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

Raheel Afsharmazayejani, Hossein Sayadi, and Amin Rezaei

California State University Long Beach

Long Beach, CA, USA

Great Lakes Symposium on VLSI (GLSVLSI)

2022

The Logic Encryption Game

Random Logic Encryption

Anti-SAT

Stripped Functionality Logic Locking

Bilateral Logic Encryption

Distributed Logic Encryption

New

SAT Attack

AppSAT Attack

Functional Analysis Attack

Fa-SAT Attack

Attacker model: Encrypted netlist + Activated IC + Scan chain access

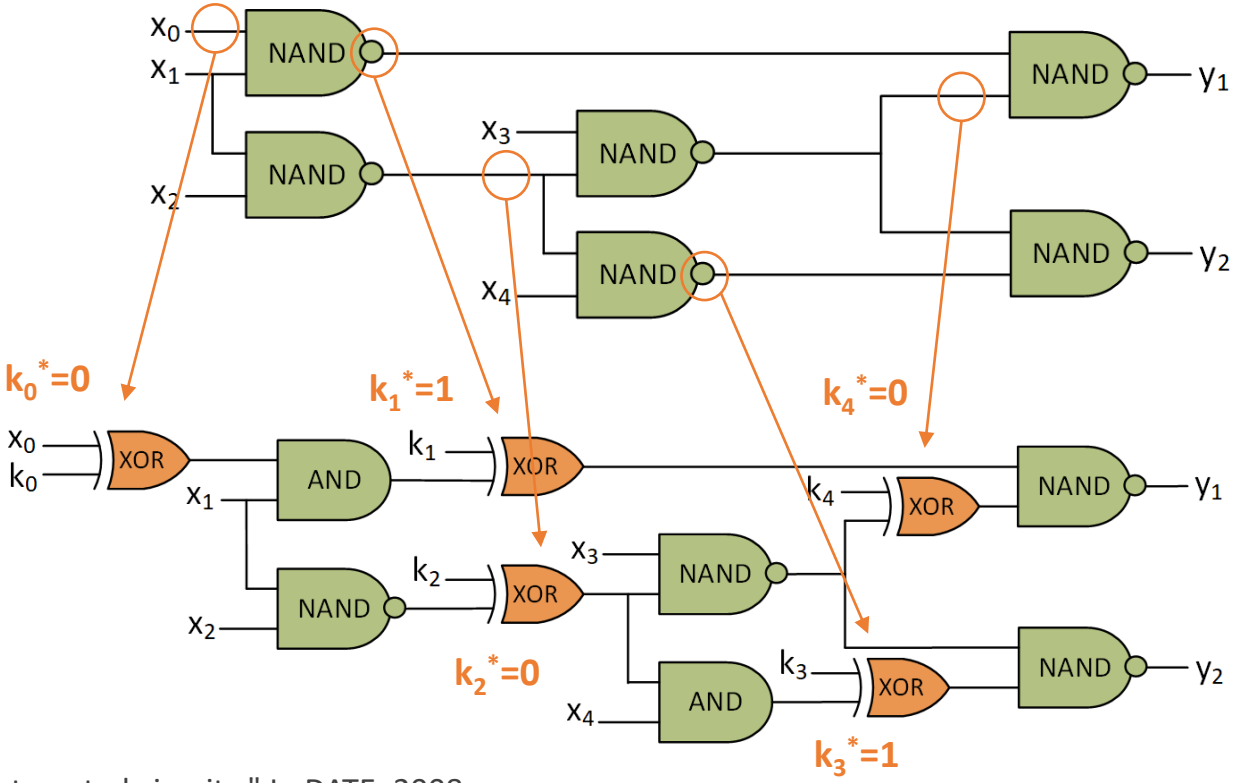
Random Logic Encryption (RLE)*

Original netlist f(x)

$$\exists \boldsymbol{k}^*: \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{k}^*) \equiv \boldsymbol{f}(\boldsymbol{x})$$

$$\boldsymbol{k}^* = k_4{}^* k_3{}^* k_2{}^* k_1{}^* k_0{}^* = 01010$$

Encrypted netlist g(x,k)



* J. A. Roy et al.,"EPIC: Ending piracy of integrated circuits," In DATE, 2008.

