# Rescuing Logic Encryption in Post-SAT Era by Locking & Obfuscation

Amin Rezaei, Yuanqi Shen, and Hai Zhou
Northwestern University, Evanston, IL, USA
me@aminrezaei.com, yuanqishen2020@u.northwestern.edu, haizhou@northwestern.edu

***Abstract*—The active participation of external entities in the manufacturing flow has produced numerous hardware security issues in which piracy and overproduction are likely to be the most ubiquitous and expensive ones. The main approach to prevent unauthorized products from functioning is logic encryption that inserts key-controlled gates to the original circuit in a way that the valid behavior of the circuit only happens when the correct key is applied. The challenge for the security designer is to ensure neither the correct key nor the original circuit can be revealed by different analyses of the encrypted circuit. However, in state-of-the-art logic encryption works, a lot of performance is sold to guarantee security against powerful logic and structural attacks. This contradicts the primary reason of logic encryption that is to protect a precious design from being pirated and overproduced. In this paper, we propose a bilateral logic encryption platform that maintains high degree of security with small circuit modification. The robustness against exact and approximate attacks is also demonstrated.**

***Keywords*—Logic Encryption; Logic Locking; Circuit Obfuscation; SAT-based Attack; Logic Complexity; Structural Complexity; Affectability Ratio; Corruptibility Ratio**

## I. Introduction

Increasing the design costs of Integrated Circuits (ICs) on the one hand, and growing the number of powerful Reverse Engineering (RE) tools [1] on the other hand, make chip protection one of the vital priorities for the semiconductor industry. The main approach to prevent unauthorized products from functioning is logic encryption in which key-controlled gates are inserted to the IC netlist in a way that the valid behavior of the circuit only happens when the correct key is applied. To encrypt a circuit with traditional XOR-based encryption [2], first a random combination of $n$ buffers (for key bit "0") and inverters (for key bit "1") are chosen, and then each selected buffer or inverter is replaced with a key bit controlled XOR gate. The correct $n$-bit key can be stored in a tamper-proof memory or embedded into the circuit using dummy-contact [3], stealthy dopant-level [4], or polymorphic logic solutions [5], [6], [7].

However, the SAT-based attack [8] can defeat almost all of the traditional logic encryption methods [2], [9], [10], [11], [12], [13], [14]. The attack uses two copies of the encrypted circuit with the same input, but different key values under a given constraint to check whether it is still possible to generate different outputs. Such input patterns are called Differentiating Input Patterns (DIPs). Each DIP is then used to query the activated IC as a black-box to get the correct output. Then, the DIP with the output is used to further constrain the keys under consideration. The power of the SAT-based attack lies on the fact that a single query can remove a large number of wrong keys.

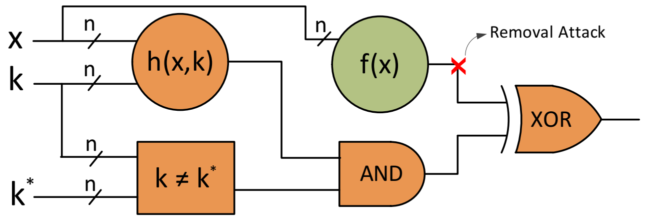
Traditional logic encryption schemes considered only key insertion without worrying about structural analysis, while the post-SAT era approaches have to explicitly protect the vulnerable structure of the SAT-proof component from the removal attack. Thus, in post-SAT era, logic encryption can be separated into two closely related goals: Logic locking and circuit obfuscation. We define logic locking as a logical request to make sure that the correct key cannot be easily figured out by studying the logic of the encrypted circuit. On the other hand, we specify circuit obfuscation as a structural request to make sure that the original circuit cannot be simply extracted by structural analyses of the encrypted circuit.

Based on the above definitions, the focus of the recent approaches has been on the locking goal with little attention to the obfuscation part. In addition, even the locking scheme may be vulnerable against improved versions of the SAT-based attack [15], [16], [17], [18], [19] that can return either an exact or approximately correct key [20]. The main contributions of the paper are three-fold:

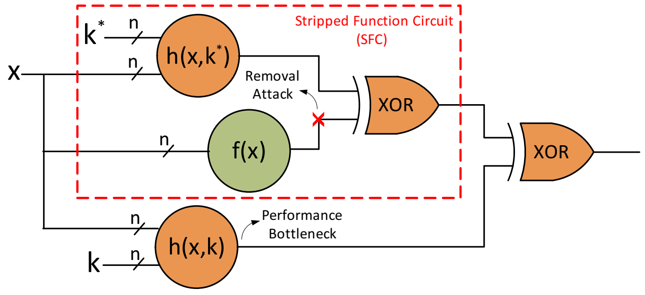
- Suggesting a secure locking scheme against the original and the improved versions of the SAT-based attack;
- Obfuscating only a small part of the original circuit to reduce the overhead and maintain the performance;
- Integrating locking & obfuscation in post-SAT era instead of postponing obfuscation to resynthesis.

### A. Definitions

Suppose a Boolean function $f : B^n \rightarrow B^q$ represented by a multi-level netlist of logic gates with a $q$-vector output and $f' : B^m \rightarrow B$ as a Boolean sub-function of $f$ (i.e., $m \leq n$) with a single output. In other words, $f'$ is a sub-circuit of $f$. Also, suppose a Boolean function $g : B^{n+l} \rightarrow B^q$ as a locked version of function $f$ in which there is a Boolean $l$-vector $k^*$ such that $g(x, k^*) \equiv f(x)$. Furthermore, suppose a class $H$ of obfuscated circuits including Boolean function $h : B^{n+l+o} \rightarrow B^q$ in which there is an $o$-vector $p^*$ such that $h(x, k, p^*) \equiv g(x, k)$ with the following properties: First, any two circuits in $H$ are structurally indistinguishable. Second, given any obfuscated circuit $h(x, k, p)$ in which $p \neq p^*$, structurally separating the original circuit $f(x)$ from the locked circuit $g(x, k)$ is exponentially hard with regard to the $p$ size.



(a)



(b)

Fig. 1: Logic locking (a) Standard [21] (b) Stripped functionality [22]

**Corruptibility Ratio (CR):** We define CR of sub-function $f'$ as the number of input patterns in $f$ in which flipping the output of $f'$ also changes the output vector of $f$ divided by the number of all input patterns in $f$.

**Affectability Ratio (AR):** We define AR of sub-function $f'$ as the number of all input patterns in $f'$ divided by the number of all input patterns in $f$. In other words, $AR = 2^{m-n}$.

**Error Number (EN):** We define error of a key $k$ as the number of input patterns in which there is at least one inconsistency between the output vector of $g$ under $k$ and the output vector of $f$. Accordingly, we define EN of the locked function $g$ as the average error among all the wrong keys.

**Logic Complexity (LC):** We define LC of a locked function $g$ as the average number of DIPs that is required to test under the SAT-based attack [8] in order to reveal $k^*$.

**Structural Complexity (SC):** We define SC of an obfuscated function $h$ as the size of its corresponding class $H$.

## *B. Motivation*

After proposing the SAT-based attack, different logic locking methods [23], [24], [25] have been introduced to increase the required number of DIPs exponentially with the key size. However, these incremental techniques have two main drawbacks. First, although they have high LC, they suffer from very low EN. Second, the lock component is nearly separated from the rest of the circuit. In other words, no SC parameter is defined for such methods.

Due to the first drawback, the above methods are vulnerable to approximate SAT-based attacks [15], [18], [17] that can return an almost correct key in which only a small number of input patterns produce wrong outputs. As an example, AppSAT [15] first uses the original SAT-based attack to prune some of the wrong keys with a certain number of DIPs. Then, the SAT solver is utilized to report a key satisfying all these DIPs. To estimate the error of the reported key, random testing is adopted. If the estimated error is below a specified threshold, the reported key is considered as an approximate key. Otherwise, the samples that resulted in disagreement will be added to the SAT formula as new constraints.

Because of the second drawback, the attacker can easily remove the lock by structural analysis of the encrypted circuit. Thus, a secure circuit obfuscation scheme is required that has been given over to resynthesis in state-of-the-art post-SAT era works. However, a large portion of the original circuit needs to be modified in order to guarantee security against the removal attack. But in most cases, the value of a design is just the structure of the efficient netlist, and resynthesizing the circuit for obfuscation will lose the design treasure. Recently it is shown that stripped functionality logic locking [22] that is viewed by the community as the most advanced and thus perhaps the most secure approach to logic encryption is still vulnerable to a series of structural analysis attacks [26] even in the case of resynthesis.

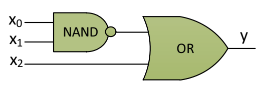
In this paper, for overcoming the first drawback, we suggest a secure logic locking scheme that defeats the original SAT-based attack and its approximate versions. In order to solve the second problem, we propose a practical circuit obfuscation scheme that protects the lock component with small circuit modification. Our proposed logic encryption platform that maintains high degree of security with low-overhead, consists of the following steps:

- First, a sensitive sub-circuit with high CR and high AR is extracted.
- Second, the extracted sub-circuit is locked using a standard or stripped functionality logic locking with high EN and high LC.
- Third, the locked sub-circuit is obfuscated adopting an efficient routing-based circuit obfuscation scheme with high SC.
- Fourth, the obfuscated sub-circuit is concatenated with the original one.

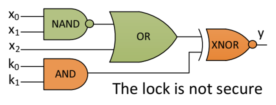
In state-of-the-art works of logic encryption, it is supposed that the attacker has access to the physical layout. Moreover, he can acquire a functioning circuit from the market as a black-box and get the correct outputs for given input vectors. Also, since almost all ICs are sequential circuits, it is assumed that scan chain is accessible to the attacker. In this paper, we also consider the above attacker model.

## *C. Warm-up Example*

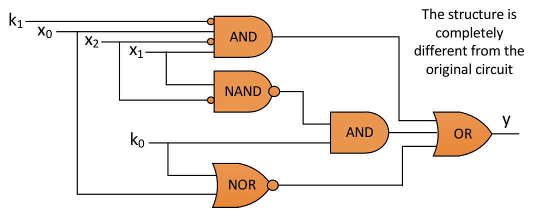
Standard and stripped functionality logic lockings [21], [22] can be viewed as Fig. 1a and Fig. 1b respectively. However, making the difference logic explicit introduces a structural vulnerability; the difference logic has to be XORed with the original circuit to form the encryption circuit. Therefore, without circuit obfuscation, the original circuit lays exposed for piracy. In order to prevent such removal attack on standard logic locking, the whole circuit needs to be obfuscated. Not



(a)



(b)


k1
x0
x2
x1
AND
NAND
AND
OR
y
k0
NOR
The structure is completely different from the original circuit

(c)

Fig. 2: Warm-up example (a) Original circuit (b) Locked circuit with $k^* = 11$ (c) Obfuscated circuit with resynthesis

only the Stripped Function Circuit (SFC) needs to be obfuscated in stripped functionality locking, but also the restore unit may dominate the system performance parameters.

Fig. 2a depicts a simple circuit to be encrypted as an example. First, the circuit is locked with an ad-hoc scheme shown in Fig. 2b. Then, the locked circuit is obfuscated with resynthesis shown in Fig. 2c to hide the vulnerable structure of the lock. This encryption has two main drawbacks. First, the secret key can be deciphered by a single SAT query. In other words, any chosen input pattern by the SAT-based attack can prune all the wrong keys. Second, the structure of the obfuscated circuit is completely different from the original one. In fact, when the resynthesis is considered for obfuscation, substantial modification of the original design is inevitable regardless of the locking approach.

## II. Bilateral Logic Encryption

In this section, we propose bilateral logic encryption that consists of four main stages namely extraction, locking, obfuscation, and concatenation. We use c17 circuit (i.e., Fig. 3) from ISCAS'85 benchmarks [27] as an example.

