5315	Yes	5	38683	3.87	28.13
dalu	Yes	7.5	9281	0.93	100

TABLE III: OUTPUT CORRUPTION EVALUATION ($n = 64$)

Bench	Corruption Rate (%)		Corruption Coverage (%)	
	SKG-Lock	CAS-Lock	SKG-Lock	CAS-Lock
des	49.4	23.47	100	0.8
c7552	49.6	8.79	58.88	0.93
c5315	23.9	12.46	78.05	1.63
i8	11.6	8.89	100	1.235
dalu	31	3.12	100	25
Average	**33.1**	11.35	**87.39**	5.59

B. Output Corruption Evaluation

We simulated each circuit 1,000,000 times, each time with a random input pattern and a random wrong key value. We measured and compared the output corruption of SKG-Lock and CAS-Lock with the SAT resilience level of 64-secure. For CAS-Lock, the CAS-Lock block is inserted at a high-controllability signal in the circuit; the inserted block contains a cascade of AND gates followed by an OR gate, which allows highest corruptibility among all configurations of CAS-Lock.

Table III presents these results. As can be seen, SKG-Lock achieves better results than CAS-Lock in both metrics. One can notice that, due to the scattering of SKGs throughout the circuit, SKG-Lock is able to affect all circuit outputs in several cases. Conversely, CAS-Lock only corrupts one signal in the circuit, thereby affecting only a few outputs.

In comparison, for the same SAT resilience, SARLock, Anti-SAT and SFLL-HD^0 have a corruption rate of $1/2^{64} = 5.4e^{-18}\%$.

C. Overhead Evaluation

The overhead of SKG-Lock was evaluated in terms of area, power and delay overheads. We used the same benchmarks as in Section IV-B. We also provide comparison with the overheads of SARLock, Anti-SAT, CAS-Lock and SFLL-HD $h = 0$ (with the same SAT resilience level).

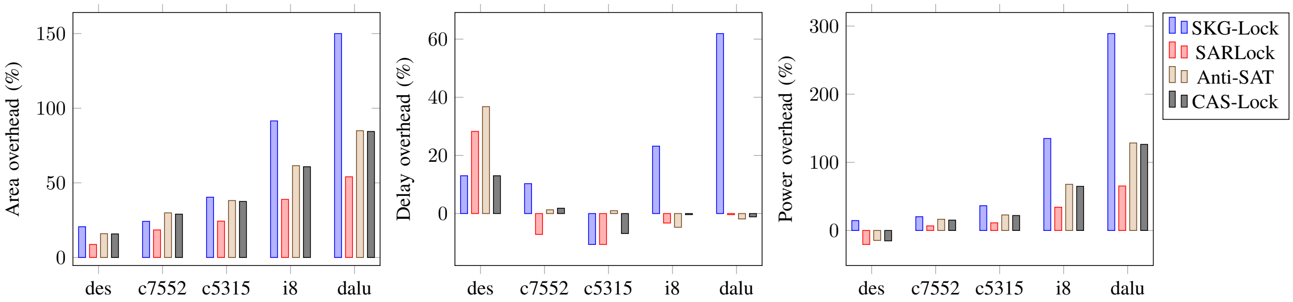


Fig. 8: Overhead evaluation and comparison with related works ($n = 64$). Benchmarks are in decreasing size order.

These results are presented in Fig. 8. SKG-Lock has higher overheads than SARLock, Anti-SAT and CAS-Lock in some cases but the differences are acceptably small. The results of SFLL-HD are not reported since they are extremely higher than others (e.g. for the benchmark $dalu$, 574.5% in area, 341.4% in delay and 4552.6% in power); its high overheads are due to added comparators and Hamming Distance counters.

In fact, for SKG-Lock, the numbers of additional gates in each benchmark are quite similar: 350 (des), 281 ($c7552$), 344 ($c5315$), 408 ($i8$) and 441 ($dalu$). For benchmark des, the additional gate count scales rather linearly with the key size n: 230 ($n = 32$), 312 (48), 350 (64) and 412 (80). Therefore, the area overhead of SKG-Lock is dependent of the key size rather than of the benchmark size.