SAT Attack*

Attacker model:

- ✓ Encrypted netlist
- ✓ Activated IC
- ✓ Scan chain access

Algorithm 1: SAT attack

Input: Locked circuit $g(x, k)$ and original function $f(x)$
Output: Correct key k^* such that $g(x, k^*) \equiv f(x)$
while $\hat{x} = SAT(g(x, k_1) \neq g(x, k_2))$ **do**
  $g(x, k_1) = g(x, k_1) \wedge (g(\hat{x}, k_1) = f(\hat{x}));$
  $g(x, k_2) = g(x, k_2) \wedge (g(\hat{x}, k_2) = f(\hat{x}));$
$k^* = SAT(g(x, k_1));$

DIP

Goal: Each DIP excludes at least one wrong key.

The truth: Each DIP may exclude many wrong keys.

* P. Subramanyan et al., "Evaluating the security of logic encryption algorithms," In HOST, 2015.

Anti-SAT*, AppSAT&, SFLL%, and FAA$

Two tips:

1) Point function defenses like Anti-SAT are SAT-hard, but they are vulnerable to Approximate SAT (AppSAT) attacks that report an approximately correct key with low error.

2) Stripped Functionality Logic Locking (SFLL) as the strongest version of the defensive mechanism by 2017, is both SAT-hard and AppSAT-hard, but it is vulnerable to structural attacks like Functional Analysis Attack (FAA) that extracts the original netlist from the encrypted one.

* Y. Xie et al., "Mitigating SAT attack on logic locking," In CHES, 2016.
& K. Shamsi et al., "AppSAT: Approximately deobfuscating integrated circuits," In HOST, 2017
% M. Yasin et al., "Provably-secure logic locking: From theory to practice," In CCS, 2017.
$ D. Sirone et al., "Functional analysis attacks on logic locking," In IEEE Trans. on Information Forensics and Sec., 2020.

Logic Complexity (LC) vs Error Number (EN)

The LC is the minimum number of DIPs that are required to be checked under the SAT attack to reveal the correct key.

Tip: The higher the LC of an encryption approach, the more secure it is against the SAT attack.

The error of a key is the number of input patterns in which there is an inconsistency between the output of the original netlist and the encrypted one.

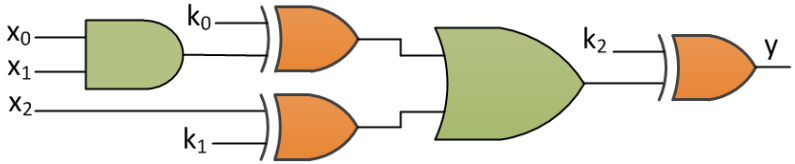
Accordingly, the EN of the whole netlist is the minimum error among all the wrong keys.

Tip: The higher the EN of an encryption approach, the more secure it is against approximate SAT attacks.

Error Matrix (EM)

Activated IC:

Encrypted Netlist:



$X_i \equiv x_2x_1x_0 = i$ $K_j \equiv k_2k_1k_0 = j$

	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
X_0	0	1	1	1	1	0	0	0
X_1	0	1	1	1	1	0	0	0
X_2	0	1	1	1	1	0	0	0
X_3	0	1	0	0	1	0	1	1
X_4	1	1	0	1	0	0	1	0
X_5	1	1	0	1	0	0	1	0
X_6	1	1	0	1	0	0	1	0
X_7	1	1	1	0	0	0	0	1

Structural Complexity (SC)

Suppose a class S of encrypted netlists in which:

First, any two netlists in S are structurally indistinguishable.

