Logic Locking: Exploration of a new key-gate based on tristate logic

Sophie Dupuis, Nassim Riadi, Clémy Moroukian, Florence Azaïs, Marie-Lise Flottes

Logic Locking: Exploration of a new key-gate based on tristate logic

Sophie Dupuis, Nassim Riadi, Clémy Moroukian, Florence Azaïs and Marie-Lise Flottes
LIRMM, University of. Montpellier/CNRS, Montpellier, France
firstname.lastname@lirmm.fr

***Abstract*—As a consequence of the tremendous rapid pace of semiconductor technology evolution over the last decades, most semiconductor companies have become fabless and outsource the manufacturing step of their designs to offshore foundries. Furthermore, most integration companies rely on Intellectual Property (IP) modules purchased from third parties. Consequently, new threats have arisen such as reverse engineering, IP theft, Integrated Circuits (ICs) overproduction, hardware Trojan insertion, to name just a few.**

Taking these trust issues into account at design time has therefore become mandatory for protecting IPs despite this new ecosystem. Logic locking is one of these so-called Design-for-Trust approaches. It involves inserting extra key-gates controlled by a secret key in a design so that the manufactured ICs behave correctly only when the correct key value is provided, and erroneously otherwise.

This paper introduces a novel key-gate based on tristate logic. The primary goal here is to demonstrate its capability to provide optimal output corruption on circuit outputs when the circuit is not controlled by the correct key value. The paper also provides discussion on protection against the most common attacks developed on current logic locking solutions.

Keywords— Hardware trust, Design-for-Trust, Logic locking, Tristate logic, Output corruption, SAT attack.

I. Introduction

Due to development costs and time-to-market reasons, the semiconductor business has shifted over decades from a vertical to a horizontal model, where many different entities are now involved in the design, integration, manufacture, test, packaging and distribution of chips. This new business model has led to safety issues and numerous frauds have emerged, including IP piracy/theft, potentially perpetrated by several (untrustworthy) entities in the IP life cycle: i. a foundry, which can cheaply overproduce ICs, ii. a system integrator, who buys an IP for a specific design but can reuse this IP for another design, iii. an end-user, who can use reverse engineering to gain knowledge about design and reuse this information for another device.

To counter these threats, multiple Design-for-Trust approaches enabling IP/IC protection have been developed over the past fifteen years and the logic locking approach has emerged as a prominent solution for protecting IP against piracy, counterfeiting and overproduction [1]. Logic locking involves inserting extra locking circuitry into the original design and extra key inputs to control this circuitry. In order to fulfill its original function, the circuit must be programmed with the correct key by the designer after production; the circuit is said activated or unlocked. Any chip programmed with a different key is said locked; it is rendered inoperative.

Initially proposed in 2008 to counteract IC overproduction, logic locking similarly protects against IP overuse. Furthermore, any attempt to reverse-engineer an IC/IP is rendered useless since retrieving the netlist becomes useless without knowledge of the correct key value. For the same reasons, attempting to insert a hardware Trojan becomes all the more complicated as the functionality of the original design is obfuscated by the locking logic.

Logic locking has attracted considerable attention from the research community and has been forced to continually evolve ever since. The most effective attacks for recovering the designer-defined secret key, and most studied and countered ones in recent years are based on SAT solvers [1]. The paradigm shift induced by these attacks has led to the emergence of new locking schemes, often at the expense of poor output corruption and enabling the development of new attacks such as approximate attacks - taking advantage of low output corruption of locked circuits - or removal attacks - taking advantage of the locking scheme implementation.

The present work aims to thwart SAT attacks on logic locking schemes while avoiding drawbacks of previous proposals from the literature, i.e. insufficient output corruption and possible removal of the extra locking logic.

To this end, the contribution of this paper is a new type of key-gates providing near optimal solution in terms of output corruption when the device is not controlled from the correct key. Unlike solutions proposed in the literature, the associated locking scheme counteracts approximate and removal attacks. A discussion on resilience against SAT attacks and key-gate design requirements is also provided.

