Power Analysis Attack Against post-SAT Logic Locking schemes

Nassim Riadi, Florent Bruguier, Pascal Benoit, Sophie Dupuis, Marie-Lise Flottes

Power Analysis Attack Against post-SAT Logic Locking schemes

Nassim Riadi, Florent Bruguier, Pascal Benoit, Sophie Dupuis, Marie-Lise Flottes

LIRMM, Univ. Montpellier/CNRS
Montpellier, France
first-name.last-name@lirmm.fr

***Abstract*—Due to the globalization of the semiconductor industry, Integrated Circuits (ICs) and Intellectual Properties (IPs) are susceptible to specific threats. IP piracy, overproduction, and introduction of hardware Trojans can indeed compromise valuable design information and trust in the design flow. Logic Locking (LL) is one of the most popular Design-for-Trust techniques that aims to thwart these threats because of the wide range of risks it can prevent. This approach evolves from year to year in order to make it resistant to ever more advanced attacks. While most advanced LL solutions are assumed to be resistant against differential power analysis (DPA), we propose a new attack framework for challenging these approaches and show on several benchmarks that it is possible to reveal more than 88% of the key bits used for locking the designs thanks to DPA.**

***Index Terms*—Design-for-Trust, Logic Locking, Differential Power Analysis.**

I. Introduction

Numerous frauds in the integrated circuit production have been made possible as the semiconductor industry has evolved over the decades from a vertical to a horizontal model with many entities involved in the design, integration, manufacturing, assembly and testing of a chip. Security and trust have since received particular attention for preventing piracy, copy, cloning, hardware Trojan insertion.

Design-for-Trust (DfTr) solutions [1] have thus emerged to thwart the theft of intellectual property during production, prevent any malicious modification during manufacturing and avoid copying after deployment on the market. Passive watermarking and fingerprinting techniques at the end of the 90's consist of embedding designer's signature and end-user's one, respectively. They can be used to detect piracy but cannot prevent it. Split manufacturing [2] consists in manufacturing two parts of a design, front-end and back-end-of-line, in separate foundries. It prevents piracy by an untrusted foundry. Camouflaging [3] replaces gates with camouflaged counterparts which look alike from the top but where the true functionality remains unknown. It aims at preventing reverse engineering by end-users. Logic Locking (LL) [4][5] consists in inserting extra logic into the design so that it becomes functional upon activation with a secret key defined by the designer. This approach received the most attention because it prevents piracy, overbuilding and reverse engineering and thus provides protection against the largest set of untrusted entities, SoC integrators, foundries, test facilities and end-users.

This work was funded by the French National Research Agency (ANR) under the ARSENE project (ANR-22-PECY-0004)

Since the end of the 2000s, Logic Locking solutions have however been challenged by new attacks. In 2015, a Boolean Satisfiability based attack (SAT attack) [6] has been presented that broke all the combinational logic locking techniques proposed so far. The attack aims at gradually pruning the key search space. SARLock [7] and Anti-SAT [8] are examples of countermeasures against the SAT attack. They aim at minimizing the number of keys gradually discarded, rendering the attack effort exponential in the number of secret key bits. These countermeasures are however vulnerable to structural attacks [9] consisting in removing the locking logic from the attacked design for retrieving the original function. In most advanced solutions such as TTLock [10], SFLL-HD [11] and SFLL-Flex [11], the original function is modified to hide the correct implementation and thwart structural attacks.

Another physical weakness that can be exploited for key recovering is the data-related power consumption. To the best of our knowledge, the only works that focus on differential power analysis (DPA) for key recovery on LL schemes are those mentioned in [12] where authors propose a framework to perform DPA on pre- and post-SAT LL techniques. They argue that a logic-locked circuit has the same resilience against the SAT attack and the DPA attack (with the proposed framework), and conclude on the provable security of post-SAT schemes against DPA.

In this paper we propose a new framework to perform DPA on the SAT resilient technique SFFL-HD^0/TTLock and validate the proposed attack on benchmark circuits. Experiments show that more than 88% of key bits can be retrieved on this LL technique with the proposed framework.

The remainder of this paper is organized as follows. Section II provides a background on logic locking and power analysis attacks with particular attention to the SFFL-HD technique. Section III presents a new framework to perform DPA on SFFL-HD^0. Section IV describes experiment and results on simulated benchmark circuits. The paper is concluded in Section V.