### A. Extraction

If only the sensitive component of the original circuit is extracted as the sub-circuit, it may have small CR. Thus, the effect of its encryption may not be fully transmissible to the original circuit. On the other hand, if one of the primary outputs of the original circuit is included in the chosen sub-circuit, CR will be equal to 1. Thus, the straight forward way is to specify the sensitive component of the circuit and then greedily add additional gates to its fan-out to reach the desired CR. Because the selection criteria of the sub-circuit is based on input/output sensitivity, the greedy algorithm can minimize the area overhead of the encrypted circuit. The case for making AR equal to 1 is easier since we can utilize the additional inputs of the original circuit as fan-in signals to encrypt the sub-circuit in a way that when the correct key is inserted, they act like don't care inputs but when a wrong key is inserted, they participate in forming the output of encrypted circuit. Fig. 4 shows a sub-circuit extraction example with CR=AR=1.

### B. Locking

Either the standard or the stripped functionality locking can be applied to the extracted sub-circuit as long as the adopted scheme has high EN and high LC. Fig. 5a shows a sub-circuit locking example based on the standard scheme of Fig. 1a with a 4-bit locking size. If an even key size $l = n$ is considered, in general we have:

$$\begin{aligned} h(x,k) &= \bigwedge_{i\in 0,\dots,\frac{n}{2}-1} (x_{2i} \oplus k_{2i}) \oplus (x_{2i+1} \oplus k_{2i+1}) \\ k \neq k^* &= \overline{\bigwedge_{i\in 0,\dots,n-1}} \quad k_i \overline{\oplus} k_i^* \end{aligned} \tag{1}$$

The above locking scheme has both EN and LC of $2^{\frac{n}{2}}$. Thus, exact (or approximate) SAT-based attack cannot reveal any correct (or approximate) key in linear time. Fig. 5b depicts another sub-circuit locking example this time using the stripped functionality scheme of Fig. 1b. The general form is:

$$h(x,k) = \bigvee_{i\in 0,\dots,n-1} x_i \oplus k_i \tag{2}$$

The above scheme has EN of 2 and LC of $2^{n-1}$. For the above scheme, although an exact attack cannot report any correct key in linear time, an approximate attack can easily report an approximate key with small error. To have both high EN and high LC, it is possible to adopt the following general scheme for stripped functionality locking with a squared number key size $l = n$:

$$h(x,k) = \bigwedge_{i\in 0,\dots,\sqrt{n}-1} \left[ \bigvee_{j\in i.\sqrt{n},\dots,(i+1).\sqrt{n}-1} x_i \oplus k_i \right] \tag{3}$$

### C. Obfuscation

The critical difference between the circuit obfuscation and program obfuscation is that the former one has key inputs while the latter one does not. In fact, the request to have key inputs is an intrinsic feature of logic encryption. When we allow key inputs and have protection mechanism for them, the

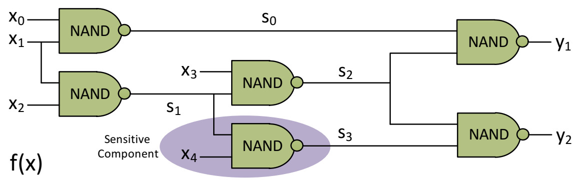


Fig. 3: Original c17 benchmark [27]

