Towards Enhancing Power-Analysis Attack Resilience for Logic Locking Techniques

Zhiming Zhang[1], Ivan Miketic[2], Emre Salman[2], and Qiaoyan Yu[1]

[1]Department of Electrical and Computer Engineering, University of New Hampshire
[2]Department of Electrical and Computer Engineering, Stony Brook University

***Abstract*—Logic locking techniques have been widely investigated to thwart intellectual property (IP) piracy and reverse engineering attacks on integrated circuits. Although extensive research efforts have been made to examine the resilience of logic locking techniques against Boolean satisfiability (SAT) and key sensitization attacks, there still lacks a comprehensive assessment of different locking methods' resilience against power-analysis attacks. In this work, we evaluate the success rate of differential power analysis (DPA) and correlation power analysis (CPA) attacks that are performed on the circuits encrypted with logic locking techniques applied at gate level or transistor level. To enhance the CPA attack resilience of the existing transistor-level locking techniques, we further propose a new strategy to search for optimal key insertion locations. Our analysis and experimental results indicate that gate-level locking and transistor-level locking should use different strategies to select the optimal key insertion locations. Our case studies confirm that the proposed key insertion strategy can improve the transistor-level locking technique's resilience against CPA attacks.**

***Index Terms*—Power-analysis attack, logic locking, key retrieval speed, guessing entropy, side-channel attack.**

I. Introduction

Logic locking is a commonly used protection mechanism to thwart the intellectual property (IP) piracy attacks happened in the untrusted semiconductor supply chain. Logic locking techniques insert key-controlled locking units in the circuits under protection so that the locked circuits only function normally when the correct key is applied. The secret locking key is crucial for the success of logic locking mechanisms. Unfortunately, the literature has demonstrated that the locking keys can be retrieved by key sensitization attack [1] and Boolean satisfiability (SAT) attack [2], which are based on the functional testing and analysis. Logic locking techniques have been improved accordingly to address the vulnerabilities identified by those attacks. However, the resilience of logic locking techniques against power-based side-channel attacks (SCAs) have not been broadly investigated.

To fill this gap, this work performs a comprehensive evaluation on the attack resilience of logic locking techniques against two power-analysis attacks: differential power analysis (DPA) and correlation power analysis (CPA) attacks. More specifically, this work makes the following contributions:

- We perform theoretical analyses on the vulnerability of both gate-level and transistor-level logic locking techniques in DPA and CPA attacks.

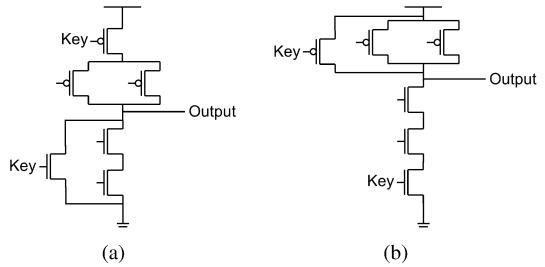


Fig. 1: A NAND gate locked with the transistor-level logic locking in (a) PSLNPL and (b) PPLNSL configurations.

- We conduct FPGA emulation and transistor-level simulation to assess the key recovery rate (KRR) [3], guessing entropy, and execution time of DPA and CPA attacks on several ISCAS benchmark circuits protected with three locking methods.
- In our case studies, we observe that the optimal key insertion locations for gate-level logic locking are different with those for transistor-level locking. We further propose a new strategy to search for suitable key insertion locations for transistor-level logic locking.

II. Related Work

Gate-level logic locking [4] inserts key-controlled gates to the original design that needs to be protected. The specific implementation of gate-level logic locking varies with the locking goal, such as obtaining higher output corruptibility or better resilience against key-retrieval attacks. For example, the fault analysis-based logic locking (FLL) inserts the key-controlled gates to the locations, which have the highest fault impacts on achieving the maximum output corruptibility [5]. A strong inference-based logic locking (SLL) is proposed in [1] to thwart the key sensitization attack. The work [6] uses multiplexers, instead of XOR/XNOR gates, to expand the logic cone size and thus improve the defense capability against cone-based brute-force attacks.

