SeqL+: Secure Scan-Obfuscation with Theoretical and Empirical Validation

Seetal Potluri, *Member, IEEE*, Shamik Kundu, *Student Member, IEEE*, Akash Kumar, *Senior Member, IEEE*, Kanad Basu, *Senior Member, IEEE*, and Aydin Aysu, *Senior Member, IEEE*.

***Abstract*—Existing logic-locking attacks are known to successfully decrypt a *functionally correct key* of a locked combinational circuit. Extensions of these attacks to real-world Intellectual Properties (IPs, which are sequential circuits) have been demonstrated through the scan-chain by selectively initializing the combinational logic and analyzing the responses. In this paper, we propose *SeqL+* to mitigate a broad class of such attacks. The key idea is to lock selective functional-input/scan-output pairs of flip-flops without feedback to cause attackers to decrypt an incorrect key, and to scramble flip-flops with feedback to increase key-length without introducing further vulnerabilities.**

We conduct a formal study of the *scan-locking* and *scan-scrambling* problems and demonstrate automating our proposed defense on any given IP. This study reveals the first formulation and complexity analysis of Boolean Satisfiability (SAT)-based attack on scan-scrambling. We formulate the attack as a conjunctive normal form (CNF) using a worst-case $\mathcal{O}(n^3)$ reduction in terms of scramble-graph size n, making SAT-based attack applicable and show that scramble equivalence classes are equi-sized and of cardinality 1. In order to defeat SAT-attack, we propose an iterative swapping-based scan-cell scrambling algorithm that has $\mathcal{O}(n)$ implementation time-complexity and $\mathcal{O}(2^{\lfloor \frac{\alpha.n+1}{3} \rfloor})$ SAT-decryption time-complexity in terms of a user-configurable cost constraint α $(0 < \alpha \leq 1)$.

We empirically validate 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], Multi-cycle [7], Scan-flushing [8], and Removal [9] attacks.

***Index Terms*—IP Piracy, Logic-locking, Scan-chains, Scan-scrambling.**

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 a 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 [10]–[16] 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.

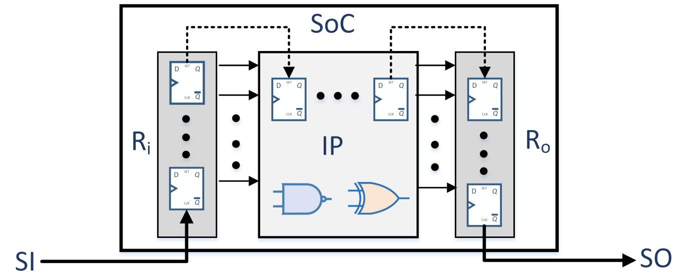

Fig. 1. Scan-based IP access for logic-locking attacks.

This paper proposes *SeqL+*, a new logic-locking technique that secures scan-chains. *SeqL+* advances the state-of-the-art on design-for-security (DfS) [7], [17], [18], by conducting a formal study of the problem 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 are an open problem, attacks through the scan-chains currently exist. Thus, *SeqL+* serves as the proper first line of defense.

Figure 1 outlines the system we consider. The primary inputs (PIs) and primary outputs (POs) of the IP under consideration are not accessible, while only the scan-chains are accessible to the attacker. Input-register R_i thus can apply the primary inputs to the IP and the output-register R_o can store the primary outputs. The scan-chain connects all scan flip-flops (SFFs) in R_i, subsequently to the SFFs internal to the IP, and finally to the SFFs in R_o. SI and SO show the scan-input and scan-output ports, respectively, from the external world. In this case, the attacker can apply selective inputs to the IP using the SI input-port and observe corresponding IP responses through the SO output-port. For ease of explanation, we have shown a single scan-chain in this figure, which contains only R_i, R_o and SFFs internal to the IP under consideration. However, in practice, there can be multiple chains and compression logic, and the chain may contain SFFs from other IPs and glue-logic.