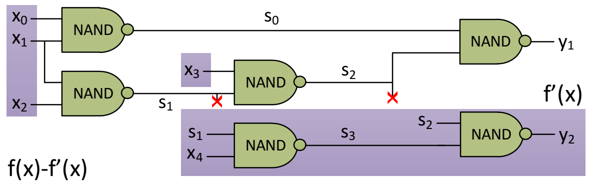


Fig. 4: Sub-circuit extraction example

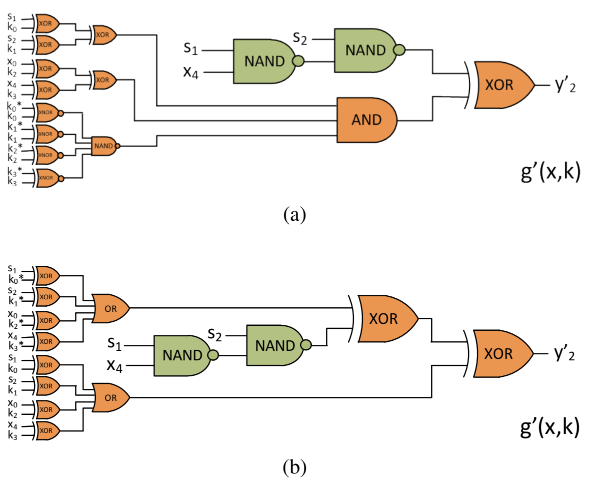


Fig. 5: Sub-circuit locking example (a) Standard (b) Stripped functionality

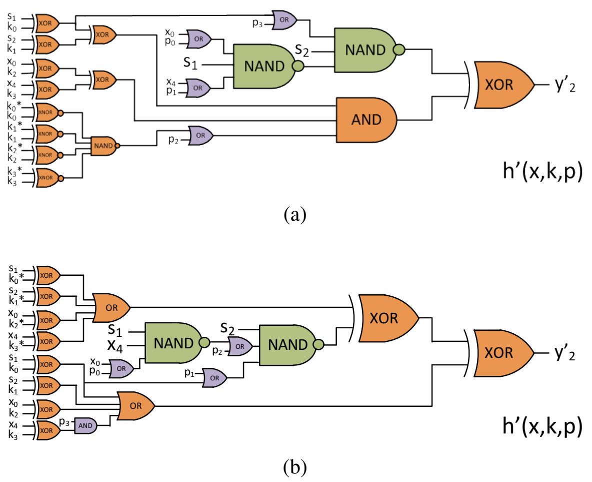


Fig. 7: Sub-circuit obfuscation example (a) Fig. 5a (b) Fig. 5b

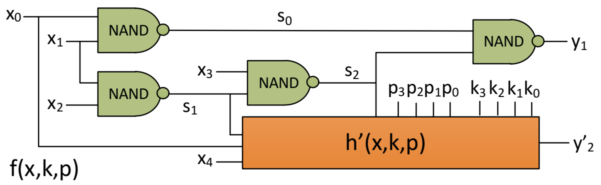


Fig. 8: Sub-circuit concatenation example

circuit obfuscation problem becomes simpler than the program obfuscation.

Signal routing can be utilized in order to obfuscate the locked circuit. First, we use breadth-first traversal of the locked circuit to assign a level to each gate based on its critical path from the primary inputs. In other words, all the gates with more than one input (i.e., inverters are not considered) in which they have the longest path $t$ from the primary inputs will be assigned into group $t$. Then, we obfuscate all the possible connections between adjacent levels using the obfuscation key. If the gate in level $t$ is AND gate, each signal in level $t-1$ will be connected as fan-in signal of the AND gate with an OR gate and a obfuscation key bit. If the connection really exists in the locked circuit, the correct value of the key bit is “0”; otherwise in order to neutralize the dummy signal, the correct value should be “1”. On the other hand, if the gate in level $t$ is OR gate, the same scenario will take place with the help of a key bit controlled AND gate. In this case, the correct value of the key bit for real signals is “1” and for dummy ones is “0”. Fig. 6 shows a signal routing obfuscation example.

In order to reduce the key size while still cope with the SC requirements, instead of obfuscating all the possible connection, the priority will be given to the signals that connect a gate in the inserted lock to the extracted sub-circuit and vice versa. In this case, the sub-circuit and the lock component will become structurally mixed and thus fulfilled the requirements of the class $H$ of obfuscated circuits. Fig. 7a and Fig. 7b show the obfuscation procedure for the locked circuits of Fig. 5a and Fig. 5b respectively with a 4-bit obfuscation key.