Transistor-level logic locking modifies the conventional logic gates internally by injecting the key-controlled transistors. The transistor-level logic locking can be configured in various ways to achieve different goals. For example, the locking method introduced in [7] protects the IP security of monolithic 3D ICs by independently inserting parallel or serial locking transistors and camouflaged contacts in multiple tiers. Its follow-up configuration shown in Fig. 1 suggests that the

key-controlled transistors can be inserted into a logic gate with two styles: PMOS serial locking plus NMOS parallel locking (*PSLNPL*) and PMOS parallel locking plus NMOS serial locking (*PPLNSL*). The wrong key in the PSLNPL style will lead the NAND gate output to be a constant 0. Likewise in the configuration of PPLNSL, the wrong key will yield a constant 1 at the output of NAND. Multiple key-controlled transistors are used to lock one single logic gate in [8] so that the key-interference achieved by locking can thwart the key sensitization attack with minor hardware cost. The work [9] substitutes the conventional CMOS transistors with silicon nanowire based field effect transistors, where a locking key bit controls the logic output of the modified gate via the polarity gate, to reduce the overhead induced by logic locking.

There is limited work discussing the DPA and CPA attack resilience of logic locking techniques. In the existing literature, only DPA attacks are performed on gate-level logic locking [3] but CPA attacks have not been examined in the context of logic locking techniques. This work provides a comprehensive analysis of DPA and CPA attacks on both gate- and transistor-level logic locking and then proposes possible ways to revise the locking configuration to improve its attack resilience.

III. Theoretical Basis of DPA and CPA

DPA and CPA attacks retrieve the crypto key from the hardware implementation of encryption systems. Both attacks leverage the correlation between the crypto key and the switching activities of the crypto hardware module to significantly shorten the time spent on key guessing compared to brute force attacks.

DPA attacks [10] have drawn significant attention over the last two decades. DPA exploits the fact that the power consumption of a chip is correlated with its internal data switching to retrieve the secret key applied in the crypto hardware module. DPA attackers need to collect power traces from their target chip that runs the encryption algorithm with an unknown secret key and a set of known plaintexts. The same plaintexts will be used to calculate the outputs of encryption with different guessing keys. Next, the collected power traces and the calculated outputs will be utilized to guess the correct key applied in the crypto module. The key guessing process relies on a statistical metric *difference of means (DoM)*.

Let's denote the target encryption process as $\mathcal{E}(p, k)$, in which $\mathcal{E}$ stands for the encryption algorithm and the variables p and k represent a plaintext and an encryption key, respectively. A ciphertext c is the output of $\mathcal{E}(p, k)$. Attackers first randomly guess a key, k_{guess} and then calculate a set of ciphertexts, C, with regards to a group of plaintexts P. The same process is repeated for different k_{guess} while P remains the same. The same P will be applied to the real chip to produce the corresponding power traces T. Each guessing key k_{guess} will have a specific C for P. In DPA attacks, depending on whether each calculated element c is 0 or 1, the corresponding power trace t in T for the current p is first classified to one of the two subsets $T0$ and $T1$. Next, the DoM defined in Eq. (1) is calculated for each guessing key k_{guess}. The DPA attack will consider the k_{guess} that yields the highest DoM as the correct key.

$$DoM = |\overline{T0} - \overline{T1}| \tag{1}$$

Where $\overline{T0}$ and $\overline{T1}$ are the averages of $T0$ and $T1$, respectively.

The experimental setup for CPA and DPA attacks is the same. However, the statistical analysis approach employed in CPA is different. In CPA attacks, attackers adopt a power estimation model, either Hamming distance or Hamming weight [11], to generate hypothetical power consumption [11] and then use the metric Pearson Correlation Coefficient (PCC) [12] to retrieve the secret key. A hypothetical power consumption T_{hyp} is estimated based on the set C for each k_{guess} using the power estimation model. Equation (2) describes how the PCC is calculated.

$$PCC = \frac{E[T_{hyp} \cdot T] - E[T_{hyp}] \cdot E[T]}{\sqrt{E[T_{hyp}^2] - (E[T_{hyp}])^2} \cdot \sqrt{E[T^2] - (E[T])^2}} \tag{2}$$

Each guessing key k_{guess} will have its own PCC. The k_{guess} that has the highest PCC will be considered as the correct key by the CPA attack.