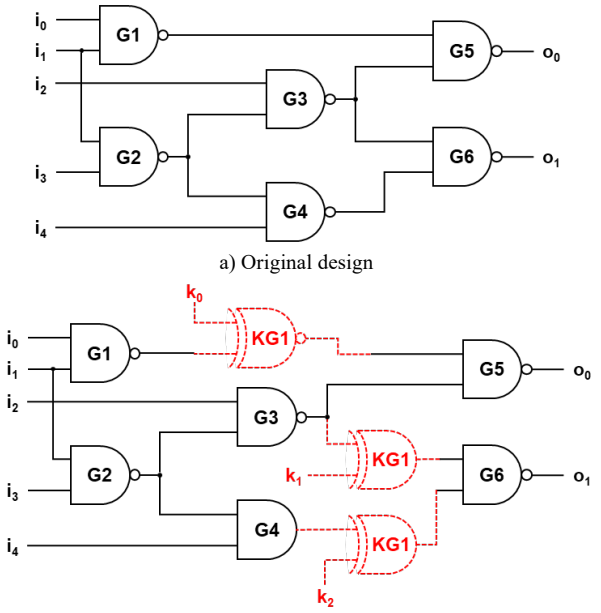
This paper is organized as follows. Section II presents a background on locking methods and attacks, focusing on SAT attacks and related countermeasures. Section III describes the proposed tristate-based logic locking technique. Resilience to approximate/removal/SAT attacks is discussed in Section IV and experimental results are presented in Section V. Eventually, Section VI concludes the paper and presents future explorations to fully validate this approach.

II. Background

The generally accepted threat model assumes that the attacker may be: i. an untrustworthy foundry, which can overproduce extra ICs from the GDSII provided by another company; ii. a system integrator, who can reuse IPs from 3rd parties in many projects; iii. an end-user, who can re-use or sell

the IP of a circuit. The attacker therefore has access to either the netlist of the circuit, or the GDSII file, or an activated chip from the market, which can be reverse engineered to obtain the layout and ultimately the netlist. The attacker does not know the correct value of the key for unlocking the circuit in its expected behaviour because they have in their possession either a locked circuit not yet programmed with the correct key (e.g., the foundry), or an activated circuit for which the value of the key is protected in a secure memory (e.g., an end-user). Additionally, in case the attacker has an activated chip, they can use it as an oracle to know the value of the original device's responses for any value of the inputs, in order to perform a so-called oracle-based attack.

The attacker's goal is to thwart the IP protection provided by logic locking, either by discovering the correct key value or by recovering the original unprotected netlist and use it without any need of knowing the correct key value.



a) Original design

b) One logic locking solution based on XOR/XNOR key-gates

Fig. 1. Logic locking principle (added key-gates in red dotted lines)

The initial proposal EPIC in 2008 [2] consisted in inserting XOR/XNOR key-gates into the original design at random. Fig. 1 shows an example of an original design (Fig. 1a) and a logic locking solution for that design (Fig. 1b) where XOR/XNOR key-gates have been introduced in the original design. The key value in this example is (k_2, k_1, k_0) = (1,0,1), so that key-gates controlled from k_0 and k_1 act as buffers and key-gate controlled from k_2 acts as an inverter (note that G4 NAND gate in the original design has been transformed into an AND gate after logic locking insertion).

Early research in logic locking has explored different types of key-gates such as MUXes or LUTs and more importantly, different insertion algorithms for optimal output corruption and resilience to the first attacks [3]. The insertion algorithm's goal is to define the interconnects on which the key-gates should be introduced. With a random key-gate insertion strategy, incorrect keys may lead to correct outputs for certain keys. The fault analysis-based logic locking proposal (FLL) [4] has been proposed for providing optimal output corruption when an incorrect key is used.

The first corruption metric used in the literature was the percentage Hamming distance between correct and corrupted outputs, ideally equals to 50%. In other words, an incorrect key value should affect half of the output bits on average [4]. It was then proposed in [3] to add a complementary metric, as the number of input vectors causing corruption, in order to ensure that most input vectors created corruption. A third metric was then proposed in [5] as the total number of outputs having been corrupted at least once, ideally all. In the sequel of the paper, we will use the following denomination:

Output corruptibility is the average percentage Hamming distance between correct and erroneous outputs,

Corruption rate is the percentage of input vectors that lead to output corruption,

Output coverage is the percentage of outputs that have been corrupted at least once.

