Distributed Logic Encryption: Essential Security Requirements and Low-Overhead Implementation

Raheel Afsharmazayejani
Independent Researcher, CARS-Lab
CSULB, CA, USA

Hossein Sayadi
California State University Long Beach, CA, USA

Amin Rezaei
California State University Long Beach, CA, USA

ABSTRACT

Due to outsource manufacturing, the semiconductor industry must deal with various hardware threats such as piracy and overproduction. To prevent illegal electronic products from functioning, the circuit can be encrypted using a protected key only known to the designer. However, an attacker can still decipher the secret key utilizing a functioning circuit bought from the market, and the encrypted layout leaked from an untrusted foundry. In this paper, after introducing essential conformity and mutuality features for secure logic encryption, we propose *DLE*, a novel Distributed Logic Encryption design that resists against all known oracle guided and structural attacks including the newly proposed fault-aided SAT-based attack that iteratively injects a single stuck-at fault to thwart the locking effect. *DLE* forces the attacker to insert multiple stuck-at faults simultaneously in critical points to achieve a smaller but meaningful encrypted circuit; thus, exponentially reducing the chance to hit all the critical points with properly located stuck-at fault injections. Our experiments confirm that *DLE* maintains an exponentially high degree of security under diverse attacks with the polynomial area and linear performance overheads.

CCS CONCEPTS

• **Security and privacy → Security in hardware**.

KEYWORDS

Logic Encryption; Logic Lockig; SAT Attack; Fault Injection

ACM Reference Format:
Raheel Afsharmazayejani, Hossein Sayadi, and Amin Rezaei. 2022. Distributed Logic Encryption: Essential Security Requirements and Low-Overhead Implementation. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3526241.3530372

1 INTRODUCTION & BACKGROUND

Since an Integrated Circuit (IC) usually consists of several parts from different foundries, each with its possible hardware threats, the semiconductor industry must deal with different security challenges such as piracy and overproduction. A well-studied but still vulnerable approach to prevent unauthorized ICs from working is logic encryption (a.k.a., logic locking or hardware obfuscation) [2, 3, 6, 7, 10, 13, 14] in which it modifies a given netlist with extra key inputs only known to the designer. The designer then inserts (or embeds) the correct key in a tamper-proof memory (or camouflaged gates) before releasing the IC fabrics to the market. However, the SAT attack [1] has been able to successfully decrypt the Random Logic Encryption (RLE) methods [2, 3]. From then on, there have been cat-and-mouse attacks such as App-SAT [5] and defenses like Anti-SAT [7] and Stripped Functionality Logic Locking (SFLL) [13] in the logic encryption research field mostly due to the lack of well-defined encryption design requirements. While RLE schemes considered only key insertion without worrying about the structural attacks such as Functional Analysts Attack (FAA) [9], the post-SAT era approaches have to explicitly protect the vulnerable structure of the SAT-hard lock.

Recently, Bilateral Logic Encryption (BLE) [10] has been proposed, where a compound SAT-hard and approximate SAT-hard locking scheme was combined with a routing-based obfuscation method to thwart variants of exact and approximate logic and structural attacks [1, 4, 5, 8, 9]. The BLE technique was unbroken until the proposal of Fa-SAT [11], an efficient and immensely powerful fault-injection-based method, that assists the original SAT attack to decrypt BLE in sub-exponential time. Fa-SAT inserts a single stuck-at fault at each signal of the encrypted circuit iteratively; then, it feeds the faulty encrypted circuit to the SAT attack [1] framework with a given timeout. In Fa-SAT, inserting the fault at an incorrect point can result in reporting a wrong key. Therefore, additional functional verification (implemented by random sampling) is required to check the correctness of the reported key. Since the BLE scheme has a single CP as depicted in Figure 1a, if a stuck-at-1 fault will be inserted in the output of the h-function, Fa-SAT will be able to nullify the security effect of this function and report the correct key. Table 1 summarizes the state-of-the-art attacks on various logic encryption techniques.