A. Contributions

Our previous work, *SeqL*, is available at the proceedings of the 21st International Symposium on Quality Electronic Design (ISQED 2020) Conference [19]. The main contributions of *SeqL* were as follows:

S. Potluri and A. Aysu are with the Electrical and Computer Engineering Department, North Carolina State University, Raleigh, NC, 27606.

S. Kundu and K. Basu are with the Department of Electrical and Computer Engineering, University of Texas at Dallas, Richardson, TX, 75080.

A. Kumar is with the Department of Computer Science, Technical University of Dresden, 01062 Dresden, Germany

1) We identified there is 100% correlation between SFF input (FI) locking and functional output corruption. Exploiting this property, we proposed *SeqL*, that: (a) isolates functional path from the locked scan path; (b) locks FIs for SFFs *without* feedback and causes functional output corruption;
2) *SeqL* hides the majority of the scan-correct keys which are functionally correct, thus maximizing the probability of the decrypted key being *functionally incorrect*; and
3) The security of *SeqL* is empirically evaluated and verified against common attacks. The small area, timing, and energy-per-toggle overheads of *SeqL* and its ease of implementation make it attractive for industry practice.

This paper proposes *SeqL+*, which enhances *SeqL* with the following extensions:

1) We verify resilience of *SeqL* to the *scan-flushing attack* by flushing the content of the scan chains to reveal some information about the key bits used to obfuscate the scan path;
2) For the special case of circuits without an adequate number of SFFs without feedback (R_{wof}), we propose *scan-scrambling* to use SFFs with feedback, to improve security;
3) We discuss the limitations of existing works on scan-scrambling in the context of logic-locking and propose the first practical adaptation of scan-scrambling to the logic-locking problem, and also validate that *scan-scrambling* doesn't introduce a vulnerability against other attacks which are successfully defended by *SeqL*;
4) We show the first formulation of Boolean Satisfiability (SAT)-based attack on scan-scrambling; and
5) We provide both formal analysis and empirical evaluation to show the security against the SAT-based attack and quantify the overheads of *scrambling*.

II. Prior Work

The first wave of logic-locking techniques [10]–[12] have been shown to be vulnerable to SAT-based attack [1]. In SAT-based attack, distinguishable input patterns (DIPs) are obtained from the locked circuit and incorrect keys are pruned-off based on the oracle's responses to the DIPs. Several defenses were then proposed to mitigate SAT-based attack, such as *Anti-SAT* [20], *SARLock* [14] and *Cyclic Obfuscation* [21], but they have failed to address the vulnerability to *AppSAT* [22], *Double-DIP* [2], *CycSAT* [23], *HackTest* [3], *BeSAT* [24], *TGA* [25] and SAIL (machine-learning) [26] attacks. While *unreachable state encryption* [27] proposes a new cyclic logic-locking technique to defend *CycSAT* [23], *TTLock* [13] and Stripped-Functionality Logic-Locking (SFLL) [16] were the only schemes that were broadly resilient to attacks, yet they recently failed against functional analysis of logic-locking (FALL) [5] and SMT [4] attacks.

Additionally, to address the issue of defending against SAT-based attack on sequential circuits, several DFS techniques have been proposed: (1) *FORTIS* [17], (2) Robust DFS (*RDFS*) [7] and (3) Encrypt Flip-Flop (*EFF*) [18]. *FORTIS* [17] 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 SFFs, adds significant overheads, vulnerable to shift-and-leak [6] attack and increases test generation effort. *EFF* [18] addresses these issues by locking SFF outputs (FOs). But *EFF* is insecure against *ScanSAT* [8], thus there is a need for a better defense that is both secure and practical.

