SeqL: Secure Scan-Locking for Sequential Circuits

Seetal Potluri
Department of ECE
North Carolina State University
Raleigh, U.S.
spotlur2@ncsu.edu

Aydin Aysu
Department of ECE
North Carolina State University
Raleigh, U.S.
aaysu@ncsu.edu

Akash Kumar
Department of Computer Science
Technical University of Dresden
Dresden, Germany
akash.kumar@tu-dresden.de

***Abstract*—Existing oracle-guided and oracle-less attacks are known to successfully decrypt a *functionally correct key* of a locked combinational circuit. It is possible to extend these attacks to sequential circuits through the scan-chain by selectively initializing the combinational logic and analyzing the responses. In this paper, we propose *SeqL*, which achieves functional isolation and locks selective functional-input/scan-output pairs, thus rendering the decrypted key *functionally incorrect*.**

We conduct a formal study of the scan-locking problem and demonstrate automating our proposed defense on any given sequential circuit. We show that *SeqL* hides functionally correct keys from the attacker, thereby increasing the likelihood of the decrypted key being *functionally incorrect*. When tested on sequential benchmarks (ITC'99) and pipelined combinational benchmarks (ISCAS'85, MCNC), *SeqL* gave 100% resilience to state-of-the-art oracle-guided as well as oracle-less attacks. Therefore, *SeqL* is generic and robust defense against logic locking attacks on sequential circuits.

***Index Terms*—Logic Locking, Oracle-guided attacks, Oracle-less attacks, Scan-Locking**

I. Introduction

Dramatic increases in fabrication costs have shifted the semiconductor business to a contract foundry model, where leading-edge design houses outsource fabrication, assembly and testing steps to offshore facilities. However, the lower operational costs come at the expense of compromised security in terms of integrated circuit (IC) piracy, overbuilding, and counterfeiting in the untrusted offshore foundry [1]–[3]. United States government has recently declared trade war against loose intellectual property rights (IPR) protection laws in Asia, and declared that this has forced technology transfer and alleged IP theft [4]. Subsequent tariffs imposed by the U.S. government on semiconductor imports, will cost millions of dollars annually [5]. With the recent U.S. ban on Huawei, the problem has become more challenging than ever.

Reverse engineering the mask reveals the netlist, while analyzing the scan data either on an activated IC or at an outsourced tester reveals critical design information. This raises concerns regarding piracy, reverse engineering, overproduction, IPR violation, and hardware trojan insertion and counterfeit electronics [6]–[8]. Thus, it is imperative to consider trust in early phases of chip design, to secure it in the entirety of supply chain. Camouflaging and Logic-locking are two such holistic solutions that were touted to address all the aforementioned threats. While Camouflaging has reached industry practice, e.g., Mentor Graphics TrustChain Camo technology [9], however the powerful Boolean Satisfiability attack [8] (popularly known as the SAT-attack) and its variants have troubled logic-locking from doing so. This paper addresses the concerned challenges and propose *SeqL*, that has potential for mainstream industry practice.

A. Related work

Logic locking uses a low-overhead combinational chip-locking system to combat these issues [3]. Its purpose is to generate circuits that will reveal the correct output if and only if the secret key is entered correctly. Basically, logic locking encrypts selective nodes inside the circuit by adding a logical gate (such as XOR/XNOR/OR/AND/MUX, etc.) with one input driven by a secret key. Hence, the function of the circuit will not change if this key is entered correctly. But if the key is incorrect, the function can alter.

The first wave of logic locking techniques [3], [10]–[13] are vulnerable to SAT-attack [8]. Several defenses were then proposed to mitigate SAT-attack, such as *Anti-SAT* [14], *SARLock* [15] and *Cyclic Obfuscation* [16], but they have failed to address the vulnerability to *AppSAT* [17], *Double-DIP* [18], *CycSAT* [19], *HackTest* [20] *BeSAT* [21] and machine-learning [22] attacks. While [23] proposes a new cyclic logic locking technique to defend *CycSAT* [19], *TT-Lock* [24] and Secure Function Logic Locking (SFLL) [25] were the only locking schemes that were broadly resilient to these attacks, but they recently failed against functional analysis of logic locking (FALL) [26] and SMT [27] attacks.

The fundamental assumption of all these successful attacks is that the adversary can control and observe the inputs of a combinational circuit. Most real-world circuits are, however, sequential in nature. Indeed, it is possible to launch the SAT-attack or its variants on sequential circuits, if this fundamental assumption is true. Although model checking [23], [28] aims an attack without scan access, decryption is only partial. *Hence scan-chain access is necessary for full controllability/observability and thereby to decrypt the complete key.* Attacks through the scan access are feasible in practice because most IC vendors do not deactivate scan ports to allow post-silicon debug. The *HackTest*-attack [20], e.g., uses scan-chains and Automatic Test Pattern Generation (ATPG) information to launch the attack.

To address this issue, encrypt flip-flop (*EFF*) based scan-locking [29] was recently proposed as a defense to SAT-attack on sequential circuits. In *EFF*, flip-flip outputs (FOs) are locked, and the locked FOs drive both the combinational logic as well as the next flip-flops in the scan chain. *ScanSAT* [30], however, showed that it is possible to convert the *EFF*-style sequentially locked instance to a locked combinational instance, and thereby decrypt the entire sequential key using SAT-attack. Although, existing works like *SARLock* [15] and *AppSAT* [17] consider sequential circuits, they target unrolled versions and do not consider FO locking. Therefore, there is a clear need for a fundamentally different logic locking technique to enable practical, secure and scalable logic encryption.

B. Contributions

The main contributions of this paper are as follows:

1) We identify there is 100% correlation between flip-flop input (FI) locking and functional output corruption;
2) Exploiting this property, we propose *SeqL*, that has two properties: (a) it isolates functional path from the locked scan path; (b) it locks FIs and causes functional output corruption;
3) Since the attack is launched through the scan-chain, the returned key will be scan-correct. *SeqL* hides majority of the scan-correct keys which are functionally correct, thus maximizing the probability of the decrypted key being *functionally incorrect*;
4) The small area, timing and energy-per-toggle overheads of *SeqL* and its ease of implementation makes it attractive for industry practice.