Table 1: Attacks and defenses in logic encryption

Attacks	Defenses				
	RLE [2, 3]	Anti-SAT [7]	SFLL [13]	BLE [10]	DLE (this paper)
SAT [1]	✓	✗	✗	✗	✗
AppSAT [5]	✓	✓	✗	✗	✗
FAA [9]	✗	✗	✓	✗	✗
Fa-SAT [11]	✓	✓	✓	✓	✗

Note: AppSAT can report an approximate key of Anti-SAT.

In this paper, we propose *DLE*, a Distributed Logic Encryption design against existing oracle-guided & structural attacks including Fa-SAT. *DLE* forces the attacker to insert multiple stuck-at

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA.

ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530372

faults simultaneously in Critical Points (CPs) to achieve a smaller but meaningful encrypted circuit; thus exponentially reducing the chance to hit all the CPs with the properly located stuck-at fault injections at the same time. The primary contributions of this paper are three-fold:

- Introducing essential conformity and mutuality features for secure logic encryption.
- Proposing a novel and low-overhead distributed logic encryption design that is simultaneously secure against SAT, approximate SAT, fault-aided SAT, and structural attacks.
- Presenting the exponential security gain of our encryption method on diverse attacks.

In state-of-the-art post-SAT era works [6, 7, 10, 13], it is assumed that the attacker has full access to the physical layout. Moreover, he/she 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 the scan chain is accessible to the attacker. In this paper, we also consider the above well-accepted attacker model in the community.

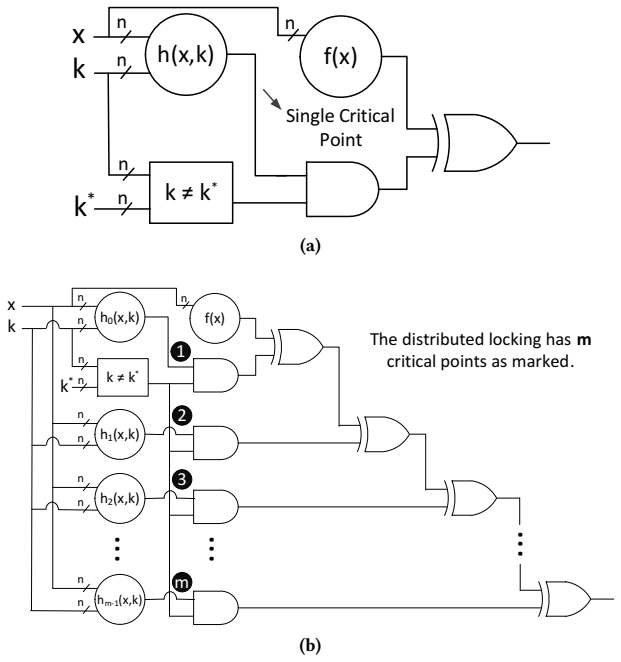

Figure 1: Logic locking (a) Standard [10, 14] (b) Distributed (this paper)

2 DISTRIBUTED LOGIC ENCRYPTION

In order to defeat all the existing attacks on logic encryption [1, 4, 5, 8, 9, 11, 12], we propose *DLE* consisting of joint low-overhead locking and obfuscation as follows.

2.1 Definitions and Problem Statement

Suppose a Boolean function $f : B^n \rightarrow B$ represented by a multi-level netlist of logic gates with a single output. Please note that a single output function is considered for simplicity. Also, suppose a Boolean function $g : B^{2n} \rightarrow B$ as a locked version of function f in which there is a Boolean n-vector k^* such that $g(x, k^*) \equiv f(x)$. Furthermore, suppose a class S of obfuscated circuits including Boolean function $s : B^{3n} \rightarrow B$ in which there is a Boolean n-vector p^* such that $s(x, k, p^*) \equiv g(x, k)$ with the following properties: First, any two circuits in S are structurally indistinguishable. Second, given any obfuscated circuit $s(x, k, p)$ in which $p \neq p^*$, structurally separating the original circuit f from the locked circuit g is exponentially hard with regard to the p size.

Logic Complexity (LC): We define LC of a locked function g as the minimum number of test cases (i.e., iterations) that is required under the SAT attack [1] to reveal k^*. Obviously, the higher the LC of a locked function, the more secure against exact SAT attack [1].

Error Number (EN): We define error of a key $\hat{k}$ as the number of input patterns in which $g(x, \hat{k}) \neq f(x)$. Accordingly, we define EN of the locked function g as the minimum error among all the wrong keys. The higher the EN of a locked function, the more secure against approximate SAT attacks [4, 5] that can return an almost correct key. The reason is that when EN is high, lots of input patterns will produce wrong output under "any" wrong key.

Error Matrix (EM): Generally, we can build the following EM for any locked circuit $g(x, k)$.

$$EM = \begin{array}{c} \\ \hat{x}_0 \\ \hat{x}_1 \\ \\ \hat{x}_i \\ \\ \hat{x}_{2^n-1} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \hat{k}_0 & \hat{k}_1 & & \hat{k}_j & & k^* & & \hat{k}_{2^n-1} \end{array} \\ \begin{pmatrix} b & b & \dots & b & \dots & 0 & \dots & b \\ b & b & \dots & b & \dots & 0 & \dots & b \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ b & b & \ddots & b & \ddots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ b & b & \dots & \dots & \dots & 0 & \dots & b \end{pmatrix} \end{array}$$

Where $b = 0$ when $g(x, k) = f(x)$ under each key and input pair and $b = 1$ otherwise. Since k^* is the correct key, all the associated values for that column should be 0. On this EM, a logic encryption attack formulates as a covering problem: A subset of rows $\hat{x}_i s$ are sufficient if and only if they cover all the columns with ones [14]. We want a lock design such that the number of ones in each column other than the column corresponding to the correct key is large (i.e., EN is high,) while the number of rows needed to cover them is also large (i.e., LC is also high.) However, there is a clear contention between high EN and high LC. So, we need to think about how to achieve the best of both parameters.

Structural Complexity (SC): We define SC of an obfuscated function s as the size of its corresponding obfuscation class S. The higher the SC of an obfuscated function, the more secure against structural attacks [8, 9]. When SC is high, many functions can be implemented with the same structure as the obfuscated circuit, and in all of them (except the one with $p = p^*$), the original function f is structurally mixed with the locked function g.

Immediately after proposing the SAT attack, different defensive mechanisms [6, 7] were proposed using point functions to increase the required number of iternations exponentially with the key size. However, these techniques have two main drawbacks. First, although they have high LC i.e., they are SAT-hard, they suffer from low EN i.e., they are not approximate SAT-hard; thus, they

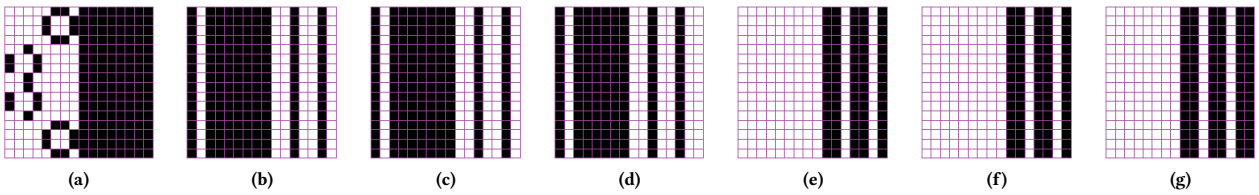

Figure 2: EM of distributed locking in Figure 1b based on Equation 1 with $n = 4$, $m = 3$, and correct key $k^* = \hat{k}_1 = 0001$. Black cells depict $b = 1$ and white cells depict $b = 0$. (a) No stuck-at fault (b) Stuck-at-1 fault in CP1 (c) Stuck-at-1 fault in CP2 (d) Stuck-at-1 fault in CP3 (e) Stuck-at-0 fault in CP1 (f) Stuck-at-0 fault in CP2 (g) Stuck-at-0 fault in CP3