Additionally, existing work on scan-scrambling [28] assumes multiple inputs to the scramble-multiplexer coming from different scan-chains. Since the adversary knows that the correct input to the scramble-multiplexer comes from the same scan-chain (which is unique), one can easily know which input is the correct one and hence easily infer the secret-scrambling-key, making that kind of scan-scrambling is impractical/insecure. Thus, there is a need for a better scrambling 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. Threat model

We follow the standard assumption in logic-locking framework that assumes a malicious foundry offering fabrication, assembly and testing services [3], [8], [17], [28]–[31]. Such an adversary has access to gate-level netlist as well as scan-chain access on an activated integrated circuit (IC). There is a growing concern in such attacks given the move to offshore fabs: even Intel is planning to outsource its 5nm designs to TSMC [32]. Scan-chains provide finer access, making attacks without scan-access a subset of attacks with scan-access. Our threat model considers full access to scan-ports. Since we can protect circuits with access to scan-chains, by definition, it shall protect circuits without access to scan-chains [33]. Further, we do not compare against dynamically obfuscated scan [34], because the context is not logic-locking.

There are two possible scenarios: (1) The attacker uses an activated IC (*oracle*), applies scan patterns to the IP and observes corresponding scan responses in the embedded deterministic test (EDT)-bypass mode. Since the IP is located somewhere deep inside the *System-on-Chip (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-based attack; or (2) The attacker is at the outsourced tester [30], where the attacker can place the dies in the EDT-bypass mode, apply desired scan patterns to the IP, and observe corresponding scan responses.

B. Scan-Locking & State-of-the-Art Attacks

In *EFF* technique [18], SFFs 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 SFFs in the scan-chain. Figure 2(a) shows a sample sequential circuit with 2 out of 4 SFFs encrypted using *EFF*-style scan-locking. In Figure 2(a):

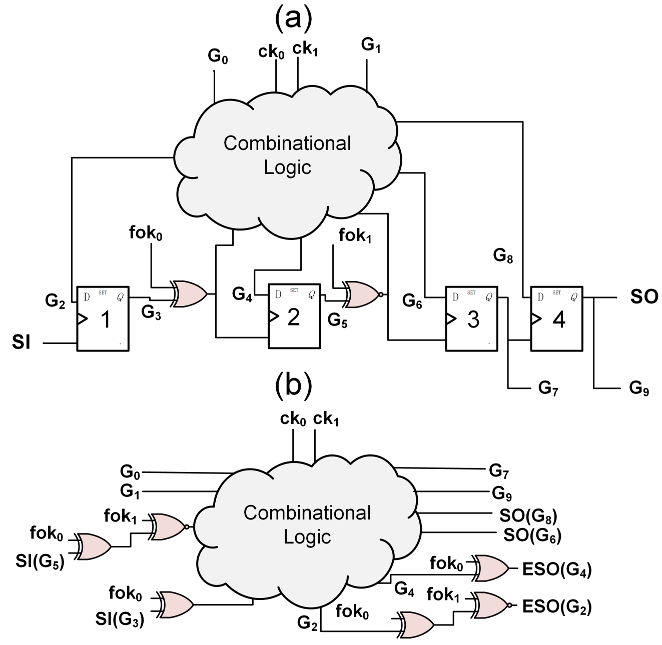

Fig. 2. (a) EFF-style locking (b) Scan-unrolled equivalent

- G_0 and G_1 are the PIs. SI and SO are the circuit's scan-input port and scan-output port, respectively;
- SFFs 1 and 2 have feedback, while SFFs 3 and 4 do not have feedback. G_2, G_4, G_6 and G_8 are corresponding SFF inputs (FIs) respectively. G_3, G_5, G_7 and G_9 are corresponding SFF 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 [8] shows that it is possible to convert this scan-locked instance to the scan-unrolled locked instance of Figure 2(b), launch the SAT-based attack on this 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 SFFs 1 and 2, respectively; and $ESO(G_2)$ and $ESO(G_4)$ are the locked-scan-output bits corresponding to SFFs 1 and 2 respectively. Hence, *EFF* technique is not secure. Similar to *ScanSAT* [8], 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.