In the next section, we use the two metrics, DoM and PCC, to analyze the DPA and CPA attack resilience of logic locking applied at gate and transistor levels.

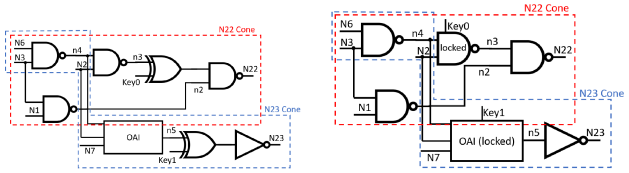
IV. DPA and CPA Attack Resilience of Gate-Level and Transistor-Level Logic Locking

A. Resilience against DPA Attack

We performed the DPA attack on an ISCAS'85 benchmark circuit, c17. As the circuit c17 has two output ports *N22* and *N23*, there are two logic cones highlighted by the two dash-line boxes shown in Fig 2. The c17 locked by XOR-based gate locking is shown in Fig. 2(a). We also applied PSLNPL and PPLNSL to the NAND and OR-AND-INVERT (OAI) logic gates in c17, as shown in Fig. 2(b). The detailed experimental setup is described in Section VI-A.

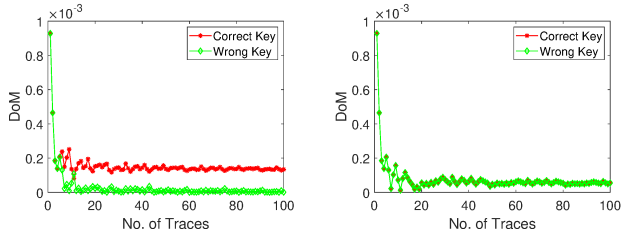
The DoM measured by the DPA attack on c17 protected with three logic locking methods are reported in Figs. 3, 4 and 5. For the N22 cone, Fig. 3(a) shows that the DoM line for the correct key is above that for the wrong key, which indicates that the locking key bit applied in the N22 cone can be retrieved by the DPA attack. For the N23 cone, as shown in Fig. 3(b), the DoM lines for the correct and wrong keys are overlapped, which means that the DPA attack does not find the correct key. Overall, Fig. 3 confirms that gate-level logic locking has 50% resilience against the DPA attack. Based on the measured DoM metrics for PSLNPL and PPLNSL transistor-level locking shown in Figs. 4 and 5, we conclude that the DPA attack fails to retrieve the locking key bits in 75% of the test cases.

According to the analysis in the previous section, the wrong locking key at the transistor-level locking will lead to a constant output. This characteristic could form a natural defense line to thwart the DPA attack. When the constant output induced by the wrong key is fed to another logic gate



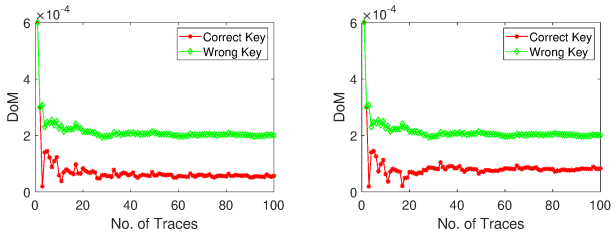
(a) (b)

Fig. 2: c17 protected with (a) XOR-based gate-level and (b) transistor-level logic locking.



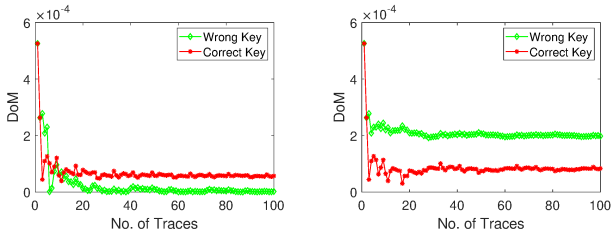
(a) (b)

Fig. 3: DoM for (a) N22 cone and (b) N23 cone in c17 locked with XOR-based gate-level locking.



(a) (b)

Fig. 4: DoM for (a) N22 cone and (b) N23 cone in c17 locked with PSLNPL.