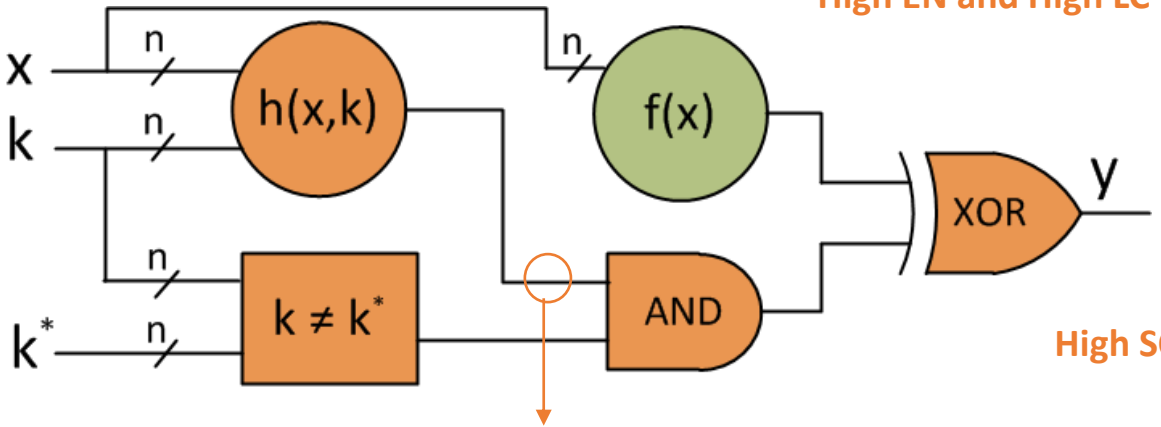
Second, given any netlist in S in which a wrong key is inserted, structurally extracting the original netlist from the encrypted one is exponentially hard with respect to the key size.

The SC is the size of the corresponding encryption class S.

Tip: The higher the SC of an encryption approach, the more secure it is against the structural attacks like FAA.

Bilateral Logic Encryption (BLE)*

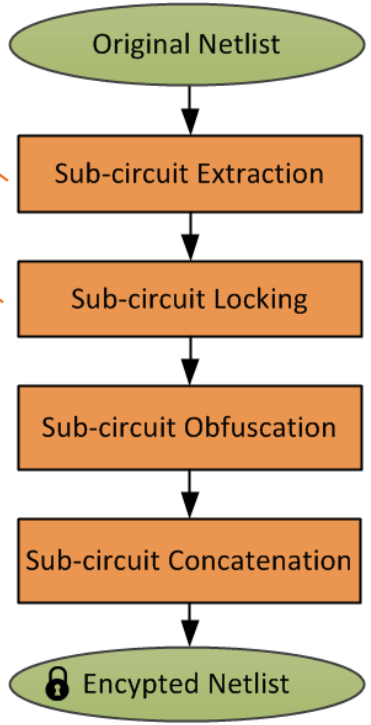
Standard locking:



High CR and High AR

High EN and High LC

High SC



* A. Rezaei et al., "Rescuing logic encryption in post-SAT era by locking & obfuscation," In DATE, 2020.

Fa-SAT Attack*

Idea:

1) Create a less secure but meaningful encrypted netlist by inserting single "stuck-at" faults at each signal.

2) Run the SAT attack on the faulty netlist and check the correctness of the reported key (if any) with random sampling.

Algorithm 2: Fa-SAT attack [13]

Input: Locked circuit $g(x, k)$ and original function $f(x)$
Output: Correct key k^* such that $g(x, k^*) \equiv f(x)$
for *each line* $i \in g(x, k)$ **do**
    // insert stuck-at-0 fault on line i.
    $g_{faulty}(x, k) = ST^0(g(x, k), i))$;
    $k^* = SAT_{timeout=90s}(g_{faulty}(x, k), f(x))$;
    // check the reported key by random sampling.
    if $g(x, k^*) \equiv f(x)$ **then**
        return k^*;
    // insert stuck-at-1 fault on line i.
    $g_{faulty}(x, k) = ST^1(g(x, k), i))$;
    $k^* = SAT_{timeout=90s}(g_{faulty}(x, k), f(x))$;
    // check the reported key by random sampling.
    if $g(x, k^*) \equiv f(x)$ **then**
        return k^*;