IV. Solution Insight

As discussed in the previous section, when SAT-based attack is launched on the scan-unrolled *EFF*-style scan-locked circuit shown in Figure 2(b), the SAT solver returns the functionally correct key. Figure 3(a) shows the proposed *SeqL*-style scan-locking idea by transforming the circuit in Figure 2(a), in the following 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*;
- SFFs without feedback i.e., 3 and 4 are selected for locking;

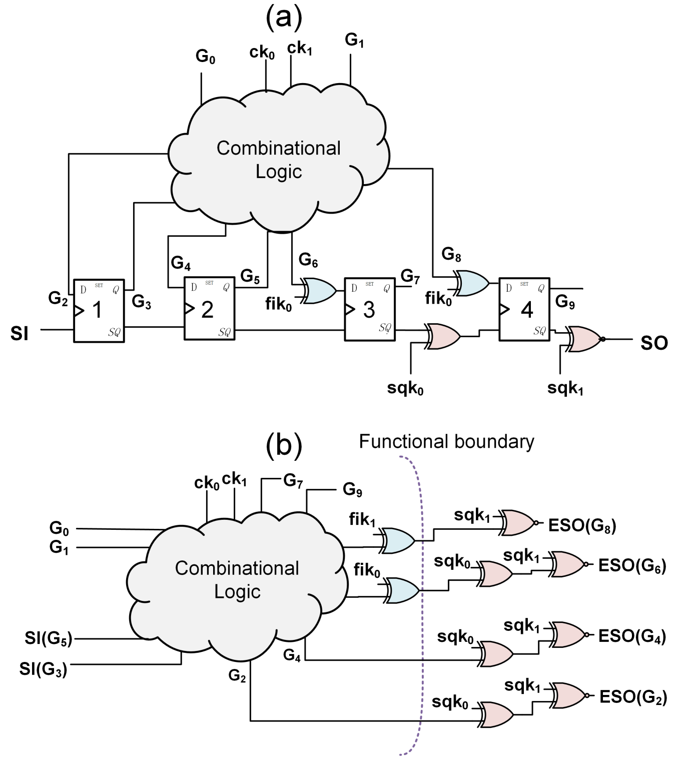

Fig. 3. (a) Proposed SeqL-style locking (b) Scan-unrolled equivalent

- sqk_0 is the key bit used to lock the SQ output of SFF 3, using an XOR-type key gate;
- sqk_1 is the key bit used to lock the SQ output of SFF 4, using an XNOR-type key gate; and
- Extra key gates (both of XOR type and without additional obfuscation logic in this case, for ease of explanation) are added at FIs of both these SFFs. fik_0 and fik_1 are the FI locking key bits respectively. These key gates are referred to as FI key gates in the rest of this paper.

Figure 3(b) 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 the normal functional operation of the circuit.* This is because the attacker uses scan mode of operation, and hence observes $ESO(G_2)$ and $ESO(G_4)$. Although conjunctive normal form (CNF) extraction and subsequent SAT-based attack are possible [34], the key returned by the SAT-solver is guaranteed to ensure correct operation only in the scan-mode and not in the functional-mode of operation, due to the functional-isolation property. In the absence of scan-locking, it is well-known that the scan-chain(s) do not impact the functionality of the IPs/SoC. However, the *functional isolation* property of *SeqL* implies that if the adversary tries to decrypt the key in the scan-mode, the decrypted key does in-fact impact the functional operation.

The reason behind this behavior is that 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*. Hence, by functional isolation and FI locking, functional output corruption was achieved, thus mak-

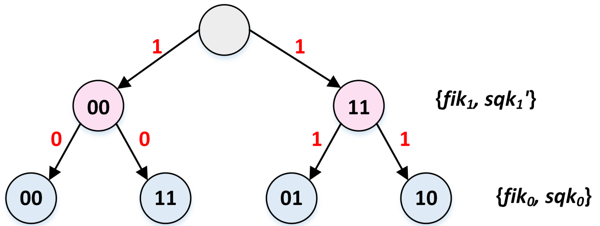

Fig. 4. Key Assignment Graph (KAG) for circuit in Figure 3(a). KAG is a binary tree, whose the leaves correspond to scan-correct keys.

ing the proposed *SeqL*-style scan-locking mechanism secure.

Figure 4 shows the corresponding key assignment graph (KAG). When SAT-based attack is launched on this scan-unrolled instance, the complete sequential key returned by the solver is $K = \{sqk_0, sqk_1, ck_0, ck_1, fik_0, fik_1\} