Assessing Correlation Power Analysis (CPA) Attack Resilience of Transistor-Level Logic Locking

Zhiming Zhang
University of New Hampshire
Durham, NH, USA

Ivan Miketic
Stony Brook University
Stony Brook, NY, USA

Emre Salman
Stony Brook University
Stony Brook, NY, USA

Qiaoyan Yu
University of New Hampshire
Durham, NH, USA

ABSTRACT

Logic locking has demonstrated its potential to protect the intellectual property of integrated circuits (ICs). The security strength of logic locking is typically evaluated through functional and structural analysis-based attacks. There is limited work analyzing logic locking techniques' resilience against power-based side-channel attacks. To fill this gap, we propose an attack flow for the correlation power analysis (CPA) attack on the circuits encrypted with transistor-level logic locking. Our case studies indicate that CPA attacks outperform DPA attacks in terms of key recovery rate (KRR). To improve the CPA attack resilience of an existing transistor-level logic locking technique, we propose a logic-cone conjunction (LCC) method to enlarge the key space and reduce the correlation between the locking key and the power consumption of locked circuits. The experimental results show that the LCC method successfully reduces the KRR from 100% to 0% by using cyclic logic structures. The FPGA emulation indicates that the proposed method incurs 2.6% more delay and 1.5% more power consumption than the baseline.

CCS CONCEPTS

• **Security and privacy → Key management**; **Side-channel analysis and countermeasures**; • **Hardware → Transistors**.

KEYWORDS

Logic locking, CPA, DPA, guessing entropy, attack resilience.

ACM Reference Format:
Zhiming Zhang, Ivan Miketic, Emre Salman, and Qiaoyan Yu. 2021. Assessing Correlation Power Analysis (CPA) Attack Resilience of Transistor-Level Logic Locking. In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461508

1 INTRODUCTION

Outsourcing modern integrated circuit (IC) manufacturing brings security threats to the chip supply chain [12]. Untrusted IC foundries having access to design source files could tamper with or reverse engineer the original netlist. Various logic locking-based countermeasures [2, 5, 9, 12] are used to mitigate intellectual property (IP) piracy attacks. Logic locking techniques utilize key-controlled logic gates or transistors to encrypt the original design netlist.

Although logic locking has the potential to thwart IP piracy, the 'arm race' is still going on between the enhancement of locking techniques and advanced attacks, such as Boolean Satisfiability (SAT) based attacks, sensitization attacks, and key removal attacks [6, 10, 13]. Most existing efforts focus on improving the locking algorithms to thwart the attacks mentioned above. There is limited work quantitatively assessing the resilience of logic locking techniques against power-based side-channel analysis attacks. The work [9] performs differential power analysis (DPA) attacks on gate-level logic locking and concludes that the locking technique has a natural defense capability against DPA attacks. However, that work does not evaluate the logic locking technique in the context of the correlation power analysis (CPA) attack, which is more advanced than DPA. To facilitate the logic locking techniques to advance further, we make the following contributions in this work:

- We expand the security analysis and quantitative assessment from gate-level to transistor-level locking techniques.
- We propose a practical attack flow to enable the CPA attack on the ICs protected with logic locking techniques. We further compare the key recovery rate (KRR) of the proposed CPA attack with that of the DPA attack introduced in [9].
- A logic-cone conjunction (LCC) is proposed to strengthen the resilience of transistor-level locking against CPA attacks.