* N. Limaye et al., "Fa-SAT: Fault-aided SAT-based attack on compound logic locking techniques," In DATE, 2021

Distributed Logic Encryption (DLE)

Problem statement:

Original netlist $f: B^n \rightarrow B$

Locked netlist $g: B^{2n} \rightarrow B$ s.t. $\exists k^*: g(x, k^*) \equiv f(x)$

Obfuscated netlist s: $B^{3n} \rightarrow B$ s.t. $\exists p^*: s(x, k, p^*) \equiv g(x, k)$

Distributed locking:

Locked netlist must be secure against SAT, AppSAT, and Fa-SAT.

Distributed obfuscation:

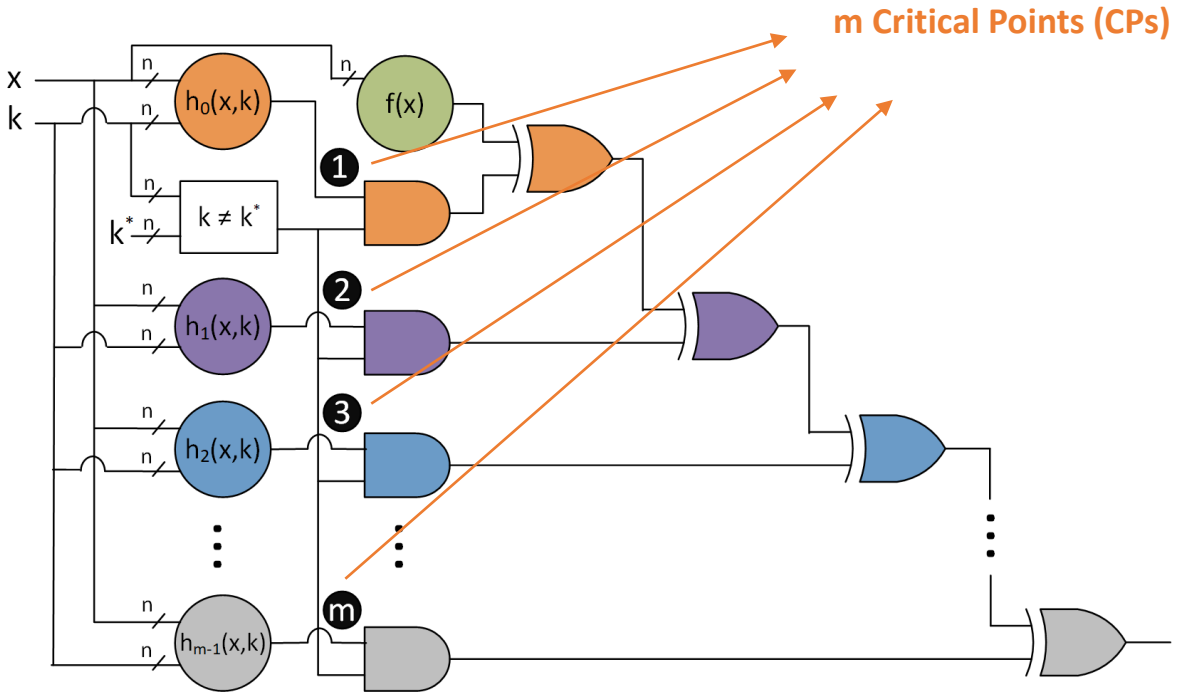
Obfuscated netlist must be secure against structural attacks like FAA.

Distributed Locking - Conformity Feature

Conformity feature:

If we use m h-functions, the distributed locking must have exponentially high LC and exponentially high EN with respect to the input size n.

Tip: In this case, the locking scheme will be secure against SAT and AppSAT.