(a) (b)

Fig. 5: DoM for (a) N22 cone and (b) N23 cone in c17 locked with PPLNSL.

as a controlling bit (e.g., constant 1 to OR gate), the output of the subsequent gate will be constant, too. If more key bits are inserted in the circuit, the probability of propagating the constant output to the primary output is likely to increase. Since the wrong key guess leads the primary output to be a constant 1 (0), all the power traces will be grouped into the power set $T1$ ($T0$). Consequently, the wrong key guess will yield a higher DoM than the correct key, and thus the DPA attack will conclude a wrong key. In summary, once the wrong key causes the primary output of the locked netlist to be constant, the DoM metric will mislead the DPA key retrieval.

We studied the three DPA failed cases shown in Figs. 4(a) and (b) and Fig. 5(b) and observed that the primary outputs in those cases are indeed constant regardless of what primary inputs were provided. For example, if the guessed Key0 is wrong in the PSLNPL configuration, the output of the locked NAND in Fig. 2(b) will be constant 0, which further causes N22 to be constant 1, as shown in Fig. 6. In this case, the locking key obtained by the DPA attack is wrong because all the power traces are grouped to $T1$. Due to the same reason (but different constant outputs of the locked gates), the other two cases represented by Figs. 4(b) and 5(b) also fail to retrieve the correct locking keys.

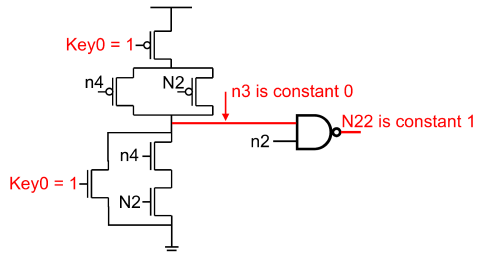
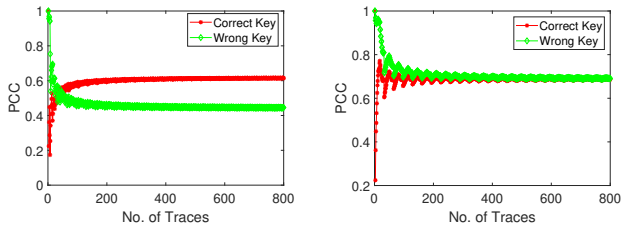


Fig. 6: The impact of transistor-level locking on the output.

B. Resilience against CPA Attack

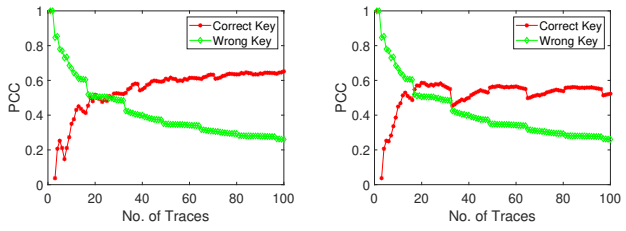
The CPA attack was executed on the same c17 circuit for both the XOR-based gate-level locking and the transistor-level locking. The metric PCC was utilized to differentiate the correct key from the wrong ones. As shown in Fig. 7, for the N22 cone, the PCC of the correct key case is much higher than the PCC of the wrong key; while for the N23 cone, the correct and incorrect key guesses lead to comparable PCCs after the initial vibration stage. This observation means, in the case of XOR-based locking, the CPA attack can successfully retrieve the key bit in the N22 cone but cannot retrieve the key bit in the N23 cone. In the case of the c17 locked with the transistor-level locking, the PCC for the correct key guess is higher than that for the wrong key guess after 40 power traces. This observation holds true for both N22 and N23 logic cones, as shown in Figs. 8 and 9. This means that the locking key bits in both cones can be successfully identified by the CPA attack. In the case of the N23 cone locked with the PPLNSL configuration, the estimated power consumption has no correlation with the real power traces. This is because the wrong Key1 shown in Fig. 2(b) leads to a constant 0 on the output of the N23 logic cone. Based on the Hamming distance/Hamming weight model, the estimated power consumption is constant 0, too. According to Eq. (2), no valid PCC can be calculated. In summary, the CPA attack can retrieve all the locking key bits in the transistor-level locking cases but the DPA attack only partially recovers the key.