are vulnerable to approximate SAT-based attacks [4, 5] that can return an almost correct key in which only a small number of input combinations produce wrong outputs. Second, no SC is defined for these approaches; thus, even their strongest version [13] is still vulnerable to structural attacks [8, 9] that can extract the original circuit from the encrypted one using structural analysis. On the other hand, BLE [10] suggested a locking method to choose the middle ground between high LC and high EN, as well as an obfuscation method to consider high SC. However, Fa-SAT [11] was successful in decrypting BLE because it combines structural and logic analyses, and thus is stronger than a standalone SAT-guided attack or an independent structural attack.

2.2 Distributed Logic Locking

To thwart Fa-SAT, one naive way is to lock the circuit multiple times each time with a new key. However, in this case, the key size will become very large. Thus, we suggest locking the circuit m times using *the same key* and a single key comparison component as depicted in Figure 1b.

Another naive way to adopt the proposed locking scheme is to repeat a single h-function multiple times i.e., $h_i(x,k) = h_0(x,k)$. If so, each pair of the h-functions neutralize each other; thus, if Fa-SAT inserts a stuck-at-1 fault on one of the CPs depicted in Figure 1b, it still will be able to break the scheme.

Based on our detailed analysis, in order to propose a secure logic locking scheme based on Figure 1b., we need to choose m distinct h-functions with the following essential security requirements:

- **Conformity feature:** If we use all the m h-functions, the distributed locking must have exponentially high LC and exponentially high EN with respect to the input size n.
- **Mutuality feature:** If we use any group of the $m-1$ h-functions, an exponential number of columns with respect to the input size n must have zero error in both the EM and the flipped EM of the distributed locking.

Lemma 1. *If the distributed locking in Figure 1b covers the conformity feature, it is secure against the SAT [1] and approximate SAT [4, 5] attacks.*

Proof. If LC is exponential, the SAT attack requires an exponential number of iterations to report the correct key, which reduces it to a brute-force attack. In addition, an exponential number of input patterns will produce wrong output under "any" wrong key if EN is exponential; thus, any reported key other than the correct key by the approximate SAT attacks performs no better than a random key. The chance that a randomly reported key hits the correct key is negligible. □

Neutralizing one of the h-functions by a powerful attack such as Fa-SAT might threaten the whole scheme. Thus, we need to define the h-functions in such a way that they maintain security even if there is a stuck-at fault in one of the CPs. Specifically, inserting a stuck-at fault at any of the CPs results in removing one of the h-functions (in the case of stuck-at-0) or flipping the EM of the scheme (in the case of stuck-at-1.)

Lemma 2. *If the distributed locking in Figure 1b covers the mutuality feature, it is secure against Fa-SAT [11] attack.*

Proof. If the scheme covers the mutuality feature, the keys with zero error in the EM and the flipped EM will behave mistakenly like a correct key when a stuck-at fault is inserted, and hence, the solver will be forced to report a random key among all the columns with zero error. Since the number of these columns is exponential, the chance of reporting the real correct key is negligible. □

To implement the proposed distributed locking scheme with conformity and mutuality features, we suggest the following formula to define h-functions with any $n = 2k$ and $3 \leq m = 2k+1 < n$.
$\forall i \in \{0, 1, ..., m-1\}$:

$$
\begin{aligned}
h_i(x,k) = \\
&\bigvee \bigwedge (x_{2j} \oplus k_{2j}) \oplus (x_{2j+1} \oplus k_{2j+1}), j \in 0, ..., \frac{n}{2} - 1 \\
&\bigwedge k \neq \hat{k}_w, \forall \hat{k}_w \in White(i) \\
&\bigvee k = \hat{k}_b, \forall \hat{k}_b \in Black(i)
\end{aligned}
\tag{1}
$$

$$
Black(i) = \{k = \hat{k}_{i+\frac{2^n}{2}+c\times m}\}, c \in 0, 1, ..., \frac{\frac{2^n}{2} - i - 1}{m}
$$

$$
White(i) = \{k = \hat{k}_d \notin Black(i)\}, d \in \frac{2^n}{2}, ..., 2^n - 1
$$

Where $Black(i)$ and $White(i)$ are the sets of keys with 2^n and zero errors, respectively.