2 RELATED WORK

2.1 Gate- and Transistor-Level Logic Locking

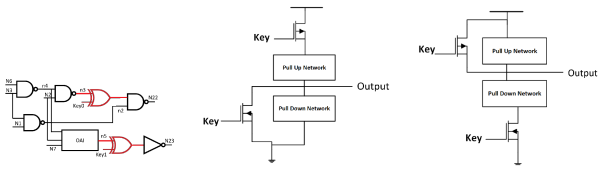
Gate-level logic locking techniques shown in Fig. 1(a) insert key gates to the original netlist so that the nets that the key gates control will be altered if incorrect locking keys are applied [7, 9]. The transistor-level logic locking method introduced in the work [2] locks the pull-up and pull-down networks with PMOS and NMOS transistors, respectively. A simplified version of that transistor-level locking is depicted in Figs. 1(b) and (c). As shown, that transistor-level logic locking has two configurations: PMOS serial locking plus NMOS parallel locking (*PSLNPL*) and PMOS parallel locking plus NMOS serial locking (*PPLNSL*). If a wrong key is applied to the circuit locked with PSLNPL, the PMOS locking transistor is turned off and the active NMOS locking transistor pulls the output of the locked gate down to the ground. As a result, the wrong key leads the gate output to be a constant 0. If the circuit is locked with PPLNSL, the wrong key yields a constant 1 at the output.

2.2 Existing Attacks on Logic Locking

The existing attacks on logic locking fall into two categories: functional or structural analysis-based attacks. The methods in the

GLSVLSI '21, June 22–25, 2021, Virtual Event, USA.

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461508



(a) (b) (c)

Figure 1: Locking techniques at gate and transistor levels. (a) XOR-based gate-level locking, and transistor-level locking with (b) PSLNPL and (c) PPLNSL configurations.

former category use the carefully designed input patterns and key guesses to derive the locking keys [11]. Structural analysis based attacks first remove the locking units and then recover the original design. For example, the work [1] applies machine learning to predict the structural changes made by IC synthesis tools. The existing functional and structural attacks on logic locking have their limitations. Functional attacks often involve high computing complexity and the attack efficiency heavily relies on the specific targets. Structural attacks are not applicable in the locking schemes that do not change the gate-level netlist. Side-channel attacks exploit the security vulnerabilities of the hardware implementation to break most of encryption systems. Current DPA and CPA attacks leverage the data-dependency between the power consumption and the internal signal switching of the target encryption system to retrieve the secret key. However, there is limited work available to investigate DPA and CPA attacks on logic locking techniques. To enhance the security of logic locking, it is imperative to conduct a comprehensive evaluation of the DPA and CPA attack resilience of various logic locking techniques.

3 ANALYSIS AND ASSESSMENT ON DPA ATTACK RESILIENCE OF TRANSISTOR-LEVEL LOGIC LOCKING

We performed DPA attacks on an ISCAS'85 benchmark circuit, c17. As c17 has two output ports *N22* and *N23*, there are two logic cones highlighted in the two dash-line boxes shown in Fig 2. The two transistor-level locking configurations, PSLNPL and PPLNSL, were applied to the NAND and OR-AND-INVERT (OAI) gates. The difference of means (DoM) [3, 9] measured in the process of DPA attacks was used in the key retrieval. Since the target circuit consumes distinct power when it generates the final output 1 and 0, the DPA attack selects the key that yields the highest DoM as the correct key. As shown in Figs. 3 and 4, the DoM for the wrong key guess is higher than that for the correct key guess in most of the cases (the only exception is the N22 cone locked by PPLNSL). Thus, we conclude that the DPA attack fails to retrieve the locking keys.

We zoomed in the three failed cases in the DPA experiments for c17 and found that their primary outputs were constant regardless of what primary inputs were provided. When the PSLNPL configuration is used in locking, the real key for both Key0 and Key1 is 0. If the guessed Key0 is wrong, the output of the locked NAND will be constant 0. That causes N22 to be constant 1, as shown in Fig. 5(a). If the guessed Key1 is incorrect, the net *n5* will be constant 0 and N23 will be constant 1, as shown in Fig. 5(b). In either case, the key retrieved by the DPA attack is wrong because all the power traces