The experimental results above indicate that the PCC metric used in CPA attacks is not affected by the constant output caused by the wrong key guess at the transistor-level locking. Instead, the constant output facilities the CPA attack to succeed. This is because the estimated power consumption T_{hyp} will be constant once the estimated output C is constant due to the wrong key guess. Based on the definition of PCC, a



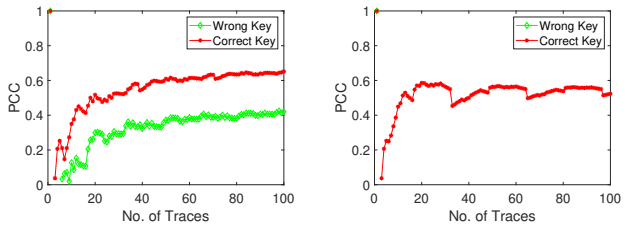
(a) (b)

Fig. 7: PCC for (a) N22 cone and (b) N23 cone in c17 locked with XOR-based gate-level locking.



(a) (b)

Fig. 8: PCC for (a) N22 cone and (b) N23 cone in c17 locked with PSLNPL.



(a) (b)

Fig. 9: PCC for (a) N22 cone and (b) N23 cone in c17 locked with PPLNSL.

constant sequence will have no correlation with the real power consumption T. As a result, the wrong key guess can be easily excluded by the CPA attack.

On the other hand, the CPA attack can be mitigated by the gate-level logic locking, as shown in Fig. 7(b). The XOR-based gate-level locking will lead the locked gate to produce a flipped output if a wrong key is applied. Once the wrong output is propagated to the primary output of the logic cone, the PCCs for the wrong and correct key cases will be the same, no matter which power model is employed in power estimation. If the Hamming distance model is used, T_{hyp} for the wrong key guess will be identical with the one for the correct key and so is PCC. For example, the original output sequence is $[10010]$ and the flipped sequence is $[01101]$. Then, the T_{hyp} based on the Hamming distance model is $[1011]$ for both sequences. If the Hamming weight model is adopted, T_{hyp} for the wrong key guess will toggle oppositely to the one for the correct key guess. Although this flipped T_{hyp} for the wrong key results in a reversed PCC, the $|\text{PCC}|$ is still the same with the one for the correct key guess. Consequently, the CPA attack cannot differentiate the wrong key from the correct key for either case. We zoomed in the failed CPA case in the c17 locked with the XOR-based locking and found that its primary output was indeed flipped when a wrong key was given. However, it is not always possible to propagate the constant output in bigger circuits. The CPA resilience provided by the gate-level locking only happens in rare cases.

V. DPA and CPA Attack Resilience Enhancement

In this section, we summarize the existing effort that investigates the resilience enhancement against DPA and CPA attacks and propose a new strategy that facilitates to search for better key insertion locations for defending CPA attacks.

The work [3] evaluates the DPA resilience of gate-level logic locking techniques. That work also provides two suggestions to harden the locking circuit against DPA attacks: (1) increase the ratio of key bits to the number of primary inputs of the logic cone, and (2) insert key bits in a way that the locked circuit functions closely to the original circuit even when a wrong key is given.

To the best of our knowledge, there is no prior work available discussing how to enhance the transistor-level logic locking with respect to the CPA attack resilience. To fill this gap, we propose a new guideline (composed of three rules) for the optimal key insertion locations in PSLNPL and PPLNSL based transistor-level locking configuration.

- **Rule 1:** *Avoid inserting a key bit to a gate, whose wrong constant output can be propagated to the primary outputs of the locked circuits.*
- **Rule 2:** *Use the PSLNPL configuration to lock the gates that have logic 0 as their majority output (e.g., AND and NOR gates).*
- **Rule 3:** *Use the PPLNSL configuration to lock the gates that have logic 1 as their majority output (e.g., OR and NAND gates).*

As we observed in Section IV-B, the wrong key induced constant primary output will result in an invalid PCC in the CPA algorithm and thus those wrong key guesses can be easily eliminated from the attack process. The proposed rule 1 defers the quick key elimination. In some cases, the primary output may be reversely constant (e.g., logic 1 at the primary output but logic 0 at the gate output), which should be avoided, too.

Output corruptibility is a classic metric evaluating the ability of logic locking techniques in altering the original logic function when wrong keys are given. Usually, a higher output corruptibility will provide a better defense to IP piracy attacks or counterfeiting. However, a lower output corruptibility is more favorable in the sense of thwarting the CPA attack. We denote the difference between the PCC values for a wrong key and a correct key as $DIFF_{PCC}$. As the CPA attack retrieves the correct key by searching for the key yielding the highest PCC, we suggest exploring countermeasures against the CPA attack that can minimize $DIFF_{PCC}$. A method that lowers the output corruptibility helps to achieve a smaller $DIFF_{PCC}$ and obtain a better CPA attack resilience.

Inspired on the relation between the output corruptibility and the CPA resilience, the proposed rules 2 and 3 will enable the transistor-level logic locking to reduce the output

corruptibility. We use an NAND gate as an example to explain the utilization of the proposed rules 2 and 3. The majority of the NAND output is logic 1. When a wrong key is given, PSLNPL (PPLNSL) will force the output of the locked gate to be a constant 0(1). In this case, the wrong key of a PPLNSL locked NAND gate may not change the original logic output of the locked circuit as much as that of the PSLNPL configuration. This is because the NAND gate outputs logic 1 for most time (if its input 0/1 is uniformly distributed). It is reasonable to infer that the PCC for the wrong key in PPLNSL is closer to that for the correct key compared to the scenario of PSLNPL. As a result, we conclude that $DIFF_{PCC}(PPLNSL) \lessapprox DIFF_{PCC}(PSLNPL)$. Thus, it is more difficult for CPA to differentiate the correct key from the wrong keys in the PPLNSL configuration than in the case of PSLNPL configuration.

VI. Comprehensive Evaluation

A. Experimental Setup

We performed various experiments to evaluate the CPA resilience of the XOR-based gate-level and the PSLNPL and PPLNSL based transistor-level locking techniques using the following setup. The three logic locking methods were applied to the ISCAS'85 benchmark circuits, including c432, c2670 and c3540. The HOPE simulator [13] was adopted to execute the FLL strategy [5] for key bit insertion. The CPA attack was performed by FPGA emulations and transistor-level simulations in Cadence Virtuoso. In the FPGA emulation, the locked circuits were mapped to the SAKURA-G FPGA board and the power traces were collected through ChipWhisperer. In the transistor-level simulation, the locked circuits were implemented with the NCSU FreePDK45 technology.

B. Resilience against CPA Attack

The KRR metric is adopted to assess the success rate of the CPA attack. In this subsection, we examine the impact of locking level, key insertion location, and other factors on the attack resilience.

1) Impact of Locking Level on Attack Resilience: The key insertion locations for the XOR-based gate-level locking were determined by the FLL strategy recommended by the HOPE simulator. The key bits for the transistor-level locking were inserted to the same locations recognized by FLL. Due to the different numbers of logic gates in c432 and c2670, 8 and 16 key bits were used in the encryption, respectively. As shown in Table I, the PPLNSL transistor-level locking achieves better CPA resilience than the XOR-based gate-level locking in the case of c432; however, as the circuit scale increases, the XOR-based locking on c2670 outperforms both PSLNPL and PPLNSL, reducing the KRR by 66%. We further analyzed the guessing entropy to compare the key retrieval speed. As shown in Fig. 10, the guessing entropy of the CPA attack on c2670 protected with XOR-based locking is always higher than that for the same circuit encrypted by the transistor-level locking. Both KRR and guessing entropy indicates that PSLNPL and PPLNSL transistor-level locking is more vulnerable to the CPA attack than XOR-based gate-level locking.