Theorem 1. *If the distributed logic locking in Figure 1b adopts the h-function formula in Equation 1, it is secure against the SAT [1],approximate SAT [4, 5], and Fa-SAT [11] attacks.*

Proof. The proposed formula in Equation 1 covers the conformity feature since the entire distributed locking has LC of $2^{\frac{n}{2}}$ and EN of $2^{\frac{n}{2}}$. It also covers the mutuality feature since the EM of the scheme has at least $2^{\frac{n}{2}}$ columns with zero error upon inserting

stuck-at faults in each CP. Because the formula covers both conformity and mutuality features, based on Lemmas 1 and 2, it is secure against the SAT, approximate SAT, and Fa-SAT attacks. □

As an example, we consider $n = 4$ and $m = 3$ and define three distinct h-functions $h_0(x, k)$, $h_1(x, k)$, and $h_2(x, k)$ based on Equation 1. The EM of this distributed scheme with and without stuck-at faults is shown in Figure 2. In this figure, black cells depict $b = 1$ and white ones depict $b = 0$. As can be seen in Figure 2a all the h-functions together have $LC = 2^{\frac{4}{2}} = 4$ and $EN = 2^{\frac{4}{2}} = 4$. In addition, there exist at least $5 > 2^{\frac{4}{2}} = 4$ wrong keys with zero error upon insertion of a stuck-at-0 or a stuck-at-1 in each CP as shown in Figures 2b to 2g.

2.3 Distributed Logic Obfuscation

The locked circuit in Figure 1b can be divided into four main zones, including h-functions, CPs, original circuit, and final XOR connections. Most state-of-the-art works stop here and postpone the obfuscation to resynthesis. However, resynthesis as a means of obfuscation has two main drawbacks. First, it does not guarantee exponential SC and second, it may drastically change the structure of the circuit that we plan to protect, which is counter-intuitive. So, we introduce key controlled OR and AND gates to obfuscate some inter-zone and intra-zone connections with minimal modification to the circuit structure.

- **Real signal obfuscation:** Inside each zone at least one random signal is chosen; if the signal is the fan-in of an AND/NAND gate, a key bit controlled OR gate is added; if it is the fan-in of an OR/NOR gate, a key bit controlled AND gate is inserted.
- **Dummy signal obfuscation:** Inside each zone at least one random signal is chosen; then a random target gate outside the zone is selected; if the target gate is AND/NAND, a key bit controlled OR gate is added; if it is an OR/NOR gate, a key bit controlled AND gate is inserted.

We choose an obfuscation key size equal to the locking key size. If the target is an AND/NAND gate, to neutralize the dummy signal, the correct value of the corresponding obfuscation key bit should be "1" while for the real signal, the correct value should be "0". The chosen values are opposite if the target is an OR/NOR gate.

Lemma 3. *The proposed distributed signal obfuscation is secure against structural [8, 9] attacks.*

Proof. Applying the proposed distributed logic obfuscation with a key size n, the different zones of the locked circuit become structurally mixed, and thus, fulfill the requirements of the class S of obfuscated circuits with $SC = 2^n$. In other words, by structural analysis of the key bit connections, the zones of *DLE* are indistinguishable since the key bit controlled AND and OR gates have the same look for real inter-zone and dummy intra-zone signals. □

2.4 Security and Overhead Discussion

The original SAT attack [1] and its approximate versions [4, 5] cannot return the correct key of *DLE* in polynomial time because of the conformity feature that guaranties a joint SAT-hard and approximate SAT-hard locking. Standalone Fa-SAT [11] cannot return the correct key in polynomial time either because of the mutuality feature that forces the attack to return a random key. Thus, a modified version of Fa-SAT is required to insert m stuck-at faults simultaneously on the CPs marked by circles in Figure 1b.