C. Scan-Locking [29] & State-of-the-Art Attacks [30] [20] [26]

In *EFF* technique [29], flip-flops on the non-critical timing paths of the sequential circuits are selected, and XOR/XNOR-type key gates are added to lock the Q-outputs. The locked Q-output of a scan-flip-flop reaches combinational logic, as well as the scan-input of the next flip-flop in the scan chain. Figure 1(a) shows a sample sequential circuit with all flip-flops encrypted using *EFF*-style scan-locking. In Figure 1(a):

- G_0 and G_1 are the primary inputs;
- G_6 is the primary output;
- G_2 and G_4 are flip-flop inputs (FIs);
- G_3 and G_5 are FOs;
- SI and SO are the circuit's scan-input port and scan-output port respectively;
- ck_0 and ck_1 are the logic locking key bits;
- fok_0 is the key bit used to lock the FO G_5, using an XOR-type key gate;
- fok_1 is the key bit used to lock the FO G_3, using an XNOR-type key gate; and
- Since these key gates are used to lock FOs, they are referred to as FO key gates in rest of the paper.

ScanSAT [30] shows that it is possible to convert this scan-locked instance to the logic locked instance shown in Figure 1(b). The basic idea is that each FI, undergoes a series of inversions along the scan-chain due to the locked XOR/XNOR-gates. In effect, each FI signal propagates through a corresponding portion of the locked XOR/XNOR-chain along the scan-chain. Thus, for each FI, corresponding flip-flop can be removed, and

- equivalent locked scan-input XOR/XNOR-chain can be appended to the primary input, and
- equivalent locked scan-output XOR/XNOR-chain can be appended to the primary output,

thus arriving at the *scan-unrolled* equivalent combinational circuit shown in Figure 1(b) where:

- $SI(G_3)$ is the scan-input-bit corresponding to flip-flop 1, in the scan-mode of operation;
- $SI(G_5)$ is the scan-input-bit corresponding to flip-flop 2, in the scan-mode of operation;
- $ESO(G_2)$ is the locked-scan-output bit corresponding to flip-flop 1, in scan-mode of operation; and
- $ESO(G_4)$ is the locked-scan-output bit corresponding to flip-flop 2, in scan-mode of operation.

Since original scan-locked instance (shown in Figure 1(a)) and the generated logic-locked instance (shown in Figure 1(b)) are functionally equivalent, the attacker is able to launch the SAT-attack on the logic-locked equivalent instance, and recover the functionally correct sequential key. Hence, the *EFF* technique is not secure. Additionally, the FO key gate is along the functional path, thus it toggles during functional operation, wasting power without contributing to any useful computation. Therefore, there is a clear need to propose secure and cost-effective solutions for scan-locking.

Similar to *ScanSAT* [30], it is possible to extend some of the state-of-the-art attacks like *HackTest*-attack [20], functional-analysis-attacks on logic-locking (FALL)-attack [26], and SMT-attack [27]. In this paper, we evaluate *SeqL* on these aforementioned state-of-the-art attacks. The verification effort involved for cyclic combinational circuits does not scale well for complex designs [31], hence the cyclic logic obfuscations and corresponding attacks are not practically useful to industry. We therefore do not compare against this class of techniques in this paper.

D. Threat model

The design house receives locked IPs from third-party vendors, and integrates them with in-house designed IPs, and prepares the final locked netlist. After physical design is completed, the layout is sent to the foundry for fabrication. We consider a malicious foundry that offers fabrication, assembly and testing services [32]. Thus, the attacker has access to layout and mask information, and is thus able to reverse-engineer the gate-level netlist. There are two possible instances, where the attacker at the malicious foundry is able to launch the attack:

1) The attacker purchases an activated IC from the open market, and applies input as well as scan patterns, and observes output as well as scan responses in embedded deterministic test (EDT)-bypass mode. Typically, scan ports are not deactivated to facilitate debug of customer returns. The attacker exploits these active scan ports to launch the SAT-attack;
2) The attacker has access to the outsourced tester, where the activated ICs are sent for testing. In this scenario, the scan ports are clearly not deactivated, hence the attacker can place the dies in EDT-bypass mode, applies desired

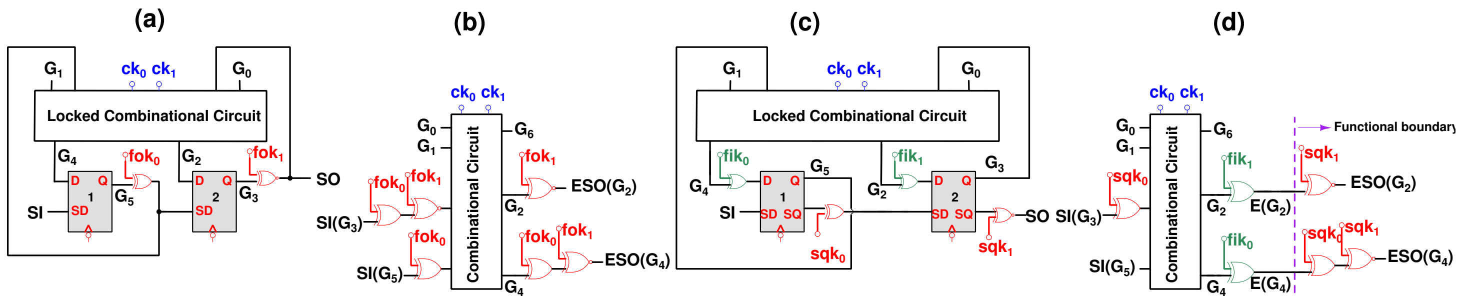


Fig. 1. (a) *EFF*-style: Sample sequential circuit with logic locking and FO locking; (b) Scan-unrolled equivalent combinational circuit for Figure 1(a); (c) *SeqL*-style: Circuit in Figure 1(a), with functional isolation and locked FIs/SQs; and (d) Scan-unrolled equivalent combinational circuit for Figure 1(c)

input as well as scan patterns and collects the outputs as well as scan response data.

There are multiple scan-locking/design-for-security (DFS) techniques in the literature. We focus on *EFF* [29] design-for-security (DFS) technique, which uses statically obfuscated scan-chains, and do not consider dynamically-obfuscated-scan (DOS) architecture proposed in [33] nor design-for-security (DFS) architecture proposed in [34]. The reason behind this choice is that the DFS technique proposed in [29] is scalable, and hence feasible for industry practice.

E. Organization

Section II provides insight into the the proposed solution. This section explains the idea of functional isolation using an abstract model. This section also explains FI locking and provides mathematical analysis of the likelihood of solver returning functionally incorrect key. Section III explains the proposed solution in detail. Results obtained using the proposed solution are provided and analyzed in Section IV. Subsequently, we make a short discussion in Section V and conclude the paper in Section VI.