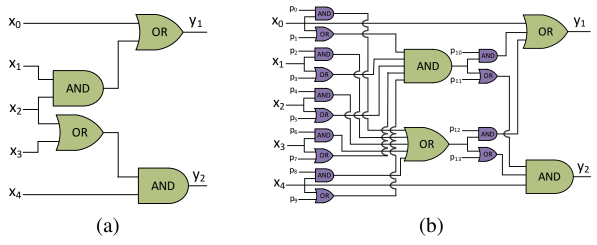


Fig. 6: Signal routing obfuscation example (a) Original circuit (b) Obfuscated circuit

## D. Concatenation

Finally, the encrypted sub-circuit can be concatenated with the original circuit. Fig. 8 depicts the concatenation stage example. Comparing Fig. 3 and Fig. 8, it is clear than input $x_0$ should not have any effect on output $y_2'$ when the correct key is inserted. However, it may have influence on the output when a wrong key is inserted.

## E. Discussion

For the circuit encrypted with the propose bilateral encryption approach, it may be possible for the attacker to identify the boundary of $h'(x, k, p)$. Then, he can remove the encrypted sub-circuit and build a partial circuit. Now, he has access to not only the activated IC but also the partially running circuit. However, when $AR = 1$ for the encrypted sub-circuit, running the SAT-based attack (either exact or approximate) on $h'(x, k, p)$ has the same attack complexity as running the attack on $h(x, k, p)$.

Another way to determine the unknown sub-circuit function is to run a brute-force attack on $h'(x, k, p)$ using the partial circuit and the activated IC. Again, if $AR = 1$, $2^{n+l+o}$ cases need to be checked which is equivalent to run a brute-force attack on $h'(x, k, p)$. Please note that the differentiation

TABLE I: Decryption results on the encrypted benchmarks with bilateral logic encryption approach

| Benchmark | #Inputs | #Keys | #Outputs | #Gates | Original SAT-based Attack [8] | | | AppSAT Attack [15] | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPU Time | #Iterations | Note | CPU Time | #Iterations | Note |
| apex2 | 39 | $40_{\times 2}$ | 3 | 610 | - | - | No Result | 17.548s | 262 | Wrong Key |
| apex4 | 10 | $10_{\times 2}$ | 19 | 5360 | 1.276s | 32 | Correct Key | - | - | No Attack |
| c17 | 5 | $6_{\times 2}$ | 2 | 6 | 0.02s | 7 | Correct Key | - | - | No Attack |
| c432 | 36 | $36_{\times 2}$ | 7 | 160 | - | - | No Result | 10.376s | 262 | Wrong Key |
| c499 | 41 | $42_{\times 2}$ | 32 | 202 | - | - | No Result | 15.724s | 262 | Wrong Key |
| c880 | 60 | $60_{\times 2}$ | 26 | 383 | - | - | No Result | 21.54s | 262 | Wrong Key |
| c1355 | 41 | $42_{\times 2}$ | 32 | 546 | - | - | No Result | 22.04s | 262 | Wrong Key |
| c1908 | 33 | $34_{\times 2}$ | 25 | 880 | - | - | No Result | 12.92s | 262 | Wrong Key |
| c2670 | 233 | $234_{\times 2}$ | 140 | 1193 | - | - | No Result | 106.82s | 262 | Wrong Key |
| c3540 | 50 | $52_{\times 2}$ | 22 | 1669 | - | - | No Result | 27.956s | 262 | Wrong Key |
| c5315 | 178 | $178_{\times 2}$ | 123 | 2307 | - | - | No Result | 96.136s | 262 | Wrong Key |
| c6288 | 32 | $34_{\times 2}$ | 32 | 2406 | - | - | No Result | 47.632s | 262 | Wrong Key |
| c7552 | 207 | $208_{\times 2}$ | 108 | 3512 | - | - | No Result | 106.212s | 262 | Wrong Key |
| dalu | 75 | $76_{\times 2}$ | 16 | 2298 | - | - | No Result | 39.732s | 262 | Wrong Key |
| des | 256 | $256_{\times 2}$ | 245 | 6473 | - | - | No Result | 156.664s | 262 | Wrong Key |
| ex5 | 8 | $8_{\times 2}$ | 63 | 1055 | 0.18s | 16 | Correct Key | - | - | No Attack |
| ex1010 | 10 | $10_{\times 2}$ | 10 | 5066 | 1.128s | 32 | Correct Key | - | - | No Attack |
| i4 | 192 | $192_{\times 2}$ | 6 | 338 | - | - | No Result | 87.628s | 262 | Wrong Key |
| i7 | 199 | $200_{\times 2}$ | 67 | 1315 | - | - | No Result | 92.992s | 262 | Wrong Key |
| i8 | 133 | $134_{\times 2}$ | 81 | 2464 | - | - | No Result | 66.5s | 262 | Wrong Key |
| i9 | 88 | $88_{\times 2}$ | 63 | 1035 | - | - | No Result | 37.16s | 262 | Wrong Key |
| k2 | 46 | $46_{\times 2}$ | 45 | 1815 | - | - | No Result | 26.336s | 262 | Wrong Key |
| seq | 41 | $42_{\times 2}$ | 35 | 3519 | - | - | No Result | 33.48s | 262 | Wrong Key |