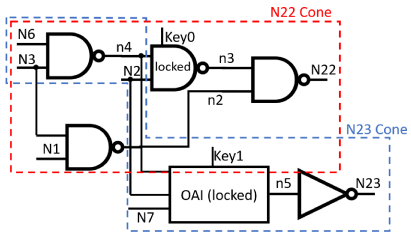
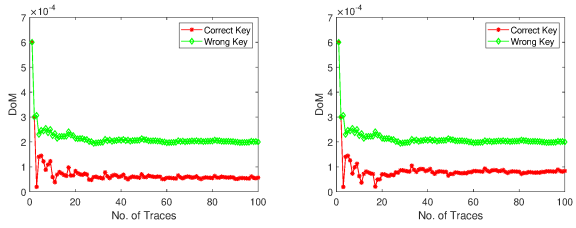
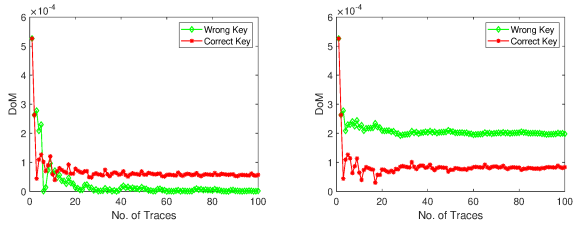


Figure 2: c17 with transistor-level logic locking.



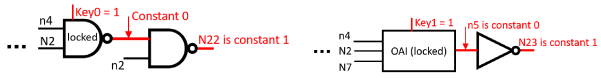
(a) (b)

Figure 3: DoM for (a) N22 cone and (b) N23 cone in c17 locked with PSLNPL.



(a) (b)

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



(a) (b)

Figure 5: The impact of transistor-level locking key bit in the PSLNPL configuration on (a) N22 cone and (b) N23 cone.

are grouped to the set that stores the power traces for the scenarios of output 1. The constant output of the locking gate will lead to a similar DPA failure in the PPLNSL configuration. Our case study reveals that if the wrong key guess causes the primary output of the locked netlist to be constant, the DoM metric used in the DPA attack will mislead the key retrieval. In addition, transistor-level logic locking has some resilience against DPA attacks because of its constant output induced by a wrong key.

4 PROPOSED ATTACK FLOW FOR CPA ATTACKS ON LOCKED CIRCUITS

4.1 Attack Flow

CPA is more powerful than DPA in key retrieval because the Pearson Correlation Coefficient (PCC) metric used in CPA attacks will not be affected by the constant output feature. The hypothetical power consumption in CPA attacks is usually formed using the

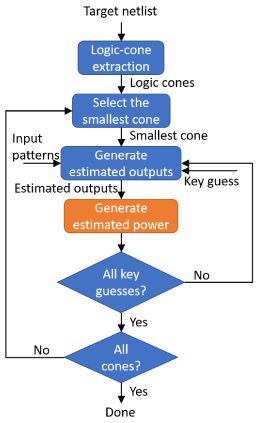


Figure 6: The proposed flow for the CPA attack on a general circuit protected by logic locking.

Algorithm 1: Proposed logic cone extraction.

Data: Locked netlist
Result: Logic function of each logic cone
$PrimaryOut[] \leftarrow$ Find primary outputs;
$PrimaryIn[] \leftarrow$ Find primary inputs;
$Key[] \leftarrow$ Find key inputs;
$i = 1$;
while $i \leq length(PrimaryOut[])$ **do**
  $Target \leftarrow PrimaryOut[i]$;
  while $Target \notin PrimaryIn[]$ && $Target \notin Key[]$ **do**
    $SelFunc \leftarrow Target$;
    Search logic gate $G(Input, Target)$;
    Substitute $G(Input, Target)$ into $SelFunc$;
    $Target \leftarrow Input$;
  end
  return $SelFunc$;
  $i = i + 1$;
end

Hamming distance or Hamming weight model. No matter which power estimation model is used, the hypothetical power will be constant once the estimated output is constant. Based on the characteristic of PCC, a constant sequence will have no correlation with the real power consumption. Then, the wrong key guess can be easily excluded by CPA. Hence, it is necessary to evaluate the resilience of transistor-level logic locking against CPA attacks. In this work, inspired by the divide-and-conquer strategy of [9], we modify the conventional CPA attack flow for cryptosystems and propose a feasible general power estimation procedure. Our CPA attack on the transistor-level logic locking includes four steps.