TABLE I: KRR results for CPA attacks on two locked benchmark circuits.

circuit \ locking configuration	XOR	PSLNPL	PPLNSL
c432	100%	100%	50%
c2670	18.75%	56.25%	56.25%

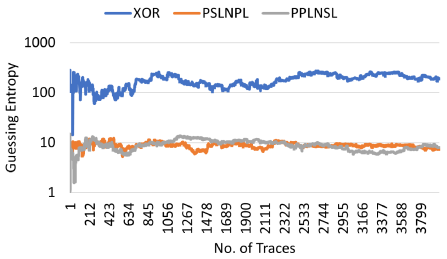


Fig. 10: Guessing entropy comparison for the case of c2670.

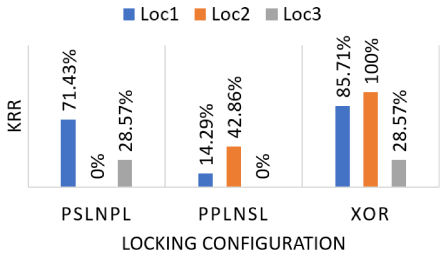


Fig. 11: The impact of random key insertion locations on KRR.

TABLE II: Impact of FLL and proposed key location selection strategy on KRR.

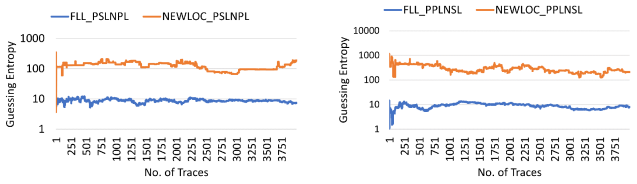
Key insertion strategy	Locking configuration	KRR
FLL	XOR	18.75%
	PSLNPL	56.25%
	PPLNSL	56.25%
Proposed Strategy	PSLNPL	18.75%
	PPLNSL	18.75%

2) Impact of Different Key Locations on Attack Resilience: We first randomly selected the key locations for the c432 locked with both the XOR-based locking and the transistor-level locking. The KRR results shown in Fig. 11 imply that the KRR has strong dependency on the locking location, no matter at gate level or transistor level. Next, we followed the proposed three rules for optimal key locations to lock c2670. Based on the comparison in Table II, our method reduces the KRR of the transistor-level locking to be the same with the KRR that the XOR-based gate-level locking obtains. As shown in Table III, the selected logic gates for the PSLNPL key insertion have a relatively high probability to have logic 0 as outputs. Likewise, the selected key gates for the PPLNSL configuration have a high probability to output logic 1. The FLL strategy for the XOR-based locking is not an option for the transistor-level locking. The constant gate outputs caused by wrong keys could be propagated to the primary outputs, which facilitate the CPA attack.

Furthermore, the guessing entropy for the transistor-level locking configured following the FLL and the proposed key

TABLE III: Key locations following the proposed strategy.

Locking configuration	Key insertion
PSLNPL	7 AND5, 8 AND4, 1 AND3
PPLNSL	2 OR5, 12 OR4, 2 OR3



(a) (b)

Fig. 12: Guessing entropy comparison for (a) PSLNPL and (b) PPLNSL configurations.

TABLE IV: Comparison of output corruptibility.

Key inserting strategy	Locking configuration	1 bit wrong	5 bits wrong	All bits wrong
FLL	XOR	1.90%	6.99%	10.24%
	PSLNPL	0.98%	4.23%	8.00%
	PPLNSL	0.93%	3.95%	7.89%
Proposed strategy	PSLNPL	0.11%	0.32%	0.62%
	PPLNSL	0.48%	1.58%	3.35%

location is compared in Fig. 12. NEWLOC_PSLNPL and NEWLOC_PPLNSL represent the PSLNPL and PPLNSL locking configured with our proposed locking locations. As can be seen, our method improves the guessing entropy by 25.36× and 26.04× for PSLNPL and PPLNSL, respectively, based on the 4000 power traces.

The output corruptibility of the XOR-based gate-level locking and transistor-level locking is compared in Table IV. We measured the output corruptibility for the cases of 1 bit, 5 bits and all bits wrong. For the cases of 1 bit wrong, we swept the wrong key bit for all 16 locations and reported the averaged output corruptibility. For the cases of 5 bits wrong, we randomly selected the wrong key bit locations four times and presented the averaged result of the four corresponding output corruptibility. As shown in Table IV, if the FLL strategy is applied, the transistor-level locking has the comparable output corruptibility with the XOR-based locking in each test case. However, the proposed new locking location strategy can significantly reduce the output corruptibility, which is consistent with the guessing entropy trend shown in Fig. 12.