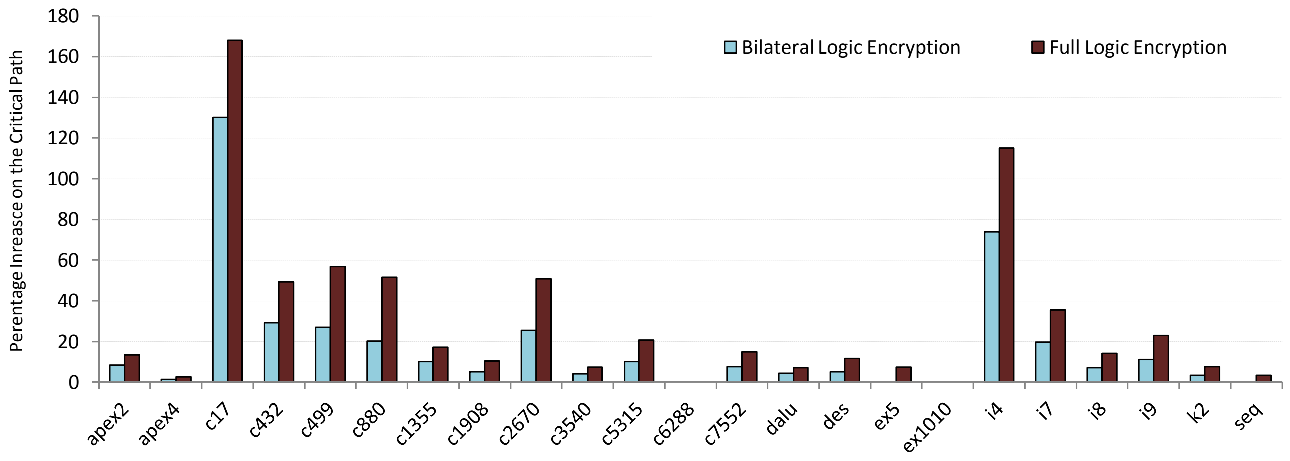


Fig. 9: Percentage increase on the critical path in full and bilateral logic encryptions

between the don't care primary inputs and the ones that actually matter to form $f'(x)$ is not possible by structural analysis of the encrypted circuit when SC is high and the correct key is unknown.

## III. Experimental Results

For experiments, we use combinational circuits of ISCAS'85 [27] and MCNC'91 [28]. For each benchmark a random sensitive component is chosen. Then additional gates are greedily added to its fan-out to reach a primary output from the sensitive component. Next, the chosen sub-circuit is locked with the proposed standard scheme of Equation 1. Please note that the locking key size of each benchmark is selected to be almost equal to the number of its primary inputs ($l \approx n$). Afterward, the locked circuit is obfuscated with the proposed routing-based obfuscation method with the obfuscation key size equal to the locking key size ($p = l$). Finally the encrypted sub-circuit is concatenated with the original circuit. The decryption results under the original SAT-based [8] and AppSAT [15] attacks are shown in Table I.