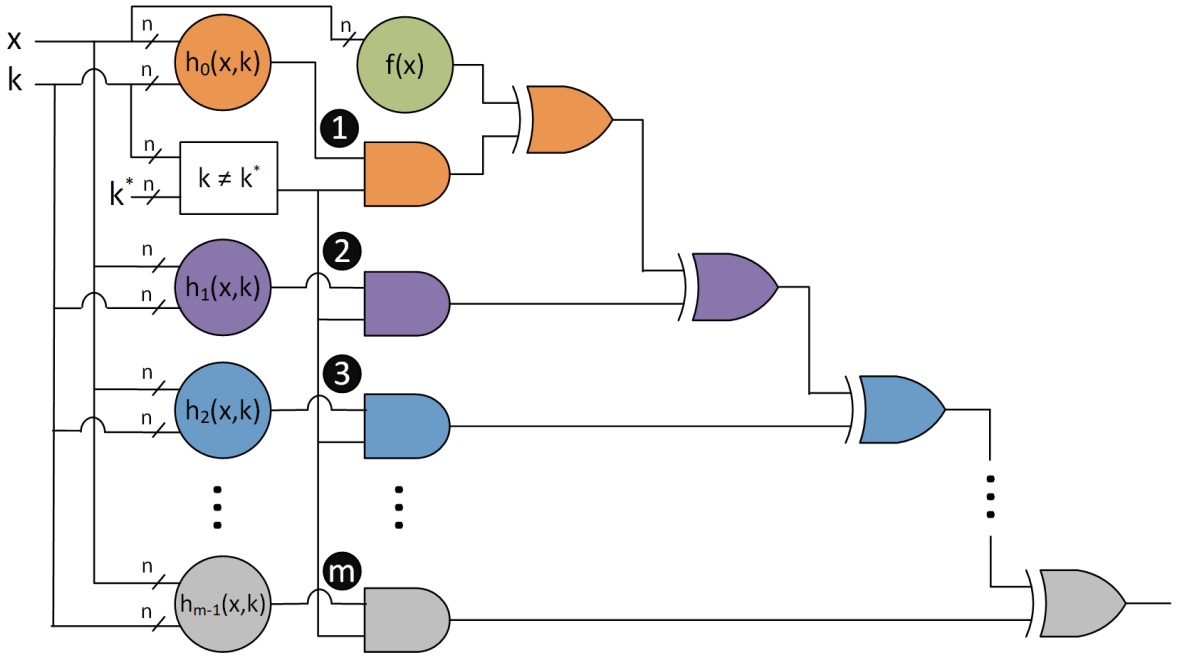


Distributed Locking - Mutuality Feature

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.

Tip: In this case, the locking scheme will be secure against Fa-SAT.



Distributed Locking - Implementation

$$\forall\, n = 2k,$$
$$\forall\, m = 2k + 1, 3 \le m < n,$$
$$\forall\, i \in \{0, 1, \dots, m-1\},$$

$$h_i(x,k) = \bigvee \bigwedge \forall\, j \in \left\{0, 1, \dots \frac{n}{2} - 1\right\}, (x_{2j} \oplus k_{2j}) \oplus \left(x_{2j+1} \oplus k_{2j+1}\right)$$

$$\bigwedge \forall\, \hat{k}_w \in White(i), k \neq \hat{k}_w$$

$$\bigvee \forall\, \hat{k}_b \in Black(i), k = \hat{k}_b$$

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

$$White(i) = \{k | \forall d \in \left\{\frac{2^n}{2}, \dots, 2^n\right\}, k = \hat{k}_d \notin Black(i)\}$$

Theorem:

If the distributed locking adopts this h-function formula, it is secure against SAT, AppSAT, and Fa-SAT attacks.