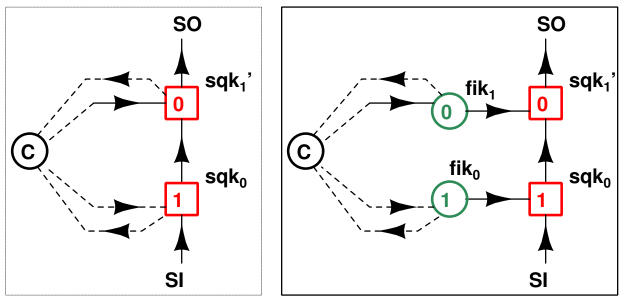
II. Solution Insight

As discussed in previous section, when SAT-attack is launched on the scan-unrolled *EFF*-style scan-locked circuit shown in Figure 1(b), the SAT solver returns the functionally correct key. In the discussion that follows, we exploit the following two principles:

1) In *EFF*-style scan-locking, the FO key-gate corresponding to the flip-flop appears *both* in the scan-input-path and scan-output-path of the flip-flop in the scan-unrolled combinational instance. Since the attacker has access only to scan-data, if functional path can be isolated from the locked scan path, functional output can be corrupted; and
2) Since the FO key gates cascade with FIs to form XOR/XNOR-chains, it is possible to obfuscate the solver by adding dummy XOR/XNOR-gates at the FIs.

Figure 1(c) shows the proposed *SeqL*-style scan-locking idea by transforming the circuit in Figure 1(a), using above principles. Figure 1(c) is different from Figure 1(a) in two ways:

- There is a separate Q and SQ, and the key gate is added at SQ (referred-to henceforth as SQ key-gate), thus leaving the functional output Q unencrypted. This is referred to as *functional isolation*;



(a) *EFF*-style scan-locking (b) *SeqL*-style scan-locking
Fig. 2. Abstract models for circuits in Figures 1(a) and 1(c)

- sqk_0 is the key bit used to lock the SQ output of flip-flop 0, using an XOR-type key gate;
- sqk_1 is the key bit used to lock the SQ output of flip-flop 1, using an XNOR-type key gate;
- Extra key gates (both of XOR type in this case) are added at FIs of both the flip-flops. fik_0 and fik_1 the FI locking key bits. These key gates are referred to as FI key gates in the rest of this paper.

Figure 1(d) shows the corresponding scan-unrolled equivalent combinational circuit. *The purple dashed line is the functional boundary. This means that the key gates to the right of this boundary (SQ key-gates) only affect scan-operation, and do not affect normal functional operation of the circuit.* This is because the attacker uses scan mode of operation, and hence observes $ESO(G_2)$ and $ESO(G_4)$.

However, the circuit's normal functional operation is purely influenced by $E(G_2)$ and $E(G_4)$, and hence the XOR/XNOR-chains (in red) cease to exist. This renders the scan-correct decrypted key, being *functionally incorrect*. When the following are inputted to the formal equivalence checker

- the combinational portion of the sequential circuit shown in Figure 1(c);
- the combinational portion of the original unencrypted sequential circuit; and
- the combination portion of this solver key K,

the result will be either *equivalent* or *different*. In the example shown in Figure 1(d), it was found that the result was *different*, or in other words *not equivalent*. Hence, by functional isolation and FI locking, functional output corruption was achieved, thus making the proposed *SeqL*-style scan-locking mechanism secure. Next subsection explains this behavior using an abstract model.

TABLE I
TRUTH TABLE OF OUR PROPOSED SCAN-LOCK IN FIGURE 1(C)

fik_1	sqk_1'	fik_0	sqk_0	Scan-Correct	Functional-Correct
0	0	0	0	TRUE	TRUE
0	0	0	1	FALSE	TRUE
0	0	1	0	FALSE	FALSE
0	0	1	1	TRUE	FALSE
0	1	0	0	FALSE	TRUE
0	1	0	1	FALSE	TRUE
0	1	1	0	FALSE	FALSE
0	1	1	1	FALSE	FALSE
1	0	0	0	FALSE	FALSE
1	0	0	1	FALSE	FALSE
1	0	1	0	FALSE	FALSE
1	0	1	1	FALSE	FALSE
1	1	0	0	FALSE	FALSE
1	1	0	1	TRUE	FALSE
1	1	1	0	TRUE	FALSE
1	1	1	1	FALSE	FALSE

A. Abstract model

Figure 2 shows an abstracted model of the sequential circuit, with the combinational logic abstracted into a source-sink circular vertex *C*, each FI key-gate abstracted into a green circular vertex, and each flip-flop key-gate abstracted into a red rectangular vertex. Figures 2(a) and 2(b) show the models corresponding to sequential circuits in Figures 1(a) and 1(c) respectively.

In the abstract model corresponding to the proposed scan-locking shown in Figure 2(b), the following are functional paths:

- $FP_0 : fik_0 \longrightarrow C$
- $FP_1 : fik_1 \longrightarrow C$

and the following are scan-out paths:

- $SP_0 : fik_0 \longrightarrow sqk_0 \longrightarrow fok_1' \longrightarrow SO$
- $SP_1 : fik_1 \longrightarrow sqk_1' \longrightarrow SO$

It can be noted from Figure 2(b), that the number of inversions for the scan-output-paths SP_0 and SP_1, are 2 and 0 respectively. Since all scan-output-paths have even inversion parity, the proposed locked circuit is *correct for scan operation*. However, when it comes to functional paths, the number of inversions for FP_0 and FP_1, are 1 and 0 respectively. Since functional path FP_0 has odd-inversion-parity, the circuit is *incorrect for functional operation*.

To understand this behavior more systematically, table I enumerates all possibilities for the scan-lock $\{fik_1, sqk_1, fik_0, sqk_0\}$ for the circuit in Figure 1(c).

The rows in this table, that show up as $TRUE$ for the scan-correct column, are the possible keys returned by the SAT-solver. There are four possible scan-correct keys, among which only one is functionally correct. Thus, *SeqL* maximizes the odds against the functionally-correct-key among the scan-correct-keys, thus increasing the likelihood of functional output corruption.

Figure 3 shows the key assignment graph (KAG) for the circuit shown Figure 1(c). The sequential key returned by the solver, as discussed previously, corresponds to the second leaf from the left. Since this leaf is a functional incorrect key, the technique is able to achieve functional output corruption.