II. Background

A. Logic Locking

Logic Locking is a DfTr technique that aims to prevent many ICs and IPs threats such as Reverse Engineering, Overproduction or IP Piracy. LL adds logic into an original design so that the resulting IC after manufacturing cannot

perform the expected function until the designer has unlocked it with her/his secret key after manufacturing.

Formally, I being the n-bits circuit input, Ks the m-bits designer secret key, $F_i(I)$ the true and initial function of the design before the introduction of the locking logic (i.e. original design), F(I, K) the function after the introduction of the logic locking elements:

- The true function is executed after LL when the correct key is provided,
$$F(I, Ks) = F_i(I), \forall I \in (0, 1)^n$$
- Output corruption occurs at least for some inputs when any incorrect key is provided,
$$\exists I \in (0, 1)^n \mid F(I, K \neq Ks) \neq F_i(I), \forall K \neq Ks$$

First LL techniques were based on the insertion of key-gates into the original netlist, typically XOR/XNOR gates. The insertion strategy has evolved from a purely random approach to optimized approaches for output corruption improvement and resistance to sensitization attack [13]. Optimal corruption corresponds to 50% of erroneous outputs over all the output bits and the processed input data.

The SAT attack [6] was successful against all these former LL schemes. The threat model for this attack assumes that the attacker has access to: i., a functional integrated circuit obtained from the market, the 'oracle' already programmed with the correct key by the designer, and ii., a locked netlist, stolen or legally acquired from the IP provider, or obtained from reverse engineering. The SAT attack consists in pruning out incorrect keys iteratively. It relies on a miter circuit including two copies of the locked netlist whose primary inputs are interconnected 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 are included in the list of candidates for being the secret key. Two keys Ki, Kj are randomly selected and the SAT solver searches for an input data I such that the two keys lead to different outputs, i.e. it solves the Boolean satisfiability problem: Find I such that F(I, Ki) != F(I, Kj). The I found is called a Distinguish Input Pattern (DIP). Then the oracle is exercised with the DIP for discovering the expected "golden" output value. This iteration allows to discard either Ki or Kj from the list of potential secret keys. The new constraint F(DIP, K) = Fi(DIP) for any K is added to the solver. Any key that does not satisfy this new constraint is thus discarded from the list of candidates for the following iterations. The process iterates while the solver can find DIPs for key differentiations. By the end of the process, the list of candidates includes only the correct secret key.

In order to thwart this attack, more recent LL approaches rely on the limitation of output corruption in case of an incorrect key. This strategy prevents key discarding in each iteration and thus increases the number of iterations performed by the SAT solver. Anti-SAT for instance [8] integrates a point function with the original netlist in such a way that for any incorrect key, the output differs from the excepted one for only one input vector and this input value is different for every incorrect key. This property prevents discarding more than one key per iteration (only one incorrect key produces a corrupted output for a given DIP). Thanks to the point functions, the number of iterations of the SAT solver is maximal (2^key-bits), resuming the SAT attack to a brute force attack. However, the locking logic in Anti-SAT is implemented apart from the original function, which remains implemented as is. After identification and removal of the locking logic, an attacker accesses the original netlist that no longer requires any key to be unlocked. Because of removal attacks, most advanced LL solutions hide their true implementation by modifying the original logic cones. Fig. 1 depicts the concept used in SFLL-HD. The Functionality-Stripped Circuit (FSC) is obtained by resynthesizing the original circuit together with a perturb unit. This first extra unit is in charge of corrupting the output but only for a given number of protected inputs vectors that satisfy the designer-defined Hamming distance condition: HD(I,K) = h. The restore unit restores the correct output for protected input vectors when the correct key is provided. With increased h, the number of protected vectors increases, removal attack resilience increases and the SAT attack resilience decreases. For h=0 (TTLock/SFLL-HD^0), only one input vector is protected, providing very low output corruption but also the best SAT attack resilience. Finally, SFLL-flex, the most complex SFLL-like scheme, allows to protect a specified set of input vectors. For SFLL-HD^0, HD(I,K) = 0, in other words the secret key is the protected input vector. The FSC is minimally different from the original function, it produces a corrupted response when the input equals the secret key while the restore unit cancels this error if controlled with the correct key (no primary output corruption). There is no corruption on FSC output when the input is different from the key, but the restore unit introduces an error on the primary output for an incorrect key equal to the processed input. To sum up, only two inputs vectors can lead an incorrect key to produce an error on output, the input vector equal to this key and the input vector corresponding to the correct key. In comparison, for any incorrect key, only one input vector can lead to an erroneous output with the Anti-SAT technique. However, SFLL-HD^0 is still very resilient to the SAT attack because discarding more than one incorrect key in one iteration is not possible, except when one of the two randomly selected keys corresponds to the correct key. The probability of finding this correct key is thus equal to q/2^key-bits where q is the number of queries to the oracle (i.e. the number of SAT iterations). SFLL-HD^0 is thus k-secure against the SAT attack as demonstrated in [11].