Step 1: logic cone extraction. The flow of estimated power generation is depicted in Fig. 6. First, the logic cones of the locked netlist are extracted based on the primary outputs. To facilitate the logic cone extraction, we develop a Python script and Algorithm 1 shows the its pseudo-code. The script returns the logic function of each logic cone, which will be used as the selection function (*SelFunc*) in the CPA attack. Given a locked netlist, Algorithm 1 searches for the logic gate (G) that generates each primary output of the netlist. The inputs of G will be the target of the next search until all the new targets are either the primary inputs or the key inputs of the netlist. The located G during this process will form the final *SelFunc* for the primary out. This process is repeated for each primary output until all logic cones are completed.

Step 2: divide-and-conquer-based power estimation. The CPA attack starts from the smallest logic cone, which includes the least number of locking key bits. This ascending order is adopted for two main reasons. First, the ratio of the number of keys to the number of primary inputs (#Keys/#Primary Inputs) of a smaller cone is smaller, too. As a result, retrieving the keys in a smaller cone is easier than in a larger one [9]. Second, some keys may appear in multiple cones and the keys that have been previously retrieved in the smaller cones can be used in the attack of the current cone. In this case, the attack will be more likely to succeed since the number of unresolved keys is reduced. Next, a set of input patterns with a random key guess are fed to the extracted selection function (*SelFunc* in Step 1) of the cone and the estimated outputs for the key guess are calculated. We utilize a Hamming distance model to generate the estimated power. The same process will be repeated for all key guesses and all logic cones.

Step 3: power trace collection. Similarly, the real power trace collection starts from the smallest logic cone and follows the ascending order. For each cone, the same set of the input patterns used in the power estimation are applied to the chip under attack and the physical power consumption is collected. Since only the inputs of the cone currently under attack will be fed with the input patterns, the switching activities of other cones will be minimized and thus there is limited interference from other cones. In parallel with the power consumption measuring, the output patterns of the same cone under attack are recorded from the chip to verify the retrieved key values.

Step 4: correlation analysis. We calculate the PCC between the estimated power and the real power consumption to retrieve the keys of each logic cone. The key guess which yields the highest PCC is considered as the correct key retrieved by the CPA attack.

4.2 CPA Resilience Assessment of Transistor-level Locking

The proposed CPA attack flow was employed to perform a quantitative assessment on the transistor-level PSLNPL and PPLNSL locking [2]. Circuits c17, c432, and c880 were locked and implemented with a FreePDK45 technology [8] for transistor-level simulation and on a SAKURA-G FPGA board for hardware emulation. The power traces for those two platforms were measured in Cadence Virtuoso and the ChipWhipserer software, respectively.

In the transistor-level simulation of the c17 locked with PSLNPL configuration, we observe that 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. 7 and 8. This means that the locking keys in both cones can be successfully retrieved 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 because the wrong key1 leads to a constant 0 on the output of the N23

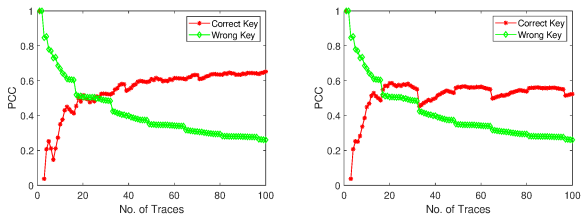


Figure 7: PCC for (a) N22 cone and (b) N23 cone in c17 locked with PSLNPL.

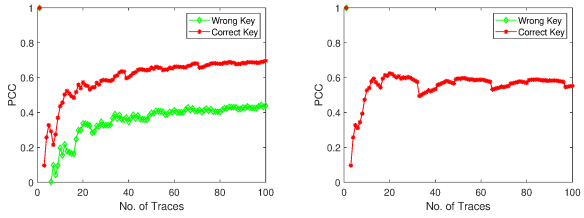


Figure 8: PCC for (a) N22 cone and (b) N23 cone in c17 locked with PPLNSL.

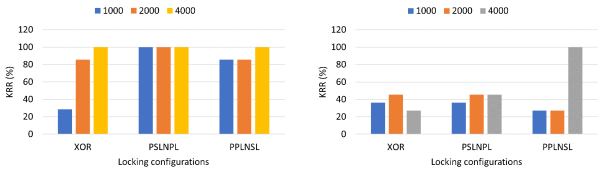


Figure 9: KRR results for (a) c432 (b) c880.

logic cone. Based on the Hamming distance model, the estimated power consumption is constant 0, too. Thus, no valid PCC can be calculated. In summary, our attack flow enables the CPA attack to retrieve all the locking key bits in c17. In contrast, the DPA attack only partially recovers the locking key.

Next, the key recovery rate (KRR) [9] defined in Eq. (1) is used to assess the efficiency of CPA attacks on the benchmark circuits protected with XOR-based gate-level logic locking and the transistor-level PSLNPL and PPLNSL logic locking. Due to the different circuit scale, 7 and 11 key bits were applied to c432 and c880, respectively.

$$KRR = \frac{No.\,Retrieved\,Key\,Bits}{No.\,Inserted\,Key\,Bits} \tag{1}$$

To accelerate the attack speed, we also examined the proposed CPA attack in an FPGA platform. The KRR of the CPA attack on c432 and c880 is shown in Fig. 9. With 4000 power traces, our CPA attack retrieved all the key bits for c432 no matter which locking configuration was used; for the bigger circuit c880, the CPA attack also achieved a 100% KRR in the PPLNSL configuration.

Furthermore, we swept the number of key bits inserted in c432 from 1 key bit per cone to 3 key bits per cone for both the gate-level and transistor-level logic locking techniques. As indicated in Fig. 10, given 800 power traces, all three locking methods achieve the KRR of 0% as the number of key bits increases. Our case study indicates

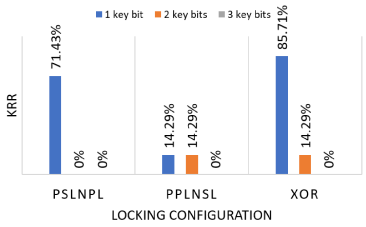


Figure 10: Impact of the number of key bits per cone on KRR.

that increasing the key space will improve the resilience against the CPA attack. Our quantitative assessment motivates us to develop a mitigation method to enlarge the key space and the logic cone size interested in the CPA attack.

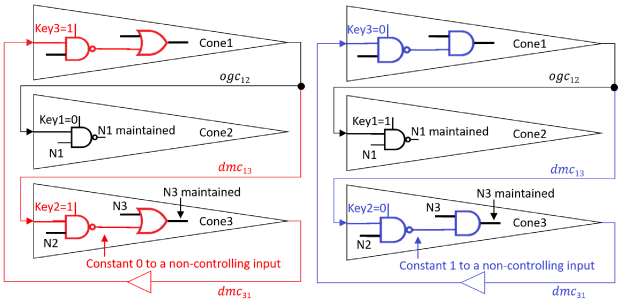
5 PROPOSED LOGIC-CONE CONJUNCTION (LCC) METHOD AGAINST CPA ATTACKS

The CPA attack in Section 4 follows the divide-and-conquer strategy to break the locked circuit cone by cone. To thwart the cone-based brute force attack, the work [4] uses MUX-based key gates to connect logic cones so that the key space and the cone size for one single cone is expanded. Inspired by that work, we propose a logic-cone conjunction (LCC) method to mitigate the CPA attack on transistor-level logic locking circuits. The MUX-based attack mitigation method uses multiplexers to create a small overlap area between two logic cones. However, there will always be some keys out of the overlap area. Those keys will not help in expanding the key space unless the two nets connected by the multiplexers are both primary outputs. In contrast, our LCC method embeds one entire logic cone into another one such that the key space can be enlarged significantly. Typically, increasing the number of key bits is a common practice to raise the difficulty of CPA attacks. The proposed LCC method does not induce additional key bit insertion; instead, our method makes full use of the existing locking keys in a locked circuit to maximize the key space of each logic cone. The key space means the number of all possible distinct key combinations.

The proposed LCC inserts a key-controlled dummy connection dmc_{ij} between two independent logic cones C_i and C_j to extend the size of each logic cone. To maximize the key space after our logic-cone conjunction, the selected independent logic cones C_i and C_j should use the exclusive key vectors, $\overrightarrow{K_i}$, and $\overrightarrow{K_j}$, respectively. The LCC method will increase the key space for the logic cone C_i from 2^{K_i} to $2^{K_i+K_j}$. In the best case, $K_i + K_j$ will be equal to the number of all key bits inserted in the locked circuit.

Figures 11 (a) and (b) illustrate how the proposed LCC method is applied to the PSLNPL and PPLNSL transistor-level locking circuits. Cone1 and Cone2 are dependent due to the original connection ogc_{12}. In contrast, Cone1 and Cone3 are originally independent since there is no logic overlap between them.

For the configuration of PSLNPL shown in Fig. 11(a), designers can insert one key bit, Key1, to the NAND gate in Cone2. The correct key value for Key1 should be logic 0 since Cone1 and Cone2 are originally connected and Cone2 needs signal ogc_{12} to switch normally. Note that inserting Key1 is an important step in LCC. In Cone 3, a dummy NAND locked by the PSLNPL configuration with the key bit Key2 is added. This NAND gate is driven by the output



(a) (b)

Figure 11: LCC diagram for (a) PSLNPL and (b) PPLNSL configurations.

signal dmc_{13} from Cone1 and the net N2, which is any existing net in Cone3. The correct key value for Key2 should be logic 1, which forces the output of the NAND to be constant 0. As the constant 0 will be given to an input of an OR gate, the dummy connection dmc_{13} will not interrupt the original Cone3 operation. Because 0 is the non-controlling bit of an OR gate, its output will be determined by the original net N3. The output of the Cone3 dmc_{31} is brought back to Cone1 to form a cyclic structure between Cone1 and Cone3 using a similar dummy connection. The dummy conjunction between Cone1 and Cone3 will increase the key space for both cones. In this case, the key space for Cone1 (Cone3) is increased from 2^{K_1} (2^{K_3}) to $2^{K_1+K_3}$. Furthermore, no matter which cone is attacked, the cyclic logic structure makes the other cones also switch, thus inducing noise to the power traces collected for CPA attacks. The power noise blurs the correlation between the locking key and the power traces.

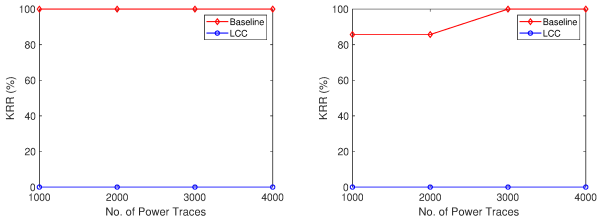
The LCC for PPLNSL configuration can be implemented in a similar way. As shown in Fig. 11(b), we replace the OR gate with an AND gate and the correct Key2 is logic 0. This is because applying a logic 0 to the NAND gate locked with PPLNSL will lead to a constant 1, which is the non-controlling bit for the AND gate. The logic gates used in Fig. 11 can be substituted with other gates as long as the normal operations of the revised cones can be maintained when the correct key is provided. To achieve the maximum key space, there could be more than two cones in the conjunction.

The proposed LCC significantly improves the CPA resilience of the transistor-level logic locking for two reasons. First, it significantly enlarges the key space for every single cone to mitigate the cone-based CPA attack. Second, as the LCC method forms the connected cones as a cyclic structure, no matter which cone in the structure is attacked, all other cones will switch. The increased switching activities lead to some power noise, which interferes with the power trace measurement for CPA attacks.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

We performed the experimental verification and evaluation for the proposed LCC through FPGA emulations. Both PSLNPL and PPLNSL configurations, with and without LCC, were applied to the ISCAS benchmark circuit c432 and implemented in a SAKURA-G FPGA board. The power traces were collected using ChipWhisperer



(a) (b)

Figure 12: KRR comparison for (a) PSLNPL and (b) PPLNSL configurations.

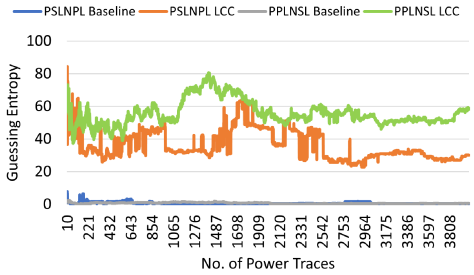


Figure 13: Guessing entropy comparison.

software. The CPA algorithm was realized in MATLAB and the Xilinx Vivado design suite. The hardware overhead of LCC was assessed in the Xilinx PlanAhead and XPower Analyzer software.