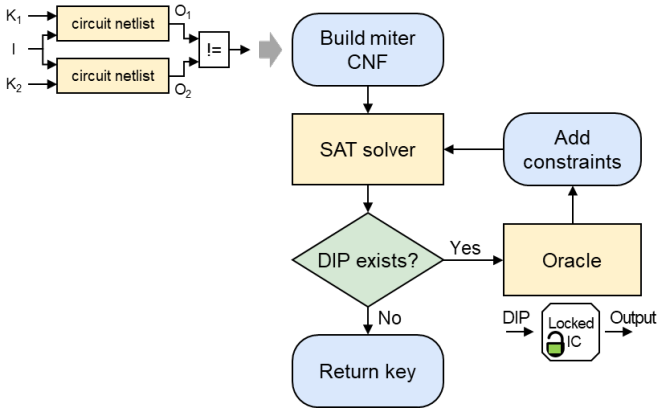


Fig. 2. The SAT attack framework, miter circuit employed by the attack and flowchart [6]

In 2015, an oracle-guided attack [6] broke all previously proposed solutions thanks to a satisfiability solver. The SAT attack consists in pruning out wrong key values iteratively. Its framework is detailed in Fig. 2. It is based on a miter circuit, i.e., a circuit containing two copies of the locked netlist, whose primary inputs are the same, but key-inputs are different. The attack formalizes this miter circuit into a Conjunctive Normal Form (CNF), the set of constraints is initially empty and all the potential key values K_j are included in the list of candidates for being the secret key (denoted K* in the following). Assuming a combinational design of function f(X, K), X being the primary input, K the key controlling the key-gates, and 1-bit y the primary output (for the sake of simplicity), two keys K_1 and K_2 are randomly selected and the SAT solver searches for an input X such that the two keys lead to different outputs, i.e. it solves the Boolean satisfiability problem f(X, K_1) != f(X, K_2). The X found is called a Distinguish Input Pattern (DIP). Then the oracle is exercised with first DIP (DIP_1) for discovering the expected "golden" output y1 = f(DIP_1, K*). This first iteration allows to discard either K_1 or K_2 from the list of potential locking-key candidates, either K_1 because f(DIP_1, K_1) != f(DIP_1, K*), or K_2 because f(DIP_1, K^2) != f(DIP_1, K*).

The new constraint $f(DIP_1, K_j) = y1$ for any j is added to the set of constrains to be satisfied by the solver. Any K_j that does not satisfy this new constraint is thus discarded from the list of key candidates in the following iterations. The process iterates while the solver can find two keys K_j, K_k , j != k, and an input X (DIP_i) such that $f(X, K_j) != f(X, K^k)$. By the end of the process, the list of candidates includes only the correct secret key K*.

Subsequent locking schemes therefore focused on thwarting the SAT attack and two types of strategies have been explored in the literature:

1. slowing down the attack, either by increasing the required number of SAT solver iterations for correct key retrieval, or by increasing computational time of each iteration,
2. preventing the attack from being launched. These alternatives are presented hereafter.

Anti-SAT [7] and SARLock [8] are based on the implementation of point-functions for corrupting the output in case of incorrect keys. For any wrong key K_j, only one input vector Xi exists such that $f(Xi, K_j) != f(Xi, K^*)$, and for any two wrong keys K_j and K_k, it exists only two inputs Xl and Xm for which $f(Xl, K_j) != f(Xl, K_k)$ and $f(Xm, K_j) != f(Xm, K_k)$. Consequently, it is not possible to discard more than one wrong key per SAT solver iteration. The DIP_i found to distinguish K_j from K_k cannot indeed distinguish another key candidate K_x since $f(DIP_i, K_x) = f(DIP_i, K^*)$ for any K_x. Thanks to the point functions, the number of iterations of the SAT solver N is maximal, $N = 2^{kb}$, kb being the number of key bits, resuming the SAT attack to a brute force attack. It's worth noting that for the same reasons, the point-function does provide an insignificant output corruption. The circuit indeed behaves properly (correct output) whatever the incorrect key set into the device, and whatever the input vector, except for one.

This new paradigm then paved the way for new types of attacks:

Approximate attacks: SAT oracle-guided attacks that find an approximate key value i.e. a key value that minimizes output corruption [9],

Removal attacks: oracle-less attacks that take advantage of the fact that this type of method no longer consists of key-gates dispersed in the circuit, but of an additional block, relatively easy to recognize and therefore remove [10].

New attacks and protections have been a real game of cat and mouse in recent years, with many proposals for improvements to the original SAT-resilient approaches, either in an attempt to compensate for the lack of output corruption (often at the expense of resistance to the SAT attack) [11] or to be resilient to new attacks [12][13]. One type of improvement has been particularly studied: Corrupt And Correct (CAC) schemes, which were initially proposed to prevent removal attacks. For those, the initial logic cone is minimally modified so that the added logic fixes the modifications only for the correct key. That way, removing the added logic does not result in the original circuit and is therefore useless [14]. Following CAC schemes have also followed the previous trend, looking for an acceptable trade-off between resilience to attacks and output corruption [15]. Note that in this frantic race for improvements, the cost of the implementation in terms of area overhead has eventually been undetermined by many of these methods.

In order to lengthen CPU time at each SAT iteration, it was first proposed to add SAT-hard structures such as cryptographic ciphers [16]. Not only this type of solution suffers from a large area overhead but it may also be identified and removed. More recently, Full-Lock proposed to insert programmable logic and routing blocks as SAT-hard instances because they contribute in creating an extremely large CNF [17]. This solution was however counteracted by a neural-network-guided SAT attack, which helps to significantly speed up the CPU iteration time [18].

In order to prevent the attack from being launched, it was proposed to encrypt the scan chains commonly implemented in sequential designs for post-manufacturing structural testing. This solution [19], while expensive, prevents usage of the oracle for differentiating two keys while the SAT-solver is supposed to be executed on a miter circuit based on the combinational part of the original circuit. It was also proposed to introduce dummy wires and gates in order to create logical loops [20], which was eventually countered by an enhanced SAT attack [21] and even more recently enhanced to resist to this new attack [22]. Another approach is to lock not only the functionality of a design but also its delay properties, which are not Boolean logical properties that cannot be model into a satisfiability problem. So-called delay locking introduces key-gates containing a tunable delay buffer aiming to alter setup and hold times [23]. These non-logical properties were however modelled and attacked by the improved SMT attack based on a Satisfiability Module Theory solver [24].

III. Logic locking with tristate-based key-gates

The new key-gate proposed in this paper is depicted in Fig. 3. It consists of a tristate buffer and a tristate inverter connected in parallel, whose enable signals are controlled by two distinct key-bits. It can be declined in four different versions named type 1 to type 4, depending on whether the tristate elements are active at high or low levels.

A major difference compared to conventional XOR/XNOR key-gates is that the proposed key-gate is controlled by two key-bits and has therefore four operating modes:

- The output is equal to the input when the buffer is active and the inverter is in the high-impedance state,
- The output is equal to the complement of the input when the inverter is active and the buffer is in the high-impedance state,
- The output is in high-impedance state (denoted Hi-Z) when both the buffer and the inverter are inactive, i.e. the output is blocked and retains the same value than in the preceding state,
- The output is in an unknown state (denoted UNK) when both the buffer and the inverter are active. In this case the value present on the output is an intermediate voltage comprised between GND and VDD, whose value depends on the relative sizing of the PMOS/NMOS transistors of the buffer and the inverter.

The logic interpretation of this voltage by subsequent gates depends on their threshold voltage.

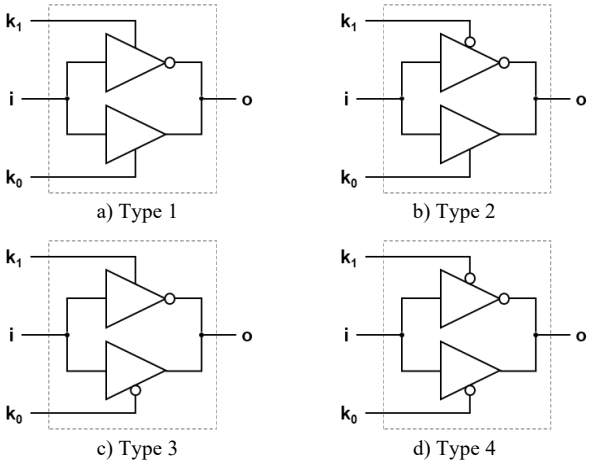


Fig. 3. Tristate-based key-gates

These four operating modes are summarized in Table I for the type 1 key-gate, where both the buffer and the inverter are active at high levels.

TABLE I. TRUTH TABLE OF THE TRISTATE-BASED KEY-GATE (TYPE 1)

k_1	k_0	o
0	0	Hi-Z
0	1	i
1	0	not i
1	1	UNK

The difference between the four types of tristate-based key-gates resides in the combination applied on the two key-bits that activates a given operating mode. For instance, the type 1 key-gate operates as a buffer with the combination (k_1, k_0) = (0,1), whereas combination (k_1, k_0) = (1,1) is required for type 2, combination (k_1, k_0) = (0,0) for type 3, and combination (k_1, k_0) = (1,0) for type 4.