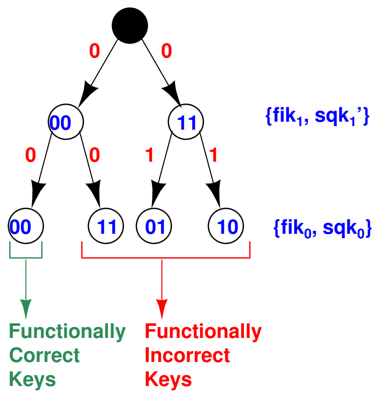


Fig. 3. Key Assignment Graph (KAG) for circuit in Figure 1(c). KAG is a binary tree, the leaves of which correspond to the rows in Table I for which the scan-correctness column is TRUE

In this example, odds against the functionally correct key is $p = \frac{3}{4} = 0.75$.

To summarize, the inferences on the proposed *SeqL*-style scan-locking:

1) It is possible to achieve functional output corruption, by isolating the functional path from the locked scan path;
2) During scan-mode of operation (used by the attacker), the FI XOR/XNOR-type key gates cascade with the unrolled XOR/XNOR-chain of the scan path, and thereby obfuscating the SAT-solver;
3) If the sequential circuit is applied with the key returned by the SAT-solver, the circuit is guaranteed to function correctly during scan operation, but not during *functional operation*;
4) The probability of decrypting *functionally incorrect key*, is higher than otherwise;
5) The combinational portion of the returned solver key, excluding the FIs, i.e., $\{ck_0, ck_1\} = \{1, 0\}$, is correct, while the portion of the key corresponding to the flip-flops and FIs is incorrect. We shall see in Section IV, that this is true in general, hence it is sufficient to lock the FIs and flip-flops (which contribute to security), thus saving area and power; and
6) In Table I, there are four rows which are TRUE for functional-correctness. Among them, only one row is TRUE for scan-correctness, and remaining three rows are FALSE for scan-correctness. This indicates that the proposed scan-locking mechanism **hides** three out of four functional-correct keys from the attacker.

B. Analysis

This section formally analyzes the security of logic locking and proves that if *SeqL* is used to lock n flip-flops in the sequential circuit, then the odds against the functionally-correct-key among the scan-correct-keys equals $1 - \frac{1}{2^n}$, assuming the attack is launched in EDT-bypass mode.

Definition 1: Given a FI, SQ pair $\{fik_i, sqk_i\}$, there are 4 possible assignments $\{00, 01, 10, 11\}$.

Definition 2: Let n be the number of locked FI, SQ pairs.

Definition 3: $KAG = (V, E)$ be a vertex-labelled edge-weighted directed graph, where the vertices correspond to FI, SQ pairs and the edges correspond to inversion parity. The direction of edges is opposite to the scan-out-path direction.

Definition 4: In KAG, the children of every vertex at depth i from the root correspond to i^{th} flip-flop from the end of the scan-out-path. All node and edge assignments are performed to ensure scan-correctness.

Definition 5: KAG is a tree, whose root vertex is a dummy node, with exactly two children 00 and 11.

Definition 6: The labels on the vertices in KAG are $00, 01, 10$ or 11, corresponding to $\{fik_i, sqk_i\}$, $\{fik_i, sqk_i^{'}\}$, $\{fik_i^{'}, sqk_i\}$ or $\{fik_i^{'}, sqk_i^{'}\}$ depending on whether FI key-gate, SQ key-gate combination is {XOR, XOR}, {XOR, XNOR}, {XNOR, XOR} or {XNOR, XNOR} respectively.

Definition 7: 00 and 11 are even-parity vertices, whereas 01 and 10 are odd-parity vertices. The children of 00 and 01 are even-parity vertices, i.e., 00 and 11. The children of 10 and 11 are odd-parity vertices, i.e., 01 and 10. Hence every non-root vertex has exactly 2 children.

Definition 8: The possible weights on the edges in KAG are 0 or 1, which signifies parity. The parity of an edge signifies the presence/absence of signal-inversion at the child flip-flop, which is same as the parity of the corresponding child vertex.

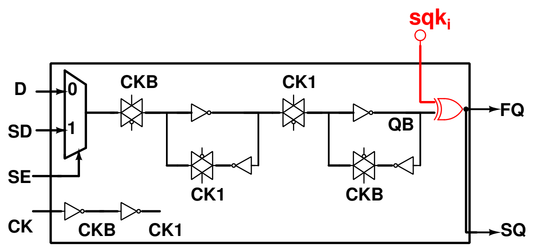
Definition 9: inv_k equals 0 or 1, depending on whether k^{th} flip-flop along the scan-chain from the scan-output is locked with an XOR or $XNOR$ key-gate respectively.

Theorem 1: Parities of left and right edges of a vertex are identical.
Proof: Assume vertex v_i in KAG at depth i. In order to ensure scan-correctness, $(fik_i \oplus sqk_i \oplus inv_i) \oplus \sum_{k=1}^{i-1}(sqk_k \oplus inv_k)$ should equal 0. If $\sum_{k=1}^{i-1}(sqk_k \oplus inv_k)$ equals 0, $(fik_i \oplus sqk_i \oplus inv_i)$ becomes 0 (possible children of v_i are 00 and 11, in both cases parity of edge is 0).
On the other hand, if $\sum_{k=1}^{i-1}(sqk_k \oplus inv_k)$ equals 1, $(fik_i \oplus sqk_i \oplus inv_i)$ becomes 1 (possible children of v_i are 01 and 10, in both cases parity of edge is 1). Thus, parity of left and right edges of a vertex are identical, hence the proof.

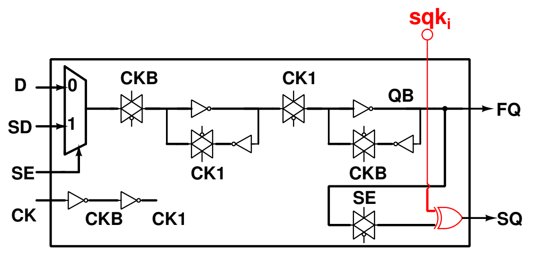
Theorem 2: KAG is a binary tree.
Proof: Root vertex has exactly two children. Additionally, every non-root vertex has exactly two children. Since every vertex in KAG has exactly two children, KAG is a binary tree, hence the proof.

Definition 10: Number of scan-correct keys equals to the number of leaves in $KAG = 2^n$.

Theorem 3: There is exactly one functionally correct leaf in KAG.
Proof: The path from the root to a functionally correct leaf should have all 00 nodes. There is exactly one such leaf in KAG, hence the proof.