Seven key bits were inserted to c432 for both PSLNPL and PPLNSL configurations following the fault analysis-based logic locking (FLL) [5] to achieve the maximum output corruptibility. c432 has seven logic cones for the primary outputs N223, N329, N370, N421, N430, N431, and N432. N223 is also an input for the logic cone of N329. Furthermore, N329 is fed to the logic cone of N370. Finally, N370 drives the logic cones for N421, N430, N431, and N432. Based on the locked netlist, we found that connecting the output of N421 cone to N223 cone can help to maximize the key space and induce the largest amount of noise to the power traces. In the following experiments, we assume that there is an extra key beside the seven keys to lock the dummy connection logic between N421 and N223 cones and this extra key is known to the CPA attacker for a fair comparison with the baseline.

6.2 Improved Resilience against CPA Attack

We collected 4000 power traces for the assessment of KRR. As shown in Fig. 12, the CPA attack successfully retrieves all the 7 keys (100% KRR) of the baseline c432 for both locking configurations. In contrast, the KRR of the c432 protected with LCC decreases from 100% to 0% for both PSLNPL and PPLNSL locking configurations. We also zoom in the guessing entropy for the baseline and the LCC-protected c432. As shown in Fig. 13, the guessing entropy of the baseline is close to 0 while the proposed LCC improves the entropy to a much higher level. Both KRR and guessing entropy indicate that the LCC method successfully enhances the locking circuit's resilience against the CPA attack.

The improved attack resilience is originated from the cyclic structure generated by LCC. Because of the cyclic logic loop, the uninterested cones will have logic switching when the target cone is under

Table 1: Comparison of Cone Interference (CI).

cones	PSLNPL		PPLNSL	
	Baseline	LCC	Baseline	LCC
N432	81.44%	82.37%	81.86%	81.45%
N431	81.30%	81.75%	81.45%	82.11%
N430	82.13%	82.05%	82.19%	81.13%
N421	90.32%	90.02%	90.62%	90.59%
N370	83.42%	82.53%	82.81%	83.25%
N329	**27.96%**	**86.72%**	**26.84%**	**86.60%**
N223	**0%**	**95%**	**0%**	**94.87%**

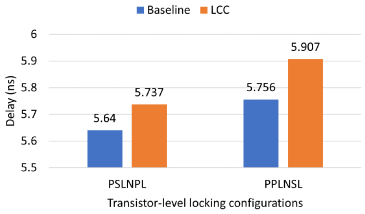


Figure 14: Delay overhead.

attack. Consequently, LCC yields some power noise and thus undermines the CPA attack. We use a metric *Cone Interference*(CI) expressed in Eq. (2) to evaluate the noise induced by LCC.

$$CI = \frac{Switching\ Events\ of\ Uninterested\ Cones}{Switching\ Events\ of\ Entire\ Circuit} \tag{2}$$

In which *Switching Events of Uninterested Cones* is the total number of bit flips on the primary outputs of the logic cones that the attacker is not interested in. *Switching Events of Entire Circuit* is the total number of bit flips on all the primary outputs of the circuit. Based on the results shown in Table 1, LCC significantly improves the cone interference in the attacks to N223 and N329 cones. This is because LCC forces all 7 cones of c432 to switch no matter which one is under attack. Because cones N223 and N329 are the smallest cones that are included in any other cone of c432, interfering with the power traces of these two cones is extremely important to securing the entire circuit. The cone interference for the LCC protected c432 and the baseline are comparably high after N329. Since all the remaining cones are driven by all the primary inputs of c432, all 7 cones of c432 will switch when any of the logic cones N370, N421, N430, N431, and N432 is under attack.

6.3 Overhead on Delay and Power

As the proposed LCC makes full use of the existing locking keys to expand the key space without inducing new key insertions, the hardware cost for our method is minor. The critical-path delay was measured via the Xilinx PlanAhead software. As shown in Fig. 14, the proposed LCC only increases the delay by 1.72% and 2.62% for the PSLNPL and PPLNSL configurations, respectively.