The principle of the proposed solution is to incorporate the various types of tristate-based key-gates into the original design of a circuit. It is also assumed that the physical implementation of the key-gate is performed using layout camouflaging techniques so that all four types present a look-alike layout. INV/BUF camouflaged gates were investigated in [25], providing also protection against untrusted foundries since the function - INV or BUF - is programmed by the designer. In the present work, note that the proposed tristate-based key-gates provide not only two functions INV and BUF as in previous works (camouflaged INV/BUF or classical XOR/XNOR key-gates), but two other functions as well, Hi-Z and UNK. Camouflaging will be implemented on our 4 types of key-gates so that correlation between the combination applied on the two key-bits and the gate operating mode cannot be established.

As in the classic method with XOR/NXOR key-gates, some of the inserted key-gates will have to operate as buffers, while others will have to operate as inverters to ensure correct functionality, as illustrated by the simple example of Fig. 1. An attacker unaware of the original netlist has no way of knowing what the key-gate's expected behavior is. Furthermore, depending on the type of inserted gate, any combination of key-bits can be a valid one. However, the combination may also yield to the UNK or the Hi-Z state, which are obviously forbidden states for the correct operation of the circuit. By implementing in a design all four gate types in a camouflaged way, we therefore expect to increase confusion while ensuring high corruption as long as the circuit is locked.

Finally, the transistor-level description of the key-gate (type 1) is described in Fig. 4. It comprises 14 transistors, the same number as the classical implementation of a XOR gate, which can be reduced to 6 transistors for the most compact design. In first approximation, the area overhead induced by the tristate-based key-gate is therefore equivalent to that of a classical XOR key-gate.

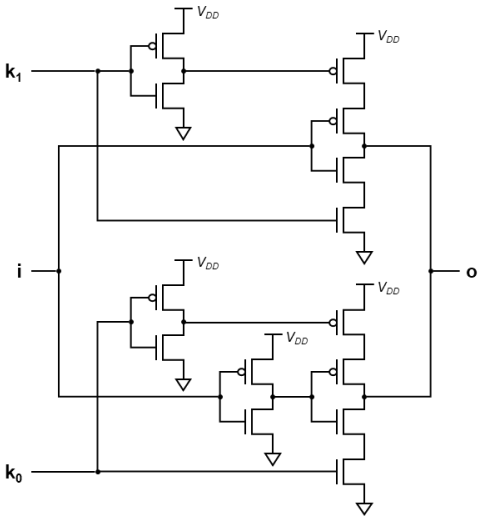


Fig. 4. Transistor-level description of the tristate-based key-gate (type 1)

IV. RESILIENCE TO ATTACKS

Like pre-SAT approaches, the proposed solution is based on the insertion of key-gates and not on the addition of 1-point functions or CAC-like schemes. Therefore, it targets high corruption of the outputs in case of incorrect keys, whatever the input pattern and the output bit, and it is immune to removal attacks while maintaining a non-prohibitive area overhead. Its ability to generate good corruption will be shown in Section V, a characteristic that makes it immune to approximate attacks.

In order to thwart SAT attacks, the proposed solution must be complemented with camouflaging measures. Despite the insertion of different types of tristate-based key-gates in the design, reverse engineering or knowledge of the GDSII file indeed provides information on the exact implementation of the key-gates and thus on "forbidden" combinations for the 2-bit key, i.e. (k_1, k_0) = (0,0) or (k_1, k_0) = (1,1) for the type 1 key-gate for instance. With this knowledge, the attacker can add extra constraints to the SAT solver such as k_0 != k_1 for type 1 key-gate again, or similar appropriate constraints for other types. These constraints allow to avoid wrong key combinations in the initial list of key candidates to be explored by the SAT solver.

For any attacker who does not know the correspondence between the key value and the operating mode (buffer/inverter/ UNK/Hi-Z), the proposed solution is valid. Finding this

correspondence is indeed impossible with the envisaged threat model: recovering f(X, K, I), X being the data in, K the key, I the function (buffer/inverter/UNK/Hi-Z) would indeed require several trials on a real circuit varying both X and K in order to discover I. This could be achieved using a modified version of the SAT attack as introduced in [26]. This solution nevertheless requires the ability to apply vectors [X, K] on an oracle and observe corresponding outputs. In other words, it requires to stimulate an oracle with a free control on K, which is not part of the threat model considered in this paper.

V. Experimental results

In order to assess the quality of the proposed solution, we locked several ISCAS benchmarks [27], inserting 5% extra key-gates and using either the conventional XOR/XNOR key-gates or the new tristate-based key-gates. Two insertion algorithms were considered, namely the FLL algorithm [4] and an improved version of the FpLL algorithm [5] for better output corruption and faster execution time.

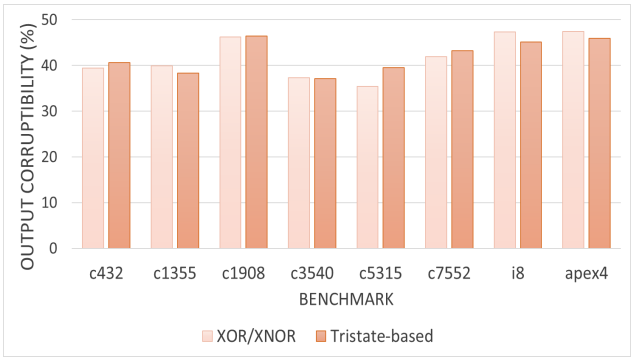
Output corruption is evaluated according to the three metrics defined in Section II: output corruptibility, corruption rate and output coverage. Computation of these metrics is based on simulation results, using 1,000 random test vectors each applied for 100 random incorrect keys (i.e. 100,000 test vectors in total).

Results are summarized in Table II, which reports the mean value of the three metrics computed over 8 benchmark circuits, for the conventional XOR/XNOR and the proposed tristate-based key-gates. It can be observed that, whatever the metric considered and the insertion algorithm used, the two types of key-gates yield to similar results. More specifically, the difference between XOR/XNOR and tristate-based key-gates is below 0.3% in terms of output corruptibility and corruption rate. A slight improvement of 2.5 to 3.5 % is observed in output coverage when using tristate-based key-gates. Regarding the insertion algorithm, the improved version of the FpLL algorithm yields to slightly better results than the FLL algorithm for output corruptibility and output coverage, and a minor degradation of corruption rate that still remains higher than 98.5%.

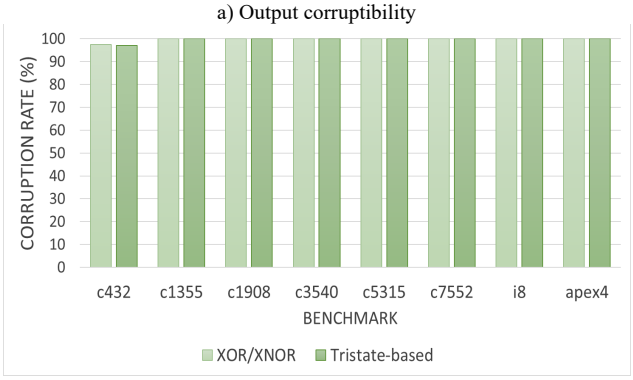
TABLE II. Comparison of corruption metrics between conventional XOR/XNOR and tristate-based key-gates: mean values over the 8 benchmark circuits using two insertion algorithms

Insertion Algorithm	Corruption Metric	Mean Value	
		XOR/XNOR	Tristate-based
FLL	Output corruptibility	41.9%	42.0%
	Corruption rate	99.7%	99.6%
	Output coverage	66.6%	70.1%
FpLL	Output corruptibility	43.5%	43.4%
	Corruption rate	98.5%	98.8%
	Output coverage	68.9%	71.4%

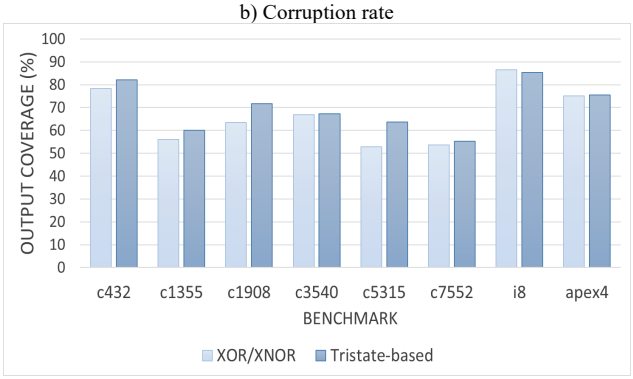
Fig.5 gives a more detailed comparison for the 8 benchmark circuits, when using the FLL algorithm. Regarding output corruptibility, a slight improvement is observed for 4 of the circuits when using tristate-based key-gates, and a slight degradation for the other 4. The maximum degradation is observed on the i8 circuit with output corruptibility reducing from 47.3% to 45.1%, which still corresponds to satisfactory corruption. Regarding corruption rate, equivalent results are obtained for both types of key-gates, with a corruption rate of 100% for all circuits except the smallest one. Finally, regarding output coverage, tristate-based key-gates lead to a slight improvement for all circuits, expect the i8 one for which there is a minor reduction of -0.3%. Overall, these results validate the ability of the proposed solution to generate good corruption since it achieves the same levels than classical XOR/XNOR key-gates.



a) Output corruptibility



b) Corruption rate



c) Output coverage

Fig. 5. Output corruption comparison, with FLL insertion algorithm

It is important to point out that these experiments were performed with an equivalent number of inserted key-gates, i.e. similar gate-equivalent (GE) area overhead. However, because tristate-based key-gates are supplied with two key-bits instead of one for XOR/XNOR key-gates, key size is doubled. A further experiment was carried out, dividing by two the number of inserted tristate-based key-gates, in order to retain the same key size. Results are reported in Table III, which gives the mean values of the three metrics obtained with the two insertion algorithms considered. In case of tristate-based key-gates, the

mean value was calculated either on all 8 benchmark circuits or only on the 4 largest (more than 2,000 gates). These results show that reducing the number of key-gates inserted leads to a degradation of corruption metrics when all the circuits are taken into account, in particular output corruptibility, but almost no degradation when only the 4 largest circuits are taken into account. This can be explained by the fact that a minimum number of key-gates have actually to be inserted to achieve satisfactory output corruptibility, condition which is not met when only 2.5% of key-gates are inserted in small circuits (less than 50 key-gates inserted). Overall, these results reveal that the proposed solution has the potential to achieve the same level of corruption as that obtained with conventional XOR/XNOR key-gates, but with a GE area overhead reduced by a factor of two, for circuits of sufficient size.

TABLE IIII. CORRUPTION METRICS ACHIEVED BY USING HALF AS MANY TRISTATE-BASED KEY-GATES: MEAN VALUES OVER ALL 8 BENCHMARK CIRCUITS OR ONLY THE 4 LARGEST

	Corruption Metric	Mean Value		
		XOR/XNOR (5% key-gates)	Tristate-based (2.5% key-gates)	
			8 benchmarks	4 largest
FLL	Output corruptibility	41.9%	37.2%	40.8%
	Corruption rate	99.7%	97.8%	99.7%
	Output coverage	66.6%	67.0%	69.8%
FpLL	Output corruptibility	43.5%	35.3%	42.2%
	Corruption rate	98.5%	98.6%	100%
	Output coverage	68.9%	66.8%	70.8%

VI. CONCLUSION

This paper introduced a new type of key-gates based on tristate logic. The goal of these key-gates is to provide near optimal output corruption as expected on locked circuits. This first property is attested by experimental results presented in this paper and thus prevents approximate attacks. In addition, dissemination of key-gates in the original design as proposed in pre-SAT logic locking approaches prevents the locking circuitry to be simply removed by an attacker. Last but not least, key-gates' functionality being mandatory for execution of a SAT attack, the proposed tristate-based key-gates (4 types randomly distributed in the design) strengthen resistance to SAT attacks compared to XOR/XNOR key-gates, as long as tristate-based key-gates' functionality remains hidden. UNK and Hi-Z states prevent direct CNF modelling, and stimulation of an oracle with chosen keys for identification of these states remains difficult and generally not covered by the classic threat model. Future work will target the practical implementation of layout camouflaging techniques for undifferentiation of types 1-4 tristate-based key-gates in order to prevent function identification. Oracle-less attacks based on deep learning will also be investigated, knowing that they currently rely on the properties of XOR-based locking.

ACKNOWLEDGMENT

This work was partly funded by the French National Research Agency (ANR) under the project MOOSIC ANR-18-CE39-0005. The authors would like to thank G. Gouvine (gabriel.gouvine_moosic@m4x.org) for valuable discussions on SAT solvers.