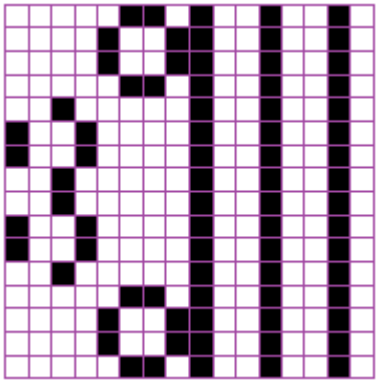
Distributed Locking - Example

Setup: $n = 4,\ m = 3,\ k^* = \hat{k}_1 = 0001$

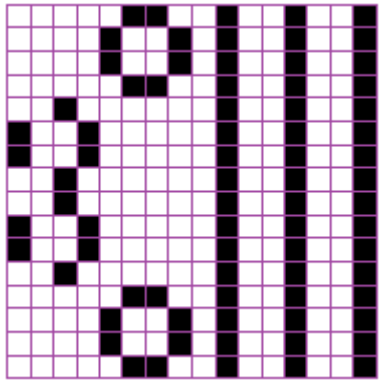
Note: Black cells depict 1 and white cells depict 0.

LC = EN = $2^{\frac{n}{2}}$ = 4

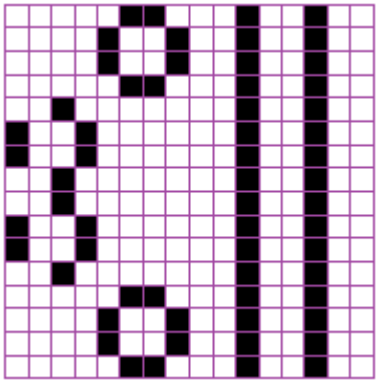
Covers conformity feature →
Secure against SAT and AppSAT



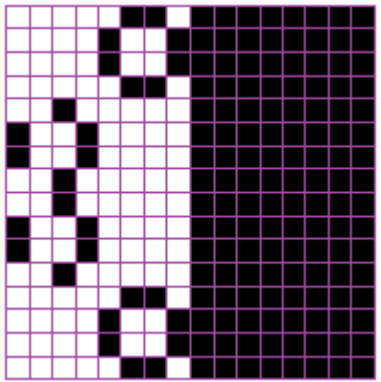
EM of standard locking with h_0



EM of standard locking with h_1



EM of standard locking with h_2

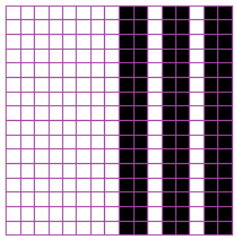


EM of distributed locking with h_0, h_1, and h_2

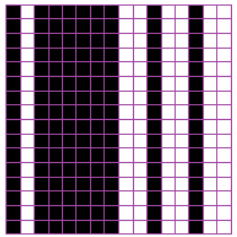
Distributed Locking - Example Cont.

∃ more than $2^{\frac{n}{2}} = 4$ wrong keys with zero error upon insertion of a stuck-at-0 or a stuck-at-1 in each CP.

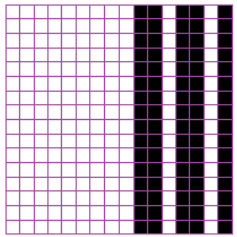
Covers mutuality feature → Secure against Fa-SAT



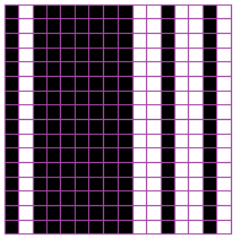
EM of distributed locking with stuck-at-0 fault in CP3



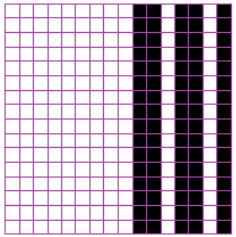
EM of distributed locking with stuck-at-1 fault in CP3



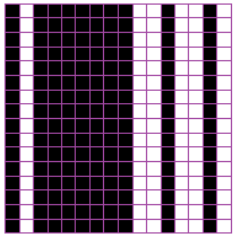
EM of distributed locking with stuck-at-0 fault in CP2



EM of distributed locking with stuck-at-1 fault in CP2



EM of distributed locking with stuck-at-0 fault in CP1



EM of distributed locking with stuck-at-1 fault in CP1

Distributed Obfuscation

Goal: Hiding four zones of the distributed locking

Real signal obfuscation: Choose random signal inside each zone:

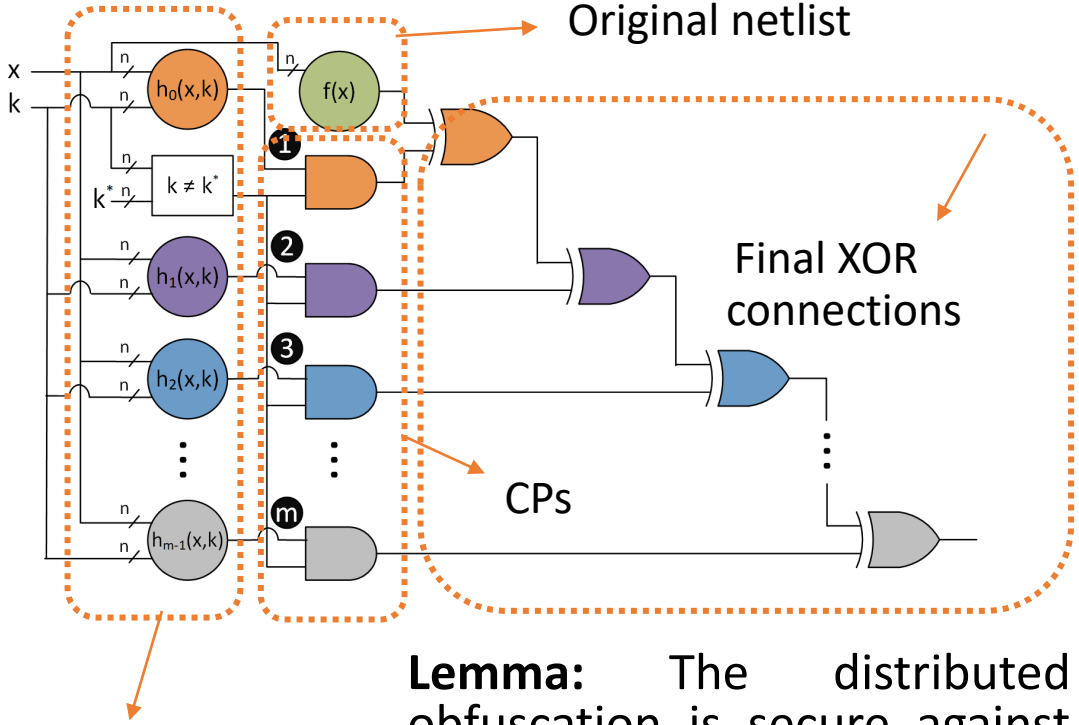
Fan-in of AND/NAND? → Add key bit controlled OR gate.

Fan-in of OR/NOR? → Add key bit controlled AND gate.

Dummy signal obfuscation: Choose random signal inside each zone and a random target gate outside the zone:

Target gate is AND/NAND? → Add key bit controlled OR gate.

Target gate is OR/NOR? → Add key bit controlled AND gate.



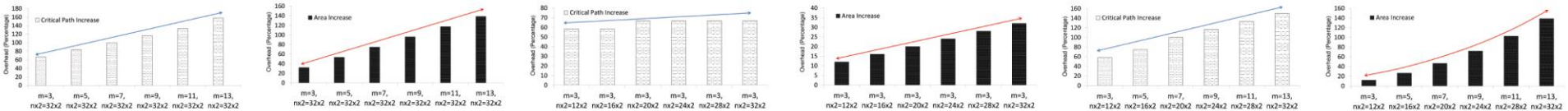
Lemma: The distributed obfuscation is secure against structural attacks like FAA.

Experimental Results

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)

DLE average overhead



Conclusion

DLE: Distributed Logic Encryption on Hardware IP Protection

Low-overhead: Polynomial area and linear performance overheads

High-security: Exponential security gain against SAT guided and structural attacks

The battle continues…

Thank You!

Raheel Afsharmazayejani

Hossein Sayadi

Amin Rezaei