The power overhead was measured via the Xilinx XPower Analyzer. Based on the results shown in Fig. 15, for the PSLNPL based logic locking, LCC consumes 1% and 1.54% more power when the correct keys and the wrong keys are applied, respectively. For the PPLNSL configuration, LCC leads to 1.34% and 1.88% more power consumption for the scenarios that the correct and wrong keys were applied, respectively.

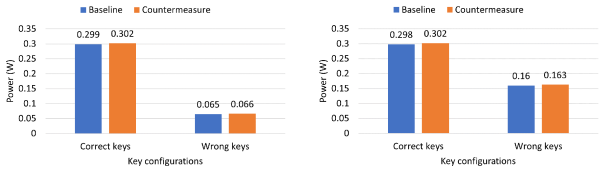


Figure 15: Power overhead for (a) PSLNPL and (b) PPLNSL configurations.

7 CONCLUSION

Logic locking has been broadly used to prevent ICs from design piracy attacks. Steady progress has been made on improving the resilience of logic locking against various functional-based attacks. However, limited works evaluate the strength of logic locking techniques on resisting power analysis attacks. This work proposes a CPA attack flow that is applicable to the transistor-level logic locking. Our analysis and experimental results indicate that the proposed CPA attack outperforms the DPA attack in transistor-level logic locking and achieves a 100% KRR in the locked c432 with 4000 power traces. Furthermore, we propose a logic-cone conjunction method to enlarge the key space. Our case study on c432 shows that our method can successfully reduce the KRR to zero with negligible overhead on delay and power.

ACKNOWLEDGMENTS

This work is partially supported by NSF award CNS-1652474 and NSF/SRC award CNS-1717130.

REFERENCES

[1] P. Chakraborty, J. Cruz, and S. Bhunia. 2018. SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation. In *Proc. AsianHOST'18*. 56–61.
[2] J. Dofe, Chen Yan, S. Kontak, E. Salman, and Q. Yu. 2016. Transistor-level camouflaged logic locking method for monolithic 3D IC security. In *Proc. AsianHOST'16*. 1–6.
[3] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Proc. CRYPTO'99*, Michael Wiener (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 388–397.
[4] Y. Lee and N. A. Touba. 2015. Improving logic obfuscation via logic cone analysis. In *Proc. LATS'15*. 1–6.
[5] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. 2015. Fault Analysis-Based Logic Encryption. *IEEE Trans Comput* 64, 2 (2015), 410–424.
[6] V. S. Rathor and G. K. Sharma. 2019. A Lightweight Robust Logic Locking Technique to Thwart Sensitization and Cone Based Attacks. *IEEE Trans. Emerg. Topics Comput.* (2019), 1–1.
[7] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *Proc. DATE'08*. 1069–1074.
[8] S. M. Satheesh and E. Salman. 2012. Power Distribution in TSV-Based 3-D Processor-Memory Stacks. *IEEE Trans. Emerg. Sel. Topics Circuits Syst* 2, 4 (Dec 2012), 692–703.
[9] A. Sengupta, B. Mazumdar, M. Yasin, and O. Sinanoglu. 2020. Logic Locking With Provable Security Against Power Analysis Attacks. *IEEE TCAD* 39, 4 (2020), 766–778.
[10] P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *Proc. HOST'15*. 137–143.
[11] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. 2016. On Improving the Security of Logic Locking. *IEEE TCAD* 35, 9 (2016), 1411–1424.
[12] D. Zhang, X. Wang, M. T. Rahman, and M. Tehranipoor. 2018. An On-Chip Dynamically Obfuscated Wrapper for Protecting Supply Chain Against IP and IC Piracies. *IEEE Trans Very Large Scale Integr VLSI Syst* 26, 11 (2018), 2456–2469.
[13] J. Zhou and X. Zhang. 2020. A New Logic-Locking Scheme Resilient to Gate Removal Attack. In *Proc. ISCAS'20*. 1–5.