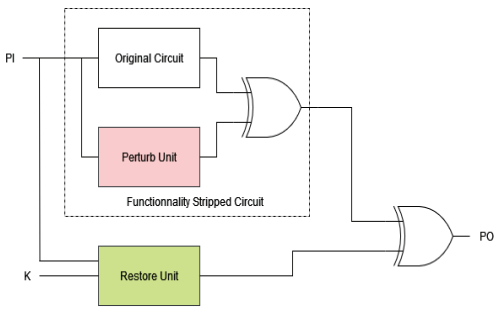

Fig. 1. SFLL-HD [11]

B. DPA Attack

Power channel attacks exploit the fact that power consumption of CMOS ICs is correlated with the data being processed. The DPA attack [14] statistically correlates this consumption and the processed data. As for the SAT attack, the threat model assumes an attacker with access to an oracle (functional integrated circuit) and a locked netlist. A DPA attack starts with the collection of numerous power traces Ti on the oracle with random data on its inputs. Then a statistical analysis of these samples with the inputs processed during power trace collection is performed to determine the unknown processed data, i.e. the secret key. For this, a DPA selection function D is defined by computing an intermediate bit logic value according to the value of all possible key guesses and all the input data processed during power trace collection. The logic value 0 (resp. 1) of this bit dictates the classification of the corresponding power trace Tij into bin T0 or T1.

$$T0 = Tij \mid D(Ii,Kj) = 0$$
$$T1 = Tij \mid D(Ii,Kj) = 1$$

Next steps consist in computing mean power values for these two bins, respectively M0 and M1, and their absolute difference of means (DoM): DoM = |M1−M0|. When the correct key guess happens, the selection function correctly classifies power traces into bins T0 (intermediate bit actually equals 0) and T1 (intermediate bit actually equals 1), M1 >> M0 and DoM is high. For an incorrect key guess, the selection function does not match with actual data, power traces Tij are randomly classified into T0 and T1 since the target bit logic value is not correctly predicted, M0and M1 have similar values and their DoM tends to 0.

DPA attack on an entire netlist requires most of the time a divide-and-conquer approach for decreasing the number of key bits to handle at a time. If the target intermediate bit used in the selection function D depends on all the key bits, the DPA indeed resumes to a brute force attack. Using intermediate bits for the selection functions, which depends on a reduced number of key bits, allows exhaustivity on a lower number of key bits.

In the experiments presented in [12], the divide-and-conquer strategy consists in targeting logic cone outputs, one at a time. The sub-keys exhaustive analysis is thus limited by the number of key bits involved in the logic cone. In addition, authors recommend to begin the attack on logic cones for which the key size is the smallest, then to deal with largest ones and key bits still unresolved from previous steps since same key bits can feed several cones. As pointed in [12], the weak output corruption on cone outputs in case of incorrect keys with SFLL-HD0 makes differential power analysis very difficult because incorrect keys lead most of the time to the same behavior and outputs than the correct one. For the same reason, the SAT attack may need to exhaustively prune all the incorrect keys before to discover the correct one.

In the following sections, we present a new DPA framework for key recovery on LL schemes. The target bits for the decision function are not the output bits of logic cones but intermediate nodes in the restore units. These units are also dependent on the secret key, their consumption is thus related to the key. This new selection of target bits for decision functions D allows to limit the number of key bits to deal with in every iteration of the DPA attack.

III. DPA Attack Against SFLL-HD^0

This section details the proposed framework for DPA attacks on the LL technique SFLL-HD^0. In SFLL-HD^h, a Restore Unit (RU) returns 1 when HD(I,K) = h, i.e. when protected input patterns have a Hamming Distance h with the secret key. In SFLL-HD^0, the restore unit returns 1 when HD(I,K) = 0, so when the input data I corresponds to the secret key K. SFLL-HD is equivalent to TTLock in this case. SFLL-HD^0 is selected among the different versions of the SFLL-HD approach because it provides the best resistance to the SAT attack.

A. Targeted intermediate bits and framework

The RU in SFLL-HD^0 can be implemented with a simple comparator as in TTLock, instead of a Hamming Distance checker [15]. In order to limit the number of key bits to deal with during one DPA attack, our divide-and-conquer strategy defines selection functions D by computing intermediate RU bit logic values. This strategy limits the number of key bits to deal with for each targeted intermediate bit, and thus the number of possible guesses about sub-keys.

Compared to previous work, we infer the DPA decision function by partitioning the RU and not the FSC. With logic cone partitioning on the FSC, the number of key bits in each partition depends on the protection provided to the cone, i.e. the number of protected vectors. It has been shown that logic cones with more than 32 key bits are difficult to break with DPA, it is thus recommended to concentrate most of the key bits in a single logic cone of the original circuit for better protection.

With the proposed strategy, we can choose the size of the partitions on the RU, i.e. the number of key bits involved in the comparison performed at the intermediate RU bit. Subsequently, we present a DPA framework for intermediate RU bits corresponding to the comparison of 4 input bits with 4 key bits, limiting the exhaustive key list analysis to 2^4=16 iterations.

Fig. 2 shows an example of a sub-part of the RU where the target intermediate bit X switches to logic 1 when the key bits values K_0-K_3 match with the input bits values I_0-I_3, respectively. During DPA, the selection function allows to classify power traces into bins T0 and T1 according to the value of this intermediate bit. The process iterates for every 4-bit sub-key.

The attacker first partitions the netlist into logic cones from the primary outputs. Then he/she recovers the RU from each partition. This RU is controlled for one part, from a tamper-proof memory storing the secret key, and for the other part, from the primary inputs. The RU ends up on the last XOR gate driving the output of the current logic cone. The attacker then parses the RU for identification of smallest comparators (cf. Algorithm 1). These sub-comparators correspond to a logic cone ending at a node X as depicted in Fig. 2, i.e. an intermediate bit of the RU controlled from 4 key bits and 4 input bits. The logic cone for bit X is defined as a single-output Boolean function D_c(Ip, Kp), c being the sub-comparator number, Ip and Kp the p-bits data input and p-bits key input respectively, with p=4 in following experiments.

The divide-and-conquer DPA attack targets one sub-comparator at a time.

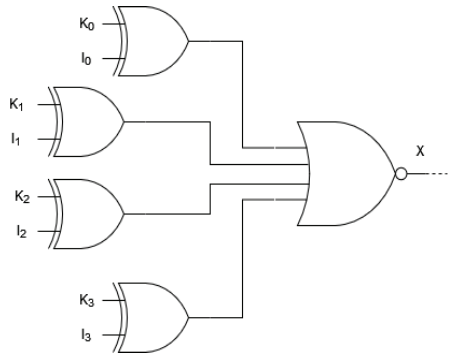

Fig. 2. Sub-part of the restore unit.

Algorithm 1 Partioning and decision functions inferring

INPUT: Locked Netlist (Lns)
OUTPUT: Decision functions for each
n: number of key bits and input bits
involved p: number of bits concerned by the partition
c ← 1 {number of partitions}
Kpar ← 0 {bits sorted}
r bits ← *n* {remaining bits}
while *r bits* ≠ 0 **do**
  D_c(Ip,Kp) ← NetlistParsing(Lns, r_bits)
  Kpar ← p
  r_bits ← r_bits – Kpar
  c ← c+1
end while

B. Locking and Power Traces Acquisition

The technique used for LL is the one detailed in [11]. Cadence Genus is used for logic synthesis on STM 28nm FD-SOI process node.

Input vectors for power trace acquisition are randomly generated. Potentially, inputs corresponding to RUs not involved in the current DPA attack can be kept constant for improving SNR for the RU under attack, as an attacker could do.

For power trace acquisition, we simulate the design with Cadence Xcelium for logic simulation, and Joules for power estimations (Fig. 3). For each simulated vector, Xcelium provides information on switching activity during a time frame set according to the RU's critical path. Joules estimates the corresponding power consumption, all contributions Leakage, Internal and Switching are selected for estimation of the total power consumption related to the processed vector.

For information, CPU time for power estimation on 1k vectors is a few seconds on 32 Intel Xeon Skylake CPU with 128 Gb of ram.

The Python script details in Algorithm 2 performs the statistical analysis on power traces (DPA attack).

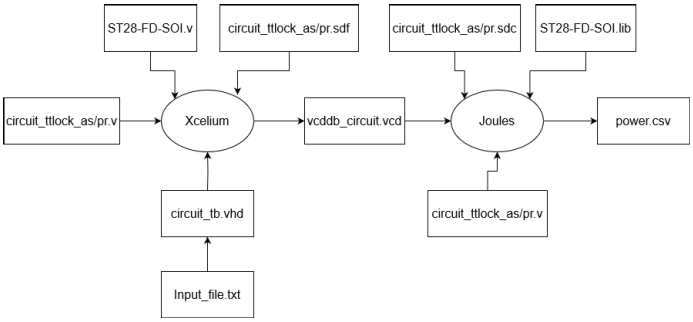

Fig. 3. Power acquisition flow.

Algorithm 2 DPA attack algorithm

INPUT: Set of power consumptions
INPUT: Set of input vectors
INPUT: D_c(Ip,Kp) for each sub-comparator
OUTPUT: *Kr* {key bits recovered}
*t*0,*t*1 {sum of power measurements when D_c=0 (resp. 1)}
*nt*0,*nt*1 {number of vectors in each set}
while *there is a new sub-comparator not yet analyzed* **do**
  for *every key guess* **do**
    for *every vector* **do** // binning
      if *Dc(Ip,Kp)* = 0 **then**
        *nt*0 = *nt*0+1
        *t*0 = *t*0+power estimated for the trace
      else
        *nt*1 = *nt*1 + 1
        *t*1 = *t*1+estimated power for the trace
      end if
    end for
    $DoM = \left|\frac{t0}{nt0} - \frac{t1}{nt1}\right|$ *// for the current key guess*
    $t0, t1 = \emptyset$; *nt0=nt1*=0
  end for
  Records key bits recovered
end

C. Preliminary DPA results on a Restore Unit

We first conducted a DPA attack against a single 16 bits RU as a standalone component. No other power consumption than the one induced by the RU is recorded, providing artificially a high SNR for the RU w.r.t other contributions in the circuit such as the one of the FSC.

Netlist parsing is performed with Algorithm 1. Four sub-comparators of 2x4 input bits each (4 key bits, 4 primary input bits) are identified. Their outputs correspond to the 4 target intermediate bits used for binning during DPA.

The DPA attack is iterated on the 4 sub-comparators using 4k random patterns at each iteration. Fig. 4 shows the result of the attack against the first sub-comparator in the RU. DoM values for every key guess have been normalized on the

graph. The maximal DoM actually corresponds to the correct key i.e. $(11)_{10}$. Similar results are obtained on other sub-comparators, the DPA attacks against this restore unit succeed to retrieve 100% of the secret key bits on every sub-comparator and thus 100% of the key bits for the RU considered as a standalone function. Next section presents experimental results on benchmark circuits where RU and FSC consumptions participate to the total power consumption of the circuit.

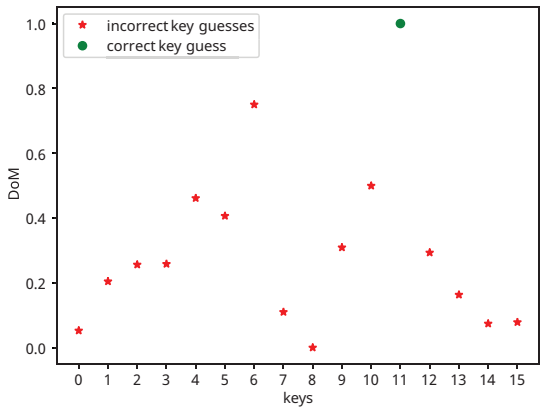

Fig.4. Normalized DoM values for the 16 4-bits key guesses on one sub-comparator

IV. DPA Attack on LL Benchmarks Circuits

The proposed DPA framework was performed on 4 ISCAS'85 benchmark circuits, c432, c880, c6288, c3540 and one 16-bits adder 16ADD. The SFLL-HD^0 strategy is implemented on each circuit using as many key bits as the number of inputs (TABLE I).

TABLE I
LOCKED BENCHMARC CIRCUITS

Circuits	Input size (#bits)	Key size (#bits)
c432	36	36
c880	60	60
c6288	32	32
c3540	50	50
16ADD	32	32

TABLES II and III present attack results on these circuits. TABLE II is based on the number of correct sub-keys retrieved over the total number of sub-keys (i.e. the number of sub-comparators). The 2nd column reports the total number of power traces used during the attack whatever the outcome. A maximum of 16k vectors have been simulated when a sub-key has not been totally recovered. The recorded number of vectors exercised when the correct sub-key has been totally recovered is the minimum number of vectors for recovering the key. A minimum of 75% of the sub-keys have been correctly guessed by the DPA attack on the experimented benchmark. TABLE III reports DPA attacks in terms of total number of key bits correctly guessed by the attack. More than 84% of these key bits have been correctly guessed in the worst case.

TABLE II
SUB-KEYS RETREIVED

Circuits	Power traces	RU area ratio	Sub-key correctly guessed (%)
c432	42k	39.04	82
c880	80k	38.76	80
c6288	46k	4.76	75
c3540	64k	17.39	82
16 ADD	34k	46.84	87.5

TABLE III
KEY BITS RETRIEVED (%)

Circuits	c432	c880	c6288	c3540	16 ADD
Key bits correctly guessed (%)	88.88	95	84.375	90	96.875

The attack success rate is mainly dependent on the useful information embedded in the captured power traces. One important parameter is the area ratio (and related power consumption) of the attacked unit compared to the area of the whole design. The 3rd column in TABLE II presents the restore unit area ratio compared to the entire circuit. RU in c6288 presents the smallest area ratio compared to other benchmark circuits with only 5% of the total area. This is also the circuit for which the lowest percentage of sub-keys and the smallest percentage of key bits have been correctly guessed by the DPA attack. Conversely in 16ADD where the RU presents the highest area ratio, more than 87% of sub-keys and more than 96% of the key bits have been correctly guessed despite the expected resistance to the SAT attack.

Guessing entropy (GE) is a widely adopted metric that measures the average computational cost needed for a successful side-channel analysis (SCA). GE relates to the number of trials that the attacker has to make before finding the actual secret key, it is defined as the average rank of the correct key among all key candidates after analysis of a given number of power traces. Fig. 5 reports GE for each of the 15 correct sub-key guesses on the c880 benchmark circuit including 15 2x4-inputs sub-comparators. As the number of analyzed power traces increases, GE of correct key guesses evolves, most of the time to be finally ranked at the first position among all possible key guesses in each sub-comparator.

Because some power traces are less relevant than others to provide information on the actual secret keys, these keys may still be incorrectly ranked even after a large number of power traces analysis. Dashed orange, red and purple curves in Fig. 5 are examples of correct sub-key guesses not correctly ranked by GE metric even after 16k power traces acquisition and analysis. They are respectively ranked in 14th, 9th and 8th positions by the end of the time-limited analysis. Other curves show that correct key guesses are ranked to the first position for the 12 remaining sub-comparators, leading to 80% (12 over 15) of sub-key recovery. The brown dashed curve shows for instance that the correct sub-key is rapidly identified as the best hypothesis for the corresponding sub-comparator after only 300 traces acquisition, and this ranking is maintained through the end of the experiments.

V. Conclusion

We propose a new framework to run DPA attacks against circuits locked with one of the most efficient logic locking technique, SFLL-HD^0. This technique is reputed resistant to SAT and DPA attacks for the same reason, that is the low corruption of the circuit behavior when controlled from a wrong key. If equivalence of both attacks remains true when attacking the same part of the circuit (logic cones of primary outputs in both cases), we demonstrate that targeting a sub-part of the design also related to the secret key (intermediate RU bits) allows to retrieve a majority of logic locking key bits thanks to DPA. Future works will focus on other LL schemes such as SFLL-HD^h and other most advanced LL techniques in order to evaluate the proposed framework on LL designs where the SAT attack is more or less effective.

References

[1] H. M. Kamali, K. Z. Azar, F. Farahmandi, et M. Tehranipoor, Advances in Logic Locking: Past, Present, and Prospects, p. 39.

[2] Jarvis R, McIntyre M, Split manufacturing method for advanced semiconductor circuits, US Patent 7,195,931

[3] M. Li et al., Provably Secure Camouflaging Strategy for IC Protection, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 8, pp. 1399-1412, Aug. 2019.

[4] J. A. Roy, F. Koushanfar and I. L. Markov, EPIC: Ending Piracy of Integrated Circuits, 2008 Design, Automation and Test in Europe, Munich, Germany, 2008, pp. 1069-1074

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

[6] P. Subramanyan, S. Ray, et S. Malik, Evaluating the security of logic encryption algorithms, in 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA: IEEE, mai 2015, p. 137-143..

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

[8] 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, vol. 38, no. 2, pp. 199-207, Feb. 2019

[9] M. Yasin, B. Mazumdar, O. Sinanoglu, et J. Rajendran, Removal Attacks on Logic Locking and Camouflaging Techniques, IEEE Trans. Emerg. Topics Comput., vol. 8, n 2, p. 517-532, avr. 2020, doi: 10.1109/TETC.2017.2740364.

[10] M. Yasin, B. Mazumdar, J. J. V. Rajendran and O. Sinanoglu, "TTLock: Tenacious and traceless logic locking," 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Mclean, VA, USA, 2017, pp 166-166.

[11] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. V. Rajendran, O. Sinanoglu, Provably-Secure Logic Locking: From Theory To Practice, in ACM SIGSAC Conference on Computer and Communications Security, Dallas Texas USA: ACM, oct. 2017, p. 1601-1618.

[12] A. Sengupta, B. Mazumdar, M. Yasin, et O. Sinanoglu, Logic Locking With Provable Security Against Power Analysis Attacks, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 39, n 4, p. 766-778, avr. 2020

[13] Sisejkovic, F. Merchant, L. M. Reimann, H. Srivastava, A. Hallawa, et R. Leupers, « Challenging the Security of Logic Locking Schemes in the Era of Deep Learning: A Neuroevolutionary Approach », J. Emerg. Technol. Comput. Syst., vol. 17, no 3, p. 1-26, juill. 2021, doi: 10.1145/3431389.

[14] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in Advances in Cryptology CRYPTO. Heidelberg, Germany: Springer, 1999, pp. 388–397

[15] M. Yasin, A. Sengupta, B.C. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, What to Lock?: Functional and Parametric Locking. In Great Lakes Symposium on VLSI, 2017, pp.351–3

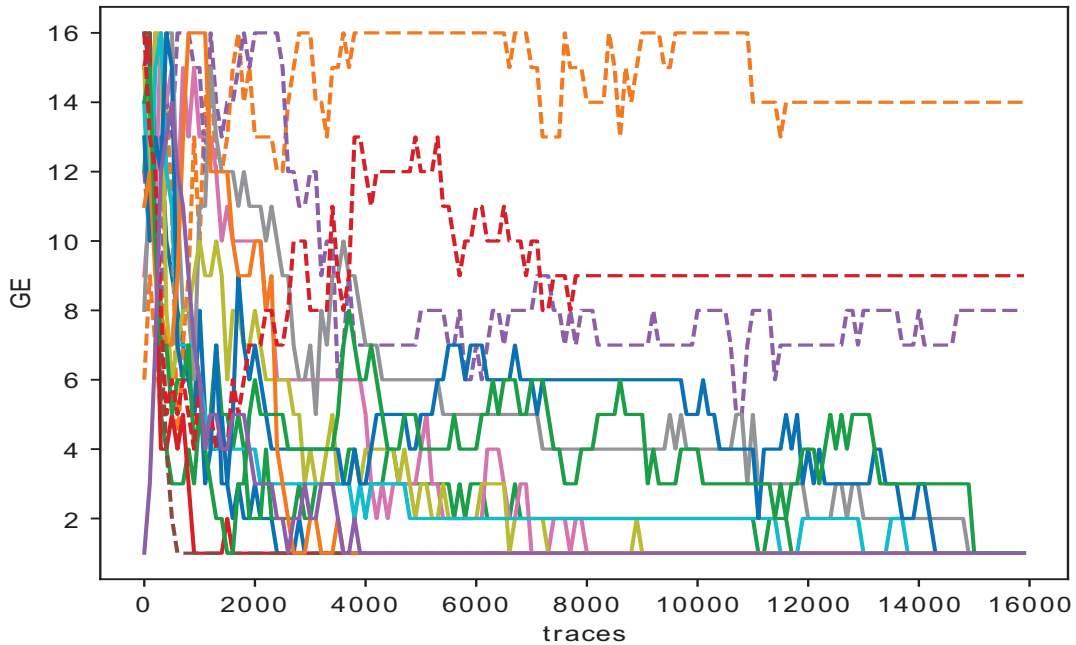

Fig. 5. Evolving GE of the 15 correct sub-key guesses for c880 circuit over 16k traces.