Lemma 4. *A modified version of Fa-SAT [11] that inserts m stuck at faults on the CPs of the distributed locking in Figure 1b has an exponentially low chance of correctly identifying all the CPs with respect to m.*

Proof. Suppose the circuit has r signals, the chance to insert all the stuck-at faults correctly is as follows:

$$P = \frac{1}{\binom{r}{1}^m} = r^{-m}$$

□

Based on Lemmas 1 to 4, the following theorem can be resulted:

Theorem 2. *DLE is secure against all known logic and structural [1, 4, 5, 8, 9, 11] attacks.*

Considering the input size n and the number of h-functions m, it can be shown that the critical path increase of *DLE* is $O(n + m)$ and its area overhead is $O(nm)$. The proof is out of the scope of the paper due to its lengthy nature. However, we support our claim with experiments in Section 3. The linear performance overhead and polynomial area overhead are reasonable compared to the exponential gain on the degree of security. Our experiments in Section 3 show that even by choosing a small value for m (i.e., $m = 3$,) *DLE* is secure against existing logic and structural attacks.

3 EXPERIMENTAL RESULTS

For experiments, we adopted ISCAS'85 [15] and MCNC'91 [16] combinational benchmarks with different number of inputs and circuit sizes. The benchmarks were encrypted with *DLE* using three h-functions (i.e., $m = 3$) based on Equation 1.

We used 8GB of RAM and a 4-core processor at 2.20Ghz to run different attacks in Ubuntu 14.04. The decryption results are shown in Table 2. First, we ran the original SAT attack [1] on each benchmark for one day long. As can be seen, it can only decrypt the small size circuits (i.e., c17 and ex5.) Even for these circuits, the required number of iterations is in the order of $2^{\frac{n}{2}}$ since *DLE* has high LC. Please note that the whole function of the small benchmarks can be revealed using brute-force checking.

Then, as a representative of approximate attacks, we ran AppSAT [5]. The threshold of the AppSAT attack is considered to be five. This is the same threshold that is used in the AppSAT paper. After every 12 iterations of the SAT 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 *DLE* has high EN.

Finally, we ran Fa-SAT [11]. The same as AppSAT, 50 iterations of random sampling are utilized. Fa-SAT in most cases finishes with no reported key and returns a wrong key in one case. This is because by injecting stuck-at-1 or stuck-at-0 faults in *DLE*, the solver forces to report a wrong key, and most of the time the reported key will

Table 2: Decryption results on the encrypted benchmarks with DLE

Benchmark	#Inputs	#Keys	#Original gates	SAT attack [1]	AppSAT attack [5]	Fa-SAT attack [11]
apex2	39	$38_{\times 2}$	610	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (432.072 s)
c17	5	$4_{\times 2}$	6	Correct Key (4 it., 0.016 s)	No attack	No attack
c432	36	$36_{\times 2}$	160	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (65.552 s)
c499	41	$40_{\times 2}$	202	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (116.352 s)
c880	60	$60_{\times 2}$	383	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (172.35 s)
c1355	41	$40_{\times 2}$	546	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (280.098 s)
c1908	33	$32_{\times 2}$	880	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (212.72 s)
dalu	75	$74_{\times 2}$	2298	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (1089.067 s)
ex5	8	$8_{\times 2}$	1055	Correct key (16 it., 0.18 s)	No attack	No attack
i4	192	$192_{\times 2}$	338	No result (24 hours)	Wrong key (262 it.)	Wrong key (74.244 s)
i7	199	$198_{\times 2}$	1315	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (5406.73 s)
seq	41	$40_{\times 2}$	3519	No result (24 hours)	Wrong key (262 it.)	Finished w/ no result (345.668 s)