3) Key Retrieval Rate and Speed in Large Circuit: We also evaluated the CPA resilience for the locking configurations in a larger benchmark circuit, c3540. We used FLL to configure the XOR-based locking and the proposed strategy to configure the PSLNPL and PPLNSL locking. 32 key bits were inserted for each locking. We define 1 day as the time limit so that the logic cone in which the CPA attack takes more than 24 hours is considered as no key bit will be retrieved. The KRR results based on 4000 power traces are shown in Table V. The CPA attack obtains a KRR of 6.25% for the case of XOR-based locking. However, the CPA attack cannot retrieve any key bit in the cases of PSLNPL and PPLNSL locking. Based on the fact that one iteration takes around 1 second in our CPA algorithm ran on a computer at 1.8GHz and with 8GB memory, we estimate the averaged execution time that the CPA attack will take on one logic cone. The comparison of the averaged execution time is listed in Table V. Interestingly, a lower KRR does not necessarily represent a higher attack effort, as the NEWLOC_PSLNPL case actually consumes less iterations and execution time than the FLL_XOR case.

TABLE V: Cone-based CPA attack effort and its KRR.

	FLL_XOR	NEWLOC_PSLNPL	NEWLOC_PPLNSL
KRR	6.25%	0%	0%
Iteration	1300	1032	6667
Execution time	0.36 hours	0.29 hours	1.85 hours

VII. Conclusion

This work performs a comprehensive analysis and a set of experiments to assess the gate-level and the transistor-level logic locking techniques' resilience against DPA and CPA attacks. We observe that the key recovery rate of CPA attack has a strong dependence on the locking key locations, no matter at gate level or transistor level. However, the optimal key locations for the gate-level locking does not guarantee the transistor-level locking to obtain the best CPA attack resilience. Different strategies for the best key insertion locations should be developed. In this work, we propose three rules for the existing transistor-level locking techniques. Our experimental results show that our method improves the guessing entropy by 25.36× and 26.04× for PSLNPL and PPLNSL, respectively, over the FLL based key insertion location, and our method also can mitigate the CPA attack successfully.

References

[1] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE TCAD*, vol. 35, no. 9, pp. 1411–1424, 2016.

[2] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. HOST'15*, pp. 137–143, 2015.

[3] A. Sengupta, B. Mazumdar, M. Yasin, and O. Sinanoglu, "Logic locking with provable security against power analysis attacks," *IEEE TCAD*, vol. 39, no. 4, pp. 766–778, 2020.

[4] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proc. DATE'08*, pp. 1069–1074, 2008.

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

[6] Y. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Proc. LATS'15*, pp. 1–6, 2015.

[7] J. Dofe, Chen Yan, S. Kontak, E. Salman, and Q. Yu, "Transistor-level camouflaged logic locking method for monolithic 3d ic security," in *Proc. AsianHOST'16*, pp. 1–6, 2016.

[8] V. S. Rathor and G. K. Sharma, "A lightweight robust logic locking technique to thwart sensitization and cone based attacks," *IEEE Trans. Emerg. Topics Comput.*, pp. 1–1, 2019.

[9] Q. Alasad, J.-S. Yuan, and Y. Bi, "Logic locking using hybrid cmos and emerging sinw fets," *Electronics*, vol. 6, no. 3, 2017.

[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO'99* (M. Wiener, ed.), pp. 388–397, 1999.

[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. CHES'04*, pp. 16–29, 2004.

[12] O. X. Standaert, E. Peeters, G. Rouvroy, and J. J. Quisquater, "An overview of power analysis attacks against field programmable gate arrays," *Proceedings of the IEEE*, vol. 94, pp. 383–394, Feb 2006.

[13] Hyung Ki Lee and Dong Sam Ha, "Hope: an efficient parallel fault simulator for synchronous sequential circuits," *IEEE TCAD*, vol. 15, no. 9, pp. 1048–1058, 1996.