First, we ran the SAT-based attack on each benchmark for one day long. As can be seen, it can only decrypt the small size circuits (i.e., apex4, c17, ex5, and ex1010). Even for these circuits, the required number of iterations is in the order of $2^{\frac{n}{2}}$ since the locking scheme has high LC. Then, we ran the AppSAT attack on the benchmarks with no reported result under the SAT-based attack. The threshold of the AppSAT attack is considered to be five. This is the same threshold that is used in the AppSAT paper [15]. After every 12 iterations of the SAT-based attack, 50 iterations are done for random sampling. Thus, it takes 262 iterations for each benchmark. However, still an exponentially large number of input patterns produce wrong outputs under the reported keys. This happens because the EN of the locking scheme is $2^{\frac{n}{2}}$.

As another experiment, each benchmark is locked with the same scheme and locking key size. Then, the whole circuit is resynthesized using ABC synthesis tool [29]. Since the same locking scheme is adopted, the full logic encryption is still secure against exact and approximate SAT-based attacks. However, the structure of the original circuit is completely changed due to resynthesis. Fig. 9 depicts the percentage increase on the critical path of the original unencrypted circuits after both bilateral and full logic encryptions. The critical path increase in the full logic encryption is on average 1.7x more than the bilateral one.

## IV. Conclusion

In this paper, we proposed a new perspective on logic encryption using integrated locking and obfuscation on a sensitive component of a circuit. As long as both CR and AR of the sensitive sub-circuit are high, the security impact of its encryption (i.e., both LC and SC) is transmissible to the whole circuit. The experiments confirmed that we can securely protect a precious design with small circuit modification if the bilateral logic encryption approach is adopted. In addition, the bilateral logic encryption imposes much less performance overhead on the circuit that the full logic encryption.

## Acknowledgment

This work is partially supported by NSF under CNS-1441695, CNS-1651695, and CCF-1533656.

## References

[1] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011, pp. 333–338.

[2] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Design, Automation and Test in Europe (DATE)*, 2008, pp. 1069–1074.

[3] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware ip protection," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, article 153.

[4] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2013, pp. 197–214.

[5] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2018, pp. 85–90.

[6] A. Rezaei, J. Gu, and H. Zhou, "Hybrid memristor-cmos obfuscation against untrusted foundries," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 535–540.

[7] Q. Alasad and J. Yuan, "Logic obfuscation against ic reverse engineering attacks using plgs," in *IEEE International Conference on Computer Design (ICCD)*, 2017, pp. 341–344.

[8] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[9] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," in *Computer*, vol. 43, Issue 10, 2010, pp. 30–38.

[10] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Design, Automation and Test in Europe (DATE)*, 2012, pp. 953–958.

[11] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," in *IEEE Transactions on Computers*, vol. 64, issue 2, 2015, pp. 410–424.

[12] S. Dupuis, P. S. Ba, G. D. Natale, M. L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *IEEE International On-Line Testing Symposium (IOLTS)*, 2014, pp. 49–54.

[13] K. Juretus and I. Savidis, "Reduced overhead gate level logic encryption," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2016, pp. 15–20.

[14] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," in *IEEE Design Test of Computers*, vol. 27, Issue 1, 2010, pp. 66–75.

[15] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately deobfuscating integrated circuits," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[16] Y. Shen, A. Rezaei, and H. Zhou, "Sat-based bit-flipping attack on logic encryptions," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2018, pp. 629–632.

[17] X. Xu, B. Shakya, M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2017, pp. 189–210.

[18] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 179–184.

[19] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "Sigattack: New high-level sat-based attack on logic encryptions," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 940–943.

[20] Y. Shen, A. Rezaei, and H. Zhou, "A comparative investigation of approximate attacks on logic encryptions," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 271–276.

[21] H. Zhou, "A humble theory and application for logic encryption," in *Cryptology ePrint Archive, Report 2017/696*, 2017.

[22] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1601–1618.

[23] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[24] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016, pp. 127–146.

[25] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for ic protection," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, article 28.

[26] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2019, pp. 936–939.

[27] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1985, pp. 677–692.

[28] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," in *Microelectronics Center of North Carolina (MCNC) International Workshop on Logic Synthesis*, 1991.

[29] B. L. Synthesis and V. Group, "Abc: A system for sequential synthesis and verification," in *http://www.eecs.berkeley.edu/ alanmi/abc/.*