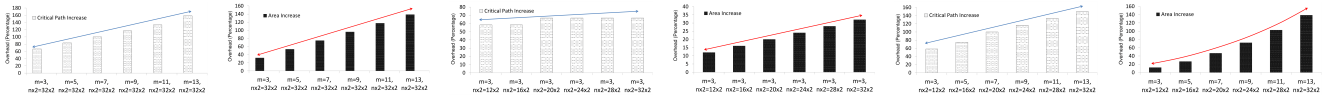

Figure 3: DLE average overhead in c1908 (a) Critical path increase, variable m (b) Area increase, variable m (c) Critical path increase, variable n (d) Area increase, variable n (e) Critical path increase, variables m & n (f) Area increase, variables m & n

not pass the random sampling step. However, the reported wrong key in *i4* benchmark passed the random sampling step by chance. It is worth mentioning that Fa-SAT can report the correct key of BLE [10] in a short time. Also, please note that FAA (FAA) [9] is primarily designed to attack SFLL [13] and is not *DLE* friendly. However, because a high SC parameter is foreseen in *DLE*, any removal attack has exponential time to differentiate the *DLE* zones with regard to the obfuscation key size.

As another experiment, without loss of generality, we chose *c*1908 benchmark and evaluated the percentage of the area and critical path increase in the encrypted benchmark with *DLE* compared to the original unencrypted benchmark. As can be seen in Figure 3, the average area and performance overhead linearly increase if we fix either the key size (i.e., $n_{\times 2}$) or the number of h-functions (i.e., m) and increase the other. If we concurrently increase both m and n, the critical path increase is still linear while the area increase is polynomial. Both the area and performance overheads are reasonable compared to the exponential security gain.

4 CONCLUSION

In this paper, we took a novel perspective on hardware intellectual property protection by proposing a low-overhead and highly secure distributed logic encryption. If both conformity and mutuality features are covered and the SC parameter is defined, none of the existing oracle guided and structural attacks [1, 4, 5, 8, 9, 11] are successful to reveal the correct key and/or the original circuit. The experiments confirmed that by adopting *DLE*, we can exponentially secure a digital design with an only linear performance overhead and polynomial area overhead.

5 ACKNOWLEDGEMENT

This work is partially supported by NSF under award No. 2131156.

REFERENCES

[1] P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 137-143.
[2] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. Epic: Ending piracy of integrated circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1069-1074.
[3] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. 2012. Logic encryption: A fault analysis perspective. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 953-958.
[4] Y. Shen and H. Zhou. 2017. Double dip: Re-evaluating security of logic encryption algorithms. In *Great Lakes Symposium on VLSI (GLSVLSI)*. 179-184.
[5] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 95-100.
[6] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu. 2016. SARLock: SAT attack resistant logic locking. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 236-241.
[7] Y. Xie and A. Srivastava. 2016. Mitigating SAT attack on logic locking. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*. 127-146.
[8] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. 2020. Removal attacks on logic locking and camouflaging techniques. *IEEE Transactions on Emerging Topics in Computing* 8, 2 (2020), 517-532.
[9] D. Sirone and P. Subramanyan. 2020. Functional analysis attacks on logic locking. *IEEE Transactions on Information Forensics and Security* 15 (2020), 2514-2527.
[10] A. Rezaei, Y. Shen, and H. Zhou. 2020. Rescuing logic encryption in post-SAT era by locking & obfuscation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 13-18.
[11] N. Limaye, S. Patnaik, and O. Sinanoglu. 2021. Fa-SAT: Fault-aided SAT-based attack on compound logic locking techniques. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1166-1171.
[12] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou. 2019. SigAttack: new high-level SAT-based attack on logic encryptions. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 940-943.
[13] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1601-1618.
[14] H. Zhou, A. Rezaei, and Y. Shen. 2019. Resolving the trilemma in Logic encryption. In *IEEE International Conference on Computer Aided Design (ICCAD)*. 1-8.
[15] F. Brglez and H. Fujiwara. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. 677-692.
[16] S. Yang. 1991. Logic synthesis and optimization benchmarks user guide version 3.0. In *MCNC International Workshop on Logic Synthesis*.