V. Conclusion

In this paper, we proposed a novel SAT-resilient logic locking scheme, SKG-Lock. SKG-Lock architecture is based on switchable key-gates and a switch controller controlled by decoy key-inputs.

Thanks to the proposed control of SKGs, our solutions, both SKG-Lock$^{\text{lw}}$ and SKG-Lock, are provably secure against the SAT attack. Improved from SKG-Lock$^{\text{lw}}$ by taking advantage of multiple SKGs, the proposed SKG-Lock provides tremendously higher output corruption, better structural entanglement and better resilience against SAT-based attacks. Compared to state-of-the-art works, SKG-Lock provides significant output corruption and high attack resilience, while incurring acceptable overhead.

Because the proposed architecture enables structural entanglement between the inserted logic and the locked circuit, we expect substantial resistance against structure analysis attacks [14], [22]. The SWC, whose structure is similar to point-function, can be obfuscated by inserting key-gates to make it less detectable by signal probability analysis. This evaluation will be a part of the future development for SKG-Lock.

Acknowledgment

This work is funded by project MOOSIC ANR-18-CE39-0005 of the French National Research Agency (ANR).

References

[1] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, “Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain,” *Proceedings of the IEEE*, vol. 102, pp. 1207–1228, Aug. 2014.

[2] S. Dupuis, M. Flottes, G. D. Natale, and B. Rouzeyre, “Protection Against Hardware Trojans With Logic Testing: Proposed Solutions and Challenges Ahead,” *IEEE Design Test*, vol. 35, pp. 73–90, Apr. 2018.

[3] J. Rajendran, O. Sinanoglu, and R. Karri, “Regaining Trust in VLSI Design: Design-for-Trust Techniques,” *Proceedings of the IEEE*, vol. 102, pp. 1266–1282, Aug. 2014.

[4] M. Yasin, J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Analog Circuits and Signal Processing, Cham: Springer International Publishing, 2020.

[5] S. Dupuis and M.-L. Flottes, “Logic Locking: A Survey of Proposed Methods and Evaluation Metrics,” *J Electron Test*, vol. 35, pp. 273–291, June 2019.

[6] J. A. Roy, F. Koushanfar, and I. L. Markov, “EPIC: Ending piracy of integrated circuits,” in *DATE*, pp. 1069–1074, Mar. 2008.

[7] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre, “A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans,” in *IOLTS*, pp. 49–54, July 2014.

[8] 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, pp. 410–424, Feb. 2015.

[9] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, “On Improving the Security of Logic Locking,” *IEEE TCAD*, vol. 35, pp. 1411–1424, Sept. 2016.

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

[11] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, “SARLock: SAT attack resistant logic locking,” in *HOST*, pp. 236–241, May 2016.

[12] Y. Xie and A. Srivastava, “Mitigating SAT Attack on Logic Locking,” in *CHES*, vol. 9813, pp. 127–146, 2016.

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

[14] Y. Xie and A. Srivastava, “Anti-SAT: Mitigating SAT Attack on Logic Locking,” *IEEE TCAD*, vol. 38, pp. 199–207, Feb. 2019.

[15] H. Zhou, A. Rezaei, and Y. Shen, “Resolving the Trilemma in Logic Encryption,” in *ICCAD*, pp. 1–8, Nov. 2019.

[16] M. Zuzak, Y. Liu, and A. Srivastava, “Trace Logic Locking: Improving the Parametric Space of Logic Locking,” *IEEE TCAD*, 2020.

[17] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, “AppSAT: Approximately deobfuscating integrated circuits,” in *HOST*, pp. 95–100, May 2017.

[18] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, “CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme,” *IACR TCHES*, pp. 175–202, 2020.

[19] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak, “Keynote: A Disquisition on Logic Locking,” *IEEE TCAD*, 2019.

[20] K. Shamsi, “Netlist encryption and obfuscation suite.” https://bitbucket.org/kavehshm/neos/src/master/.

[21] Y. Shen, A. Rezaei, and H. Zhou, “A comparative investigation of approximate attacks on logic encryptions,” in *ASP-DAC*, pp. 271–276, Jan. 2018.

[22] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, “Removal Attacks on Logic Locking and Camouflaging Techniques,” *IEEE TETC*, 2017.