(a) Encrypt flip-flop (*EFF*)



(b) *SeqL* flip-flop

Fig. 4. Flip-Flop variants for Scan-Locking

Definition 11: Let p be the odds against the functionally-correct-key among the scan-correct-keys $= \frac{2^n - 1}{2^n} = 1 - \frac{1}{2^n}$

III. AUTOMATING SEQL DEFENSE

So far, we have seen the effectiveness of *SeqL* in defending attacks on scan locking. This section shows how to automate *SeqL*, so that the technique can be practically deployed on large circuits.

Objective: Lock selective scan flip-flops (FI, SQ pairs) such that functional output corruption is achieved, while area-overhead is minimized.

Solution: The likelihood of functional output corruption is maximized with increase in $p = 1 - \frac{1}{2^n}$, whereas area-overhead increases linearly with n. Hence, the chances of functional output corruption increases very quickly with n, with minimal increase in area overhead. We exploit this principle to iteratively lock scan flip-flops (FI, SQ pairs) from the end of the scan-chain(s), until functional output corruption is achieved.

Thus, the proposed scan-locking solution, i.e., *SeqL* has two parts:

1) An functionally-isolated scan-locked flip-flop design.
2) An iterative FI/SQ locking algorithm.

A. Functionally-isolated scan-locked flip-flop design

Define the sequential key to be $K = \{K_c, K_{fi}, K_{sq}\}$, where K_c, K_{fi} and K_{sq} are portions of the key that lock the combinational logic (excluding the FIs), the FIs and the SQs respectively. In *EFF* technique, all these components influence the sequential circuit's normal functional operation. Figure 4(a) shows the *EFF*-style scan-locking scheme, where the FO key gate output, is broadcasted to *Scan-Q* (referred to as SQ in the figure), as well as *Functional-Q* (referred to as FQ in the figure). The proposed isolation-based scan-locking is shown in Figure 4(b), which isolates the functional path from the locked scan path. Hence, the SQ key gate locks

Algorithm 1: Iterative key pushing algorithm for pipelined logic-locked combinational circuits

```
Input: $C$
while $C^{'} = C$ do
    Identify a combinational key gate pair $k_c$, an
      unvisited FI/SQ pair $k_b$ and mark corresponding $k_b$
      as visited ;
    Push $k_c$ to $k_b$;
    Run SAT-solver and update $K_{fi}$, $C^{'}$;
end
Result: $C^{'}$, $K_{fi}$
```

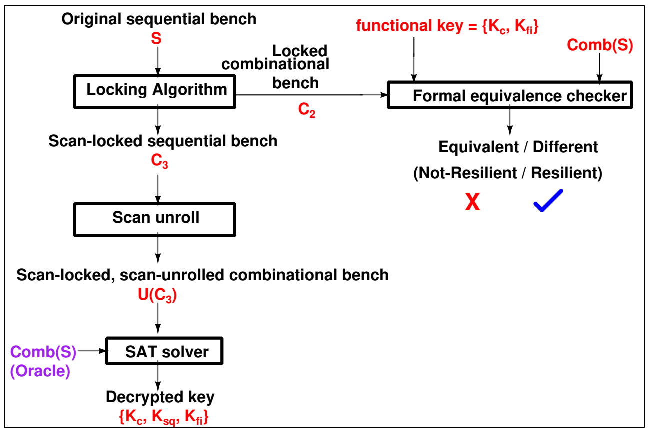
only SQ and has no influence on FQ. Thus, in the proposed *SeqL* technique, only K_c and K_{fi} influence (while K_{sq} has no effect on) the sequential circuit's normal functional operation. This assists in returning the *functionally incorrect key*, thus aiding in functional output corruption when applied with the key returned by the SAT-solver.

There is an additional transmission gate added to this structure in the scan path to avoid toggling of the locking key gate along the scan path. Although this adds 2 extra transistors per flip-flop, the overhead is marginal compared to the benefit of savings obtained in Energy-Per-Toggle (EPT) of the flip-flop during normal functional operation. The comparison of area, timing and EPT of the *EFF* as well as SeqL flip-flops are provided later in Section IV.

B. Iterative key pushing algorithm (IKPA) for pipelined combinational circuits

We shall first evaluate adding key gates on functional boundary on pipelined combinational circuits, which are already logic-locked. Algorithm 1 shows the iterative key gate pushing algorithm, that takes a logic-locked combinational circuit with pipeline stages both at its inputs and outputs. Since the circuit already has key gate overhead, to avoid any further overhead, the algorithm iteratively pushes some of the key gates inside combinational logic to the boundary.

Figure 5 shows the resilience verification flow used iteratively in *SeqL*. Algorithm 1 generates C_3 shown in Figure 5. The SAT-based key generation solver takes the original combinational circuit (**the oracle**, shown in purple in Figure 5), and scan-locked scan-unrolled combinational circuit as inputs and outputs the logic locking key $K = \{K_c, K_{sq}, K_{fi}\}$. Subsequently, the *lcmp* solver takes the combinational portion of the generated key i.e., $\{K_c, K_{fi}\}$, original combinational circuit and the combinational portion of the sequentially locked circuit as inputs, and verifies whether the corresponding circuits are formally equivalent. We measure the success of the *IKPA* algorithm using formal equivalence checking. If the returned solver key makes the two circuits *different* or in other words *not equivalent*, then we are successful i.e., *functional output corruption is achieved*. Additionally, even with *SeqL*, we have found that in all the cases, the decrypted K_c is correct. Hence, it is only the K_{fi}, that causes functional output corruption. We shall see the detailed results in Section IV. With this important observation, we proceed to the iterative boundary locking algorithm for sequential circuits.



Original sequential bench
S
functional key = {Kc, Kfi}
Comb(S)
Locked combinational bench
Locking Algorithm
Formal equivalence checker
C2
Scan-locked sequential bench
C3
Equivalent / Different
(Not-Resilient / Resilient)
X
✓
Scan unroll
Scan-locked, scan-unrolled combinational bench
U(C3)
Comb(S) (Oracle)
SAT solver
Decrypted key
{Kc, Ksq, Kfi}

Fig. 5. Resilience verification flow

Algorithm 2: Iterative boundary locking algorithm for sequential circuits

```
Input: $C$, $K_{sq}$
while $C^{'} = C$ and $|K_{fi}| + |K_{sq}| < \gamma$ do
    Identify an unvisited FI/SQ pair and mark the
      corresponding pair as visited ;
    Add key-gates at corresponding FI/SQ pair;
    Run SAT-solver and update $K_{fi}$, $C^{'}$;
end
Result: $C^{'}$, $K_{fi}$
```

C. Iterative boundary locking algorithm (IBLA) for sequential circuits

Unlike pipelined logic-locked combinational circuits, there are no already existing key gates. Hence, FI/SQ locking or in other words, boundary locking has to be done afresh. Since the higher the number of inserted key gates at the FI/SQ boundary, the higher the area overhead, we make this parameter as user-configurable.

Let γ be this user-configurable parameter. This *IBLA* algorithm runs iteratively until functional output corruption is achieved or $|K_{fi}|+|K_{sq}| > \gamma$. Algorithm 2 takes a sequential benchmark as input, and iteratively adds XOR/XNOR-type key-gates at unvisited FI/SQ pairs, until functional output corruption is achieved or $|K_{fi}| + |K_{sq}| > \gamma$. Since the implementation is simple and security is achieved with low overheads, this is attractive for industry practice.

IV. Experimental Evaluation

We validate the security of *SeqL* against a multitude of state-of-the-art attacks and quantify its reduced overheads compared to prior work. This analysis confirms our claims on genericness, robustness, and scalability of *SeqL*.

In order to perform SAT-attack resilience evaluation, we have used *sld* key generation solver and *lcmp* formal-equivalence checkers provided by [8]. Both the solvers use the *.bench* format for the input benchmark circuit. In all the subsequent experiments, we have used the open-source *.bench* designs for ITC'99 sequential benchmarks available at [35] and logic-locked ISCAS'85, MCNC combinational benchmarks provided by [8]. Algorithms 2 (*IBLA*) and 1 were used for scan-locking the sequential benchmarks and pipelined combinational benchmarks, respectively. Both the locking algorithm were implemented in *Perl*. Since the locking algorithm execution times across all the benchmarks is very small (matter of seconds), the execution times were not reported.

TABLE II
RESILIENCE OF *SeqL* FOR ITC'99 SEQUENTIAL BENCHMARK CIRCUITS. THE SCAN-LOCKING WAS DONE USING *IBLA* ALGORITHM.

Bench.	*#SFFs*	*#SCs*	*EFF* [29]		*SeqL*			
			Res.	Ov.	n	p	γ	Res.
`b01`	5	1	✖	9%	4	0.93	20%	✔
`b02`	4	1	✖	12%	3	0.88	20 %	✔
`b03`	30	1	✖	14%	5	0.97	20 %	✔
`b04`	66	1	✖	8%	4	0.93	10 %	✔
`b05`	34	1	✖	3%	3	0.88	5%	✔
`b06`	9	1	✖	14%	2	0.75	20 %	✔
`b07`	51	1	✖	9%	3	0.88	10 %	✔
`b08`	21	1	✖	10%	3	0.88	15%	✔
`b09`	28	1	✖	13%	2	0.75	15 %	✔
`b10`	17	1	✖	8%	3	0.88	10%	✔
`b11`	30	1	✖	4%	2	0.75	5%	✔
`b12`	121	2	✖	10%	2	0.75	10%	✔
`b13`	53	1	✖	12%	4	0.93	15%	✔
`b14`	247	3	✖	1.0%	8	0.99	1%	✔
`b15`	447	5	✖	0.7%	9	0.99	1%	✔
`b17`	1407	15	✖	0.3%	16	0.99	0.5%	✔

A. *Resilience of SeqL vs. EFF against SAT-Attacks*

Table II shows the results of applying the procedure shown in Figure 5 on ITC'99 sequential circuits. The columns *#SFFs*, *#SCs*, *Res.*, *Ov.* and *DT.* indicate number of scan flip-flops, number of scan-chains, overhead and decryption time respectively. The resilience rate of *EFF* was 0%, while that of *SeqL* was 100%, thus indicating the superiority of *SeqL* over *EFF*. An abort limit of $24 hours$ was used for key decryption. The key decryption time for `b18` and `b19` circuits was more than this abort limit, without any result, hence the results for these 2 benchmarks are not reported.

Table III shows the results of applying the procedure shown in Figure 5 on 4 different encryption schemes validated in [8], and compared against *EFF* [29]. This table shows that *SeqL* secured all sequential circuits against SAT-attack in 100% of the cases. As explained in Section II,

- K_c was successfully decrypted in all cases, while
- K_{fi} was incorrect, hence causing functional output corruption, thus achieving resilience.

Results on $IOLTS'14$ gave 0% resilience in *EFF* case and 100% resilience in *SeqL* case, across all benchmarks, hence not reported in Table III for shortage of space. On the other hand, results on $DTC'10/LUT$ were not reported because it is LUT-based.

TABLE III
RESILIENCE OF *SeqL* FOR PIPELINED COMBINATIONAL BENCHMARKS FOR 5% LOGIC LOCKING. THE SCAN-LOCKING IS DONE USING *IKPA* ALGORITHM.

Bench.	RND		DAC'12		ToC'13/xor		ToC'13/mux	
	EFF	*SeqL*	*EFF*	*SeqL*	*EFF*	*SeqL*	*EFF*	*SeqL*
`apex2`	✖	✔	✖	✔	✔	✔	✖	✔
`apex4`	✖	✔	✖	✔	✔	✔	✔	✔
`i4`	✔	✔	✖	✔	✔	✔	✖	✔
`i7`	✔	✔	✖	✔	✔	✔	✖	✔
`i8`	✔	✔	✖	✔	✔	✔	✖	✔
`i9`	✔	✔	✖	✔	✖	✔	✖	✔
`seq`	✔	✔	✖	✔	✔	✔	✖	✔
`ex1010`	✔	✔	✖	✔	✔	✔	✖	✔
`dalu`	✔	✔	✖	✔	✔	✔	✖	✔
`des`	✔	✔	✖	✔	✔	✔	✖	✔
`c432`	✖	✔	✖	✔	✖	✔	✖	✔
`c499`	✖	✔	✖	✔	✖	✔	✖	✔
`c880`	✔	✔	✔	✔	✖	✔	✖	✔
`c1355`	✔	✔	✖	✔	✔	✔	✖	✔
`c1908`	✔	✔	✖	✔	✖	✔	✖	✔
`c3540`	✔	✔	✖	✔	✖	✔	✖	✔
`c5315`	✔	✔	✔	✔	✖	✔	✖	✔
`c7552`	✔	✔	✖	✔	✖	✔	✖	✔

TABLE IV
RESILIENCE OF *SeqL* AGAINST STATE-OF-THE-ART ATTACKS ON PIPELINED COMBINATIONAL BENCHMARKS.

	Oracle-guided		Oracle-less	
Benchmark	ScanSAT [30]	SMT [27]	HackTest [20]	FALL [26]
`apex2`	✔	✔	✔	✔
`apex4`	✔	✔	✔	**No-key**
`i4`	✔	✔	✔	✔
`i7`	✔	✔	✔	✔
`i8`	✔	✔	✔	✔
`i9`	✔	✔	✔	✔
`seq`	$>$ **abort-limit**	✔	$>$ **abort-limit**	✔
`ex1010`	✔	✔	✔	**No-key**
`dalu`	✔	✔	✔	✔
`des`	✔	**No-key**	✔	✔
`c432`	✔	✔	✔	✔
`c499`	✔	✔	✔	✔
`c880`	✔	✔	✔	✔
`c1355`	✔	✔	✔	✔
`c1908`	✔	✔	✔	✔
`c3540`	✔	✔	✔	✔
`c5315`	✔	✔	✔	✔
`c7552`	✔	**No-key**	✔	✔

B. *Resilience of scan-unrolled versions of* SeqL-*locked design to state-of-the-art attacks on logic locking*

Table IV shows the resilience of *SeqL*-locked design to state-of-the-art attacks on logic locking like *ScanSAT* [30], *HackTest*-attack [20], functional-analysis-attacks on logic-locking (FALL)-attack [26], and SMT-attack [27]. The experiments were run on **Henry2** high-performance cluster from North Carolina State University [36], with an abort-limit of 24 hours. The resilience verification flow for oracle-guided attacks is similar to the flow in Figure 5.

For the oracle-less attacks the resilience verification flow is slightly different because of absence of the oracle, however *lcmp* verifier is still used for formal-equivalence-checking. Table IV shows that for FALL-attack [26] and SMT-attack [27], for some cases the solver returns `No-key`. This indicates the solver's inability to find the secret key for these particular input instances. We can empirically validate that *SeqL* has defended these cases as well, because of the attacker's inability to decrypt the correct key.

C. Resilience to removal-attack [37]

From Table II, we note that value of $n = |K_{fi}| = |K_{sq}|$ in most cases is less than 10. So, the total number of possible sequential key bits is $|K_{fi}| + |K_{sq}| < 20$, hence it is possible to find the functionally correct key using brute-force-search. Solution to address this issue is to lock extra FOs (after Algorithm 1 or 2 quits), which results in exponential increase in sequential key search space, with linear increase in area overhead.

Theorem 4: If circuit has n locked flip-flops, time complexity of removal attack [37] is $\mathcal{O}(2^n)$

Proof:

1) In the removal attack, the attacker needs to check if each key-gate removed introduces an inversion at the output of flip-flop. Thus, there are 2^n possibilities;
2) Assuming we use an adjacency list to represent the circuit as $G = (V, E)$, formal equivalence checking of the locked circuit with original circuit, has two steps: graph reduction in terms of decision diagrams and equivalence-checking;
3) The graph reduction step is $\mathcal{O}(G.log(G)) = \mathcal{O}((|V| + |E|).log(|V|+|E|))$. Since average-fanout ($|E|/|V|$) and ratio of gates to locked flip-flops ($|V|/n$) are both upper-bounded, graph reduction time reduces to $\mathcal{O}(n.log(n))$
4) It is well-known that combinational-equivalence-checking (CEC) is NP-complete [38] (although most practical instances run in polynomial time [39]). Assume CEC running-time is $\mathcal{O}(n^k)$, where k is a complexity constant;

Therefore, (even if assuming the NP-complete CEC step takes polynomial running time), the time it takes to launch removal attack is $\mathcal{O}((n.log(n)) + n^k)2^n) = \mathcal{O}(2^n)$, hence the proof.

D. Resilience to Multi-cycle Attacks

So far, we have discussed the attack in the context of a single-cycle test (one capture cycle). It is possible that the attacker uses the scan-chain to initialize the circuit, runs the circuit for more than one capture cycle (multi-cycle test), before observing the response through the scan-chain. In a multi-cycle scan test, there is only one scan-in cycle and one scan-out cycle per test vector, but multiple capture cycles (say N). This attack can be modelled by time-unrolling the netlist in Figure 1(c), as shown in Figure 6 ($N = 2$ in this particular case).

Due to *SeqL*'s functional isolation property, the functional-locks (FI key-gates) get unrolled N times (twice in this case) whereas the scan-locks (SQ key-gates) appear only once,

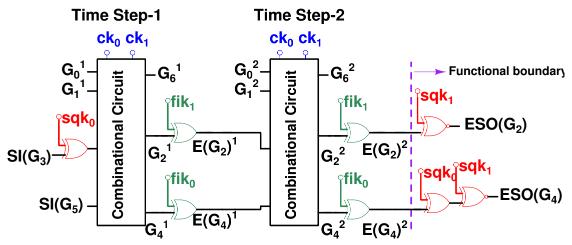


Fig. 6. Time-unrolled locked nestlist used to perform multi-cycle-attack (N=2) on *SeqL*

i.e., at the functional boundary. Because SQ key-gates are restricted to the boundary in the time-unrolled netlist as well, *SeqL*, which is resilient to single-cycle attack, is in principle also resilient to multi-cycle attacks.

E. Overheads

Table V shows the comparison of area, timing and energy for original, EFF-type and proposed $SeqL$-type locked scan flip-flops, obtained using SPICE transistor-level simulation. $NGSPICE$ open-source simulator, $Nangate$ $45nm$ library scan flip-flop and $45nm$ predictive technology model was used to arrive at these results. From Table V, it is evident that the proposed $SeqL$-style locked scan flip-flop has 22% and 19% reduction in $T_{CK-to-Q}$ and Energy-Per-Toggle (EPT) respectively, with only 4% area overhead as compared to EFF-style locked scan flip-flop.

TABLE V
FLIP-FLOPS COMPARISON: AREA, FUNCTIONAL TIMING AND ENERGY

FF	# Ts	T_{setup}	$T_{CK-to-Q}$	% Inc.	EPT	% Inc.
Orig.	38	$45ps$	$113ps$	-	$13.1fJ$	-
EFF	48	$45ps$	$163ps$	44%	$17.1fJ$	31%
$SeqL$	50	$45ps$	$127ps$	12%	$13.9fJ$	6%

V. DISCUSSION

A. Comparison to Timing-Driven SAT Defenses

In [40], *delay-locking* was proposed to mitigate SAT-attack. In *delay-locking*, the key not only determines the functionality of the circuit but also its timing profile. A functionally-correct but timing-incorrect key will result in timing violations and thus make the circuit malfunction. In [41], a novel SAT formulation based approach called *TimingSAT* was used to deobfuscate the functionalities of such delay locked designs within a reasonable amount of time. Since delay locking is orthogonal to *EFF* in defending SAT-attack, we do not compare with *delay-locking* or *TimingSAT*.

B. Limitations of SeqL

While *ScanSAT* [30] uses *EFF* as a representative example, there are other scan-locking variants. While *EFF* uses statically obfuscated scan-chains, a dynamically-obfuscated-scan (DOS) architecture is proposed in [33] and design-for-security (DFS) architecture is proposed in [34]. Recently, shift-and-leak [42] attack was proposed that is able to decrypt about 95% of the encryption key in the presence of scan-locking technique proposed in [34]. We do not cover those extensions in this work.

VI. Conclusions and Future work

We have proposed *SeqL*, that performs functional isolation and FI/SQ locking. *SeqL* hides a major fraction of the functionally correct keys, thus maximizing functional output corruption. We have shown both the theoretical and empirical improvements in the security of scan-locking. The results have shown 100% resilience to state-of-the-art oracle-guided as well as oracle-less attacks. Furthermore, since combinational key (excluding FIs) is completely recovered, it is sufficient to lock FI/SQ pairs, making *SeqL* efficient in terms of area, timing and power overheads.

References

[1] M. Pecht and S. Tiku, "Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum*, vol. 43, no. 5, pp. 37–46, 2006.

[2] S. Trimberger, "Trusted design in FPGAs," in *IEEE Design Automation Conference*, 2007, pp. 5–8.

[3] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *IEEE/ACM Design, Automation and Test in Europe*, 2008, pp. 1069–1074.

[4] The trade war with china and the problem with intellectual property rights. [Online]. Available: https://www.forbes.com/sites/davidvolodzko/2018/11/11/the-trade-war-with-china-and-the-problem-with-intellectual-property-rights/#9f2f079728e5

[5] China tariffs to hit the chip sector. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1333566#

[6] K. Huang, J. M. Carulli, and Y. Makris, "Counterfeit electronics: A rising threat in the semiconductor manufacturing industry," in *IEEE International Test Conference*, 2013, pp. 1–4.

[7] C. Gorman, "Counterfeit chips on the rise," *IEEE Spectrum*, vol. 49, no. 6, pp. 16–17, 2012.

[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*, 2015, pp. 137–143.

[9] T. Camo. [Online]. Available: https://www.mentor.com/products/sm/trustchain-security-platform

[10] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *IEEE/ACM Design, Automation and Test in Europe*, 2008, pp. 1069–1074.

[11] J. Rajendran, H. Zhang, C. Zhang, G. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 21, no. 5, pp. 410–424, 2015.

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

[13] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[14] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, Feb 2019.

[15] 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*, May 2016, pp. 236–241.

[16] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating SAT-unresolvable circuits," in *IEEE Great Lakes Symposium on VLSI 2017*, 2017, pp. 173–178.

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

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

[19] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," in *IEEE/ACM International Conference on Computer-Aided Design*, 2017, pp. 49–56.

[20] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2668–2682, Nov 2017.

[21] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott, and H. Zhou, "BeSAT: Behavioral sat-based attack on cyclic logic encryption," in *IEEE Asia and South Pacific Design Automation Conference*, 2019, pp. 657–662.

[22] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *IEEE Asian Hardware Oriented Security and Trust Symposium*, 2018, pp. 56–61.

[23] A. Rezaei, Y. Li, Y. Shen, S. Kong, and H. Zhou, "CycSAT-unresolvable cyclic logic encryption using unreachable states," in *IEEE Asia and South Pacific Design Automation Conference*, 2019, pp. 358–363.

[24] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "TTLock: Tenacious and traceless logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust*, May 2017, pp. 166–166.

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

[26] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *IEEE/ACM Design Automation and Test in Europe*, 2019, pp. 936–939.

[27] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," in *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019.

[28] M. E. Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," in *IEEE/ACM International Conference on Computer-Aided Design*, 2017, pp. 33–40.

[29] R. Karmakar, S. Chattopadhyay, and R. Kapur, "A scan obfuscation guided design-for-security approach for sequential circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2019.

[30] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking obfuscated scan chains," in *IEEE Asia South Pacific Design Automation Conference*, 2019, pp. 352–357.

[31] J. Chen, Y. Chen, W. Weng, C. Huang, and C. Wang, "Synthesis and verification of cyclic combinational circuits," in *IEEE International System-on-Chip Conference*, 2015, pp. 257–262.

[32] "TSMC integrated backend services." [Online]. Available: www.tsmc.com/english/dedicatedFoundry/services/integrated_backend.htm

[33] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure scan and test using obfuscation throughout supply chain," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, Sep. 2018.

[34] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 26, no. 5, pp. 818–830, 2018.

[35] "ITC'99 benchmarks." [Online]. Available: https://www.cerc.utexas.edu/itc99-benchmarks/bench.html

[36] "Henry2 Cluster: North Carolina State University High Performance Computing Center." [Online]. Available: https://projects.ncsu.edu/hpc//main.php

[37] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," in *IEEE Transactions on Emerging Trends in Computing*, 2017.

[38] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[39] S. Fortune, J. Hopcroft, and E. M. Schmidt, "The complexity of equivalence and containment for free single variable program schemes," in *Automata, Languages and Programming*, G. Ausiello and C. Böhm, Eds. Springer, 1978, pp. 227–240.

[40] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," in *IEEE/ACM Design Automation Conference*, 2017, pp. 1–6.

[41] A. Chakraborty, Y. Liu, and A. Srivastava, "TimingSAT: Timing profile embedded SAT attack," in *IEEE/ACM International Conference on Computer-Aided Design*, 2018, pp. 6:1–6:6.

[42] N. Limaye, A. Sengupta, M. Nabeel, and O. Sinanoglu, "Is robust design-for-security robust enough? attack on locked circuits with

restricted scan chain access," *CoRR*, vol. abs/1906.07806, 2019. [Online]. Available: http://arxiv.org/abs/1906.07806