REFERENCES

[1] M. Yasin, J. Rajendran, O. Sinanoglu. "Trustworthy Hardware Design: Combinational Logic Locking Techniques". Springer, 2020.
[2] J. A. Roy, F. Koushanfar and I. L. Markov. "EPIC: Ending Piracy of Integrated Circuits". In *Design, Automation and Test in Europe (DATE)*, pp. 1069–1074, 2008.
[3] S. Dupuis and M.-L. Flottes. "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics", In J*ournal of Electronic Testing: Theory and Applications (JETTA)*, 35, pp. 273-291, 2019.
[4] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu and R. Karri, "Fault Analysis-Based Logic Encryption", In *IEEE Transactions on Computers*, 64(2), pp. 410-424, 2015.
[5] Q.-L. Nguyen, S. Dupuis, M.-L. Flottes, B. Rouzeyre. "SKG-Lock+: A Provably Secure Logic Locking Scheme Creating Significant Output Corruption". In *Electronics*, 11(23):3906, 2022.
[6] P. Subramanyan, S. Ray and S. Malik, "Evaluating the Security of Logic Encryption Algorithms", In IE*EE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 137-143, 2015.
[7] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(2), pp. 199-207, 2019.
[8] 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, 2016.
[9] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits", In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp .95-100, 2017.
[10] M. Yasin, B. Mazumdar, O. Sinanoglu and J. Rajendran, "Security Analysis of Anti-SAT", In *Asia and South Pacific Design Automation Conference*, pp. 342-347, 2017.
[11] B. Shakya, X. Xu, M. Tehranipoor and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme", In *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1), pp. 175-202, 2019.
[12] A. Rezaei and A. Mahani, "Noise-based logic locking scheme against signal probability skew analysis", In *IET Computers & Digital Techniques*, 15(4), pp. 279-295, 2021.
[13] J. Zhou and X. Zhang, "Generalized SAT-attack-resistant logic locking", In *IEEE Transactions on Information Forensics and Security*, 16, pp. 2581-2592, 2021.
[14] M. Yasin, B. Mazumdar, J. Rajendran and O. Sinanoglu "TTLock: Tenacious and traceless logic locking", In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 166-166, 2017.
[15] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice", In *ACM SIGSAC Conference on Computer & Communications Security (CCS)*, pp. 1601-1618, 2017.
[16] M. Yasin, J. Rajendran, O. Sinanoglu and R. Karri, "On Improving the Security of Logic Locking", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* 35(9), pp. 1411-1424, 2016.
[17] H. M. Kamali, K. Z. Azar, H. Homayoun and A. Sasan, "Full-Lock: Hard Distributions of SAT instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks", In *Design Automation Conference (DAC)*, pp. 1-6, 2019.
[18] K. Z. Azar, H. M. Kamali, H. Homayoun and A. Sasan, "NNgSAT: Neural Network guided SAT Attack on Logic Locked Complex Structures", In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1-9, 2020.
[19] Q.-L. Nguyen, E. Valea, M.-L. Flottes, S. Dupuis and B. Rouzeyre, "A Secure Scan Controller for Protecting Logic Locking", In *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1-6, 2020.
[20] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan and Y. Jin. "Cyclic Obfuscation for Creating SAT_Unresolvable Circuits", In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 173-178, 2017.
[21] H. Zhou, R. Jiang and S. Kong, "CycSAT: SAT-Based Attack on Cyclic Logic Encryptions", In *International Conference on Computer-Aided Design (ICCAD)*, pp. 49-56, 2017.
[22] X.-M. Yang, P.-P. Chen, H.-Y. Chiang, C.-C. Lin, Y.-C. Chen and C.-Y. Wang, "LOOPLock 2.0: An Enhanced Cyclic Logic Locking Approach", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(1), pp. 29-34,2022.
[23] Y. Xie and A. Srivastava, "Delay Locking: Security Enhancement of Logic Locking against IC Counterfeiting and Overproduction", In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2017.
[24] K. Z. Azar, H. M. Kamali, H. Homayoun and A. Sasan, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performances Beyond the SAT Attacks", In *IACR Transactions on Cryptographic Hardware Embedded Systems (CHES)*, Vol. 2019, Iss. 1, 2019.
[25] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "CamoPerturb: Secure IC camouflaging for minterm protection," In *Proc. IEEE/ACM International Conference on Computer-Aided-Design*, pp. 29:1-28:8, 2016.
[26] C. Yu, X. Zhang, D. Liu, M. Ciesielski and D. Holcomb, "Incremental SAT-Based Reverse Engineering of Camouflaged Logic Circuits," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1647-1659, Oct. 2017.
[27] ISCAS benchmark open source: https://pld.ttu.ee/~maksim/benchmark