SeqL: Secure Scan-Locking for IP Protection

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 logic-locking attacks are known to successfully decrypt *functionally correct key* of a locked combinational circuit. It is possible to extend these attacks to real-world Silicon-based Intellectual Properties (IPs, which are sequential circuits) through scan-chains by selectively initializing the combinational logic and analyzing the responses. In this paper, we propose *SeqL*, which achieves functional isolation and locks selective flip-flop 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 IP. 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 pipelined combinational benchmarks (`ISCAS`, `MCNC`), sequential benchmarks (`ITC`) and a fully-fledged `RISC-V CPU`, *SeqL* gave 100% resilience to a broad range of state-of-the-art attacks including SAT [1], Double-DIP [2], HackTest [3], SMT [4], FALL [5], Shift-and-Leak [6] and Multi-cycle attacks [7].**

***Index Terms*—IP Piracy, Logic Locking, Scan-chains**

I. Introduction

Logic-locking is a solution that was touted to address IP piracy threats in the semiconductor supply chain. This technique adds key-gates with one input driven by secret key, to obfuscate IP's inner details. The transformation is reversed only upon application of the programmed secret key, thus preserving the IP's original function. Unfortunately, logic-locking has been a cat-and-mouse game where existing locking proposals [8]–[14] fail to ever-advancing attacks [1]–[5]. Although these attacks primarily target combinational circuits, they can be extended to real-world sequential circuits through *scan-chains*. But the fundamental attack assumption is that inputs are controllable and outputs are observable. Thus, if the scan-chains are secured, it would be possible to provide a secure logic locking solution.

This paper proposes *SeqL*, a new logic locking technique that secures scan-chains. SeqL advances the prior work on design-for-security (DFS) [7], [15], [16], by conducting a formal study and empirically validating the security against a broad class of state-of-the-art attacks. Although attacks on large-scale sequential designs through functional execution is an open problem, attacks through the scan-chains currently exist. Thus, SeqL serves as the proper first line of defense. We demonstrate how to automate SeqL and quantify its low overheads on large-scale circuits. Therefore, SeqL addresses both the security and practicality challenges of logic locking.

Figure 1 outlines the system we consider. We highlight a simple setting for ease of explanation and elaborate later in our threat model. We assume that the primary inputs and primary outputs of the IP under consideration are not accessible, while only the scan-chains are accessible to the attacker, in embedded deterministic test (EDT)-bypass mode. The input register R_i applies primary inputs to the IP, and the output register R_o stores the primary outputs. The scan-chain connects all flip-flops in R_i, subsequently to the flip-flops internal to the IP and finally the flip-flops in R_o. The scan-input (SI) and scan-output (SO) ports are controllable and observable respectively by the attacker. Hence, the attacker can apply selective inputs to the IP and observe corresponding IP responses through these ports.

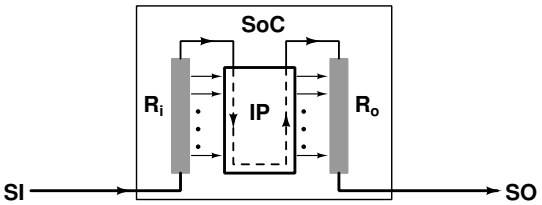


Fig. 1: Scan-based IP access for logic locking attacks.

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: (a) isolates functional path from the locked scan path; (b) locks FIs and causes functional output corruption;
3) *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 security of *SeqL* is also empirically evaluated and verified against a broad set of best known attacks;
5) The small overheads of *SeqL* and its ease of implementation makes it attractive for industry practice.

II. Prior Work

The first wave of logic locking techniques [8]–[10] have been shown to be vulnerable to Boolean Satisfiability (SAT) attack [1]. In SAT-attack, distinguishable input patterns (DIPs) are obtained from the locked circuit and incorrect keys are pruned-off based on oracle's responses to the DIPs. Several defenses were then proposed to mitigate SAT-attack, such as *Anti-SAT* [13], *SARLock* [12] and *Cyclic Obfuscation* [17], but they have failed to address the vulnerability to *AppSAT* [18], *Double-DIP* [2], *CycSAT* [19], *HackTest* [3], *BeSAT* [20] and machine-learning [21] attacks. While [22] proposes a new cyclic logic locking technique to defend *CycSAT* [19], *TTLock* [11] and Secure Function Logic Locking (SFLL) [14] were the only locking schemes that were broadly resilient to these attacks, yet they recently failed against functional analysis of logic locking (FALL) [5] and SMT [4] attacks.

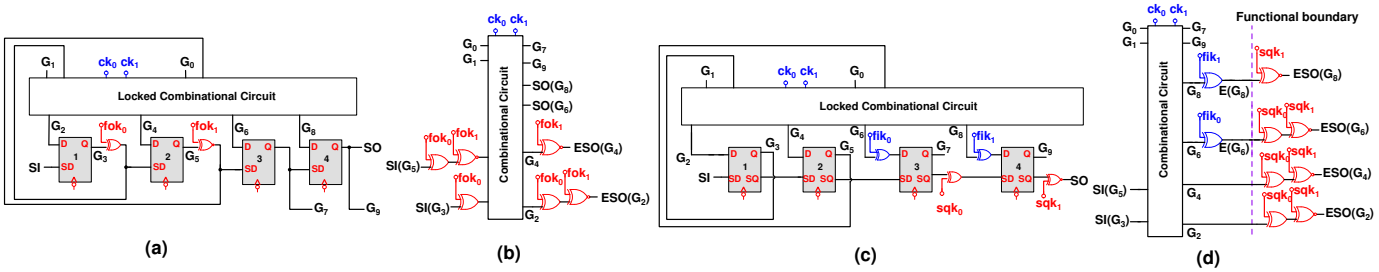


Fig. 2: (a) *EFF*-style: Sample sequential circuit with logic locking and FO locking; (b) Scan-unrolled equivalent of Fig. 2(a); (c) *SeqL*-style: Functional isolation and locked FIs/SQs; and (d) Scan-unrolled equivalent of Figure 2(c). *SeqL* considers flip-flops without feedback.

Additionally, to address the issue of defending against SAT-attack on sequential circuits, several DFS techniques have been proposed: (1) *FORTIS* [15], (2) Robust DFS (*RDFS*) [7] and (3) Encrypt Flip-Flop (*EFF*) [16]. *FORTIS* [15] is vulnerable to multi-cycle-test attacks [7] ; *RDFS* [7] addresses these issues but necessitates routing of a global *Test* signal to all the key-based scan flip-flops, adds significant overheads, vulnerable to shift-and-leak [6] attack and increases test generation effort. *EFF* [16] addresses these issues by locking scan flip-flip outputs (FOs). But *EFF* is insecure against *ScanSAT* [23], thus there is a need for a better defense that is both secure and practical.

III. Preliminaries

This section discusses the vulnerability of prior work on scan-locking and highlights the threat model.

A. Scan-Locking [16] & State-of-the-Art Attacks [4], [5], [23]

In *EFF* technique [16], flip-flops (FFs) on the non-critical timing paths of a sequential circuit are selected, and XOR/XNOR-type key gates are added to lock the Q-outputs, which drive combinational logic as well as the scan-chain. Figure 2(a) shows a sample sequential circuit with 2 out of 4 FFs encrypted using *EFF*-style scan-locking. In Figure 2(a):

- G_0 and G_1 are the primary inputs. SI and SO are the circuit's scan-input port and scan-output port respectively;
- FFs 1 and 2 have feedback, while FFs 3 and 4 do not have feedback. G_2, G_4, G_6 and G_8 are corresponding FF inputs (FIs) respectively. G_3, G_5, G_7 and G_9 are corresponding FF outputs (FOs) respectively; and
- ck_0 and ck_1 are the combinational key bits, while fok_0 and fok_1 are key bits used to lock the FO G_3 (XOR-type key gate) and FO G_5 (XNOR-type key gate) respectively.

ScanSAT [23] shows that it is possible to convert this scan-locked instance to the scan-unrolled locked instance of Figure 2(b), launch the SAT-attack on the unrolled instance and decrypt the functionally correct sequential key. Here, in scan-mode of operation: $SI(G_3)$ and $SI(G_5)$ are the scan-input-bits corresponding to FFs 1 and 2 respectively; and $ESO(G_2)$ and $ESO(G_4)$ are the locked-scan-output bits corresponding to FFs 1 and 2 respectively. Hence, *EFF* technique is not secure. Similar to *ScanSAT* [23], it is possible to extend some of the state-of-the-art attacks like *HackTest*-attack [3], functional-analysis-attacks on logic-locking (FALL) [5], and SMT-attack [4]. We thus evaluate *SeqL* on all these attacks.

B. Threat model

We consider a malicious foundry that offers fabrication, assembly and testing services [24]. 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 scenarios: (1) The attacker uses an activated IC (*oracle*), and applies scan patterns to the IP, and observes corresponding scan responses in EDT-bypass mode. Since the IP is located somewhere deep inside the *SoC*, we assume that the IP is **controllable/observable, only through scan-chains**. Typically, scan ports are not deactivated to facilitate debug of customer returns, which the attacker exploits to launch the SAT-attack; or (2) The attacker is at the outsourced tester, where the attacker can place the dies in EDT-bypass mode, applies desired scan patterns to the IP, and observes corresponding scan responses.

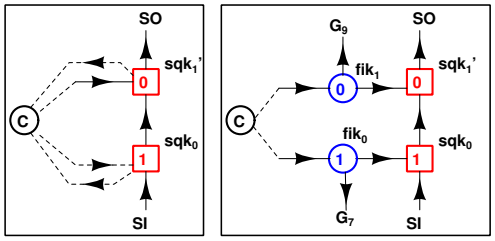
IV. Solution Insight

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

1) In *EFF*-style scan-locking, the FO key-gate corresponding to the FF appears *both* in the scan-input and scan-output paths of the FF in the scan-unrolled instance;
2) Functional path can be isolated from the locked scan path used by the attacker, thus achieving resilience;
3) To prevent vulnerability to multi-cycle-attack [7], it is important to selectively lock FFs without feedback;
4) Since the FO key gates cascade with FI key gates to form XOR/XNOR-chains, it is possible to obfuscate the solver.

Figure 2(c) shows the proposed *SeqL*-style scan-locking by transforming the circuit in Figure 2(a), using above principles. Figure 2(c) is different from Figure 2(a) in the following ways:

- There is a separate Q and SQ, and the key gate is added at SQ (SQ key-gate), thus leaving the functional output Q unencrypted. This is referred to as *functional isolation*;
- FFs without feedback e.g., 3 and 4 are selected for locking;
- sqk_0 and sqk_1 are the key bits used to lock the SQ output of FFs 3 (with XOR-type key gate) and 4 (with XNOR-type key gate) respectively; and
- Extra key gates (both of XOR type and without additional obfuscation logic in this case, for ease of explanation) are added to lock FIs of both these FFs, using fik_0 and fik_1 key bits respectively. These key gates are referred to as FI key gates in the rest of this paper.



(a) *EFF*-style (b) *SeqL*-style

Fig. 3: Abstract models for circuits in Figures 2(a) and 2(c)

Figure 2(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 the 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*. After running SAT-attack on circuit in Figure 2(d), when the combinational portions of the sequential circuit, the original unencrypted sequential circuit and the solver key are inputted to the formal equivalence checker, the result was *not equivalent*. Hence, by functional isolation and FI locking, resilience was achieved, thus making the proposed *SeqL*-style scan-locking mechanism secure. Next subsection explains this behavior using an abstract model.

A. Abstract model

Figure 3 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 blue circular vertex, and each FF key-gate abstracted into a red rectangular vertex. Figures 3(a) and 3(b) show models corresponding to circuits in Figures 2(a) and 2(c) respectively.

In the abstract model corresponding to the proposed scan-locking shown in Figure 3(b), the following are functional paths:

- $FP_0 : fik_0 \longrightarrow G_7$
- $FP_1 : fik_1 \longrightarrow G_9$

and the following are scan-out paths:

- $SP_0 : fik_0 \longrightarrow sqk_0 \longrightarrow sqk_1^{'} \longrightarrow SO$
- $SP_1 : fik_1 \longrightarrow sqk_1^{'} \longrightarrow SO$

Figure 3(b) shows 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, for functional paths, the number of inversions for FP_0 and FP_1, are 1 and 0. Since functional path FP_0 has odd-inversion-parity, the circuit is *incorrect for functional operation*.

To understand the behavior systematically, Table I enumerates all possibilities for the scan-lock $\{fik_1, sqk_1^{'}, fik_0, sqk_0\}$ for the circuit in Figure 2(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. Among the four functionally-correct rows only one is scan-correct, thus *SeqL* **hides** $\frac{3}{4}$

TABLE I. Truth table of our proposed scan-lock in Figure 2(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

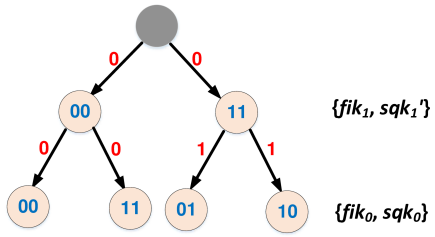


Fig. 4: Key Assignment Graph (KAG) for circuit in Figure 2(c). KAG is a binary tree with dummy root node, the leaves of which correspond to the rows in Table I whose scan-correctness column is TRUE.

functional-correct keys from the attacker. Similarly, among the four scan-correct rows, *only one* is functionally correct, thus *SeqL* maximizes the odds against the functionally-correct-key among the scan-correct-keys. Figure 4 shows the corresponding key assignment graph (KAG). The sequential key returned by the solver is the second leaf from the left. Since this leaf is a functionally incorrect key, the technique is able to achieve resilience. In this example, odds against the functionally correct key is $p = \frac{3}{4} = 0.75$.

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.

Given an FI-SQ key-pair $\{fik_i, sqk_i\}$, there are 4 possible assignments $\{00, 01, 10, 11\}$. Let n be the number of locked FI-SQ pairs. Let $KAG = (V, E)$ 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. 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. KAG is a tree, whose root vertex is a dummy node, with exactly two children 00 and 11.

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. 00 and 11 are even-

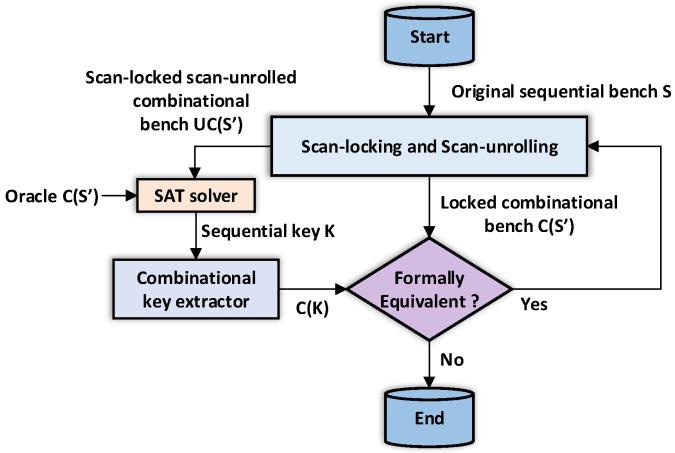


Fig. 5: Automating SeqL

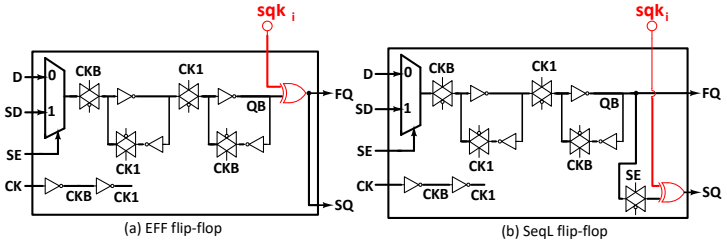


Fig. 6: Flip-flop variants for scan-locking

parity vertices, whereas 01 and 10 are odd-parity vertices. The children of 00 and 01 are even-parity vertices. The children of 10 and 11 are odd-parity vertices. Hence, every non-root vertex has exactly 2 children. 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. 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 both 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 has exactly two children, KAG is a binary tree, hence the proof.

Theorem 3: The odds against the functionally-correct-key among the scan-correct-keys is $p = 1 - \frac{1}{2^n}$
Proof: The path from root to a functionally correct leaf should have all 00 nodes, there is exactly one such leaf in KAG. Since, the total number of leaves in $KAG = 2^n$, odds $p = \frac{2^n-1}{2^n} = 1 - \frac{1}{2^n}$, hence the proof.

V. 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 it can be practically deployed on large circuits.

Objective: Lock selective scan flip-flops (FI-SQ pairs) without feedback 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 rapidly with n, with minimal increase in area overhead. Figure 5 shows *SeqL* flow, which exploits this principle to iteratively lock scan flip-flops (FI-SQ pairs) from the end of the scan-chain(s) until functional output is corrupted. Thus, the proposed scan-locking solution 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

We 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 6(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 6(b), which isolates the functional path from the locked scan path. Hence, the SQ key gate locks 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, similar to CSP-scan [25]. The comparison of area, timing and EPT of the *EFF* as well as SeqL flip-flops are provided later in Section VI.

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

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 output boundary. We measure the success of the *IKPA* algorithm using formal equivalence checking. If the

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 *for a flip-flop without feedback* 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}

Algorithm 2: Iterative boundary locking algorithm for sequential circuits

Input: S, K_{sq}

while $S^{'} = S$ *and* $|K_{fi}| + |K_{sq}| <= \gamma$ **do**
  Identify an unvisited FI-SQ pair *for a FF without feedback* and mark corresponding pair as visited ;
  Obfuscate corresponding FI-SQ pair;
  Run SAT-solver and update K_{fi}, $S^{'}$;
end
Result: $S^{'}$, K_{fi}

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 K_{fi}, that causes functional output corruption. Section VI demonstrates the detailed results.

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

Unlike pipelined logic-locked combinational circuits, there are no existing key gates for sequential circuits. Hence, FI-SQ locking or in other words, boundary locking has to be done afresh. Moreover, since K_c is always successfully decrypted, combinational logic except FIs is not locked to ensure cost-effectiveness. Since the higher the number of inserted key gates at the FI-SQ boundary, the higher the area overhead, we make this parameter, γ, user-configurable. Algorithm 2 takes a sequential benchmark as input, and iteratively obfuscates unvisited FI-SQ pairs with XOR/XNOR-type key-gates, 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.

VI. 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*. Algorithms 2 (*IBLA*) and 1 (*IKPA*) were used for scan-locking the sequential benchmarks and pipelined combinational benchmarks, respectively. Both the locking algorithms were implemented in *Perl*. Since the locking algorithm execution times across all the benchmarks were very small (matter of seconds), the execution times were not reported.

TABLE II. Resilience of *SeqL* for Pipelined Combinational Benchmarks for 5% logic locking. *'✔' is secure and '✖' is insecure.*

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	✔	✔	✖	✔	✔	✔	✖	✔
k2	✔	✔	✖	✔	✔	✔	✖	✔
ex1010	✔	✔	✖	✔	✔	✔	✖	✔
dalu	✔	✔	✖	✔	✔	✔	✖	✔
des	✔	✔	✖	✔	✔	✔	✖	✔
c432	✖	✔	✖	✔	✖	✔	✖	✔
c499	✖	✔	✖	✔	✖	✔	✖	✔
c880	✔	✔	✔	✔	✖	✔	✖	✔
c1355	✔	✔	✖	✔	✔	✔	✖	✔
c1908	✔	✔	✖	✔	✖	✔	✖	✔
c3540	✔	✔	✖	✔	✖	✔	✖	✔
c5315	✔	✔	✔	✔	✖	✔	✖	✔
c7552	✔	✔	✖	✔	✖	✔	✖	✔

TABLE III. Resilience of *SeqL* against state-of-the-art attacks on pipelined combinational benchmarks.

	Oracle-guided			Oracle-less	
Bench.	DDIP [2]	SS [23]	SMT [4]	HT [3]	FALL [5]
apex2	✔	✔	✔	✔	✔
apex4	✔	✔	✔	✔	NK
i4	✔	✔	✔	✔	✔
i7	✔	✔	✔	✔	✔
i8	✔	✔	✔	✔	✔
i9	✔	✔	✔	✔	✔
seq	✔	–	✔	–	✔
k2	–	✔	–	✔	✔
ex1010	✔	✔	✔	✔	NK
dalu	✔	✔	✔	✔	✔
des	–	✔	NK	✔	✔
c432	✔	✔	✔	✔	✔
c499	✔	✔	✔	✔	✔
c880	✔	✔	✔	✔	✔
c1355	✔	✔	✔	✔	✔
c1908	✔	✔	✔	✔	✔
c3540	✔	✔	✔	✔	✔
c5315	–	✔	✔	✔	✔
c7552	NK	✔	NK	✔	✔

All experiments are run on *IBM BladeCenter® Cluster* with abort-limit of 1 week. *'✔' is secure and '✖' is insecure. '-' indicates decryption time exceeds abort-limit, while 'NK' indicates* `No-Key`.

A. Resilience of SeqL vs. EFF [16] against SAT-Attacks on pipelined combinational benchmarks

Table II shows the results of applying the procedure shown earlier in Figure 5 on 4 different encryption schemes validated in [1], and compared against *EFF* [16]. This table shows that *SeqL* secured all sequential circuits against SAT-attack in 100% of the cases. As explained in Section IV, (1) K_c was successfully decrypted in all cases, while (2) K_{fi} was incorrect, hence causing functional output corruption, thus achieving resilience. Results on $IOLTS'14$ encryption scheme [1], [10] gave 0% resilience in *EFF* case and 100% resilience in *SeqL* case, across all benchmarks, hence not reported in Table II for brevity.

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

Table III confirms the resilience of *SeqL*-locked design to state-of-the-art attacks on logic locking like *Double-DIP* (DDIP) [2], *ScanSAT* (SS) [23], *HackTest* (HT)-attack [3], functional-analysis-attacks on logic-locking (FALL) [5], and SMT-attack [4]. All experiments were run on IBM BladeCenter® Cluster, with an abort-limit of *1 week*. Those entries in the table which are empty, correspond to all those

TABLE IV. Resilience of *SeqL* for Sequential Circuits. The Scan-locking was done using *IBLA* algorithm.

Bench.	*#Gates*	*#SFFs*	*#SCs*	$\|R_{wof}\|$	*EFF* [16]		*SeqL*								
										Resilience			Decryption Time		
					Res.	Ov.	n	p	Ov.	N=1	N=2	N=5	N=1	N=2	N=5
b14	10,012	245	3	54	✖	3.3%	8	0.99	0.24 %	✔	✔	✔	19 min	2 min	2 min
b15	12,992	449	5	70	✖	4.3 %	9	0.99	0.2 %	✔	✔	✔	47 min	11 min	164 min
b17	32,192	1,415	15	97	✖	5.2 %	6	0.99	0.05 %	✔	✔	✔	10 min	17 $hrs.$	47 $hrs.$
b18	114561	3,320	34	23	✖	3.8 %	10	0.99	0.03 %	✔	–	–	53 $hrs.$	**> abort-limit**	**> abort-limit**
b19	231,266	6,642	67	30	✖	3.7 %	10	0.99	0.01 %	✔	–	–	91 $hrs.$	**> abort-limit**	**> abort-limit**
b20	20,172	490	5	22	✖	3.3 %	10	0.99	0.15 %	✔	✔	✔	7 $min.$	15 $min.$	37 $min.$
b21	20,517	490	5	22	✖	3.2 %	10	0.99	0.15 %	✔	✔	✔	6 $min.$	34 $min.$	36 $min.$
b22	29,897	735	8	22	✖	3.3 %	10	0.99	0.1 %	✔	✔	✔	11 $min.$	37 $min.$	67 min
RISC-V flat.	25,096	2,031	20	226	✖	7.9 %	10	0.99	0.09 %	✔	✔	✔	2 $min.$	13 $min.$	6 $hrs.$

[1] N denotes the number of capture cycles in the multi-cycle scan-based test. The scan flip-flops without feedback, R_{wof}, are stitched by designer as a separate scan-chain for security considerations. *IBM BladeCenter® Cluster* with abort limit of 1 week is used. '✔' *is secure and* '✖' *is insecure.*

cases which have crossed this abort-limit while performing key decryption. Similarly, for some cases the solver returns `No-key` (indicated as NK in the table). 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.

C. Resilience of SeqL vs. EFF against SAT-Attacks on sequential benchmarks

Table IV shows the results of applying the procedure shown in Figure 5 on ITC'99 open-source sequential gate-level benchmarks and flattened `RISC-V CPU` netlist. The `RISC-V CPU` RTL is obtained from [26], and gate-level synthesis is performed using *Nangate 45nm* library using *Synopsys Design Compiler®*. Scan chains and EDT-compression are inserted into the gate-level netlist using *Mentor Graphics TestKompress®* (decompressor and compactor will not be used because the attack is launched in EDT-bypass mode).

The scan-inserted gate-level-verilog is converted to the bench format used in the attack tools, using an in-house *Python* script. The attack tools only support basic gates like *AND/OR/NAND/NOR/XOR/XNOR/NOT/BUF/MUX*, however the *RISC-V* gate-level-verilog contains more complex gates like *AOI* (and-or-invert), *OAI* (or-and-invert), *HA* (half-adder) and *FA* (full-adder). Our *Python* script internally converts each of these complex gates into a composition of basic gates, before final conversion to *bench*, which is acceptable because *IBLA* algorithm inserts scan-locks and does not affect combinational logic. The compression hardware is not converted because the attacks are meant to be launched in EDT-bypass mode. The columns *#SFFs*, *#SCs*, *Res.* and *Ov.* indicate number of scan flip-flops, number of scan-chains, resilience and overhead 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 *1 week* was used for key decryption.

D. Resilience to Multi-cycle attacks [7]

So far, we discussed attacks using a single capture cycle. The attacker can also run the circuit for $N > 1$ capture cycles (multi-cycle test), without affecting the shift cycles. This attack can be modeled by time-unrolling the reverse-engineered netlist as well as the oracle N times. Since scan-in and scan-out phases span hundreds of clock cycles and N is in general relatively very small, running at slow-speed or at-speed will not significantly affect test-time/attack-time. Table IV shows results for this attack. Similar to single-cycle attack ($N = 1$), *SeqL* was resilient to multi-cycle attack ($N = 2, 5$) across all benchmarks.

TABLE V. Area, Timing and Energy Overhead Comparison

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%

For EFF, since key is successfully recovered for $N = 1$ itself, resilience results for $N > 1$ were not shown.

E. Resilience to Shift-and-Leak attack (SaLa) [6]

RDFS [7] inserts special secure cells (SCs) into scan-chains to drive the key-gates. Unlike *RDFS*, *SeqL* key-gates are directly driven by the tamper-proof memory, without SCs in between. The first goal of *SaLa* is to find leaky cells, and shift the content of SCs into leaky cells. Due to absence of SCs in *SeqL*, this first goal is never achieved. The second goal of *SaLa* is to find the *leak condition* and satisfy it. Since *SeqL* locks the scan-chain itself, it is mandatory to know the scan-key upfront to invoke test generator and find the *leak condition*. Since the goal is itself key-decryption, it is not possible to find the *leak condition*, let alone satisfy it. Thus, *SeqL* is inherently resilient to *SaLa*.

F. Overheads

Table V shows the comparison of area, timing and energy for original, EFF-style and proposed $SeqL$-style 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 $SeqL$ 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 flip-flop.

VII. Conclusions

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* cost-efficient. Moreover, we have demonstrated *SeqL* on large designs such as `RISC-V CPU`, demonstrating its applicability in mainstream industry practice.

VIII. Acknowledgements

This work is supported in part by the German Research Foundation (DFG), Cluster of Excellence "Center for Advancing Electronics Dresden", Technische Universitat Dresden.

REFERENCES

[1] P. Subramanyan et al, "Evaluating the security of logic encryption algorithms," in *IEEE HOST*, 2015, pp. 137–143.

[2] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *IEEE GLSVLSI*, 2017, pp. 179–184.

[3] M. Yasin et al, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *IEEE TIFS*, vol. 12, no. 11, pp. 2668–2682, 2017.

[4] K. Z. Azar et al, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," in *CHES*, 2019.

[5] D. Sirone et al, "Functional analysis attacks on logic locking," in *IEEE/ACM DATE*, 2019, pp. 936–939.

[6] N. Limaye et al, "Is robust design-for-security robust enough? attack on locked circuits with restricted scan chain access," in *IEEE ICCAD*, 2019.

[7] U. Guin et al, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *IEEE TVLSI*, vol. 26, no. 5, pp. 818–830, 2018.

[8] J. Roy et al, "EPIC: Ending Piracy of Integrated Circuits," in *IEEE/ACM DATE*, 2008, pp. 1069–1074.

[9] J. Rajendran et al, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 21, no. 5, pp. 410–424, 2015.

[10] S. Dupuis et al, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *IEEE IOLTS*, 2014, pp. 49–54.

[11] M. Yasin et al, "TTLock: Tenacious and traceless logic locking," in *IEEE HOST*, May 2017, pp. 166–166.

[12] M. Yasin et al, "SARLock: SAT attack resistant logic locking," in *IEEE HOST*, May 2016, pp. 236–241.

[13] Y. Xie et al, "Mitigating SAT attack on logic locking," in *CHES*, 2016, pp. 127–146.

[14] M. Yasin et al, "Provably-secure logic locking: From theory to practice," in *ACM CCS*, 2017, pp. 1601–1618.

[15] U. Guin et al, "FORTIS: A comprehensive solution for establishing forward trust for protecting IPs and ICs," *ACM TODAES*, vol. 21, no. 4, pp. 63:1–63:20, 2016.

[16] R. Karmakar et al, "A scan obfuscation guided design-for-security approach for sequential circuits," *IEEE TCAS II: Express Briefs*, pp. 1–1, 2019.

[17] K. Shamsi et al, "Cyclic obfuscation for creating SAT-unresolvable circuits," in *IEEE GLSVLSI*, 2017, pp. 173–178.

[18] K. Shamsi et al, "AppSAT: Approximately deobfuscating integrated circuits," in *IEEE HOST*, 2017, pp. 95–100.

[19] H. Zhou et al, "CycSAT: SAT-based attack on cyclic logic encryptions," in *IEEE/ACM ICCAD*, 2017, pp. 49–56.

[20] Y. Shen et al, "BeSAT: Behavioral SAT-based attack on cyclic logic encryption," in *IEEE ASP-DAC*, 2019, pp. 657–662.

[21] P. Chakraborty et al, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *IEEE AsiaHOST*, 2018, pp. 56–61.

[22] A. Rezaei et al, "CycSAT-unresolvable cyclic logic encryption using unreachable states," in *IEEE ASP-DAC*, 2019, pp. 358–363.

[23] L. Alrahis et al, "ScanSAT: Unlocking obfuscated scan chains," in *IEEE ASP-DAC*, 2019, pp. 352–357.

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

[25] S. Potluri, A. S. Trinadh, S. B. Ch., V. Kamakoti, and N. Chandrachoodan, "Dft assisted techniques for peak launch-to-capture power reduction during launch-on-shift at-speed testing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 1, pp. 14:1–14:25, 2015.

[26] A. Magyar. V-scale, an implementation of an RV32IM core in Verilog. [Online]. Available: https://riscv.org/2015/09/risc-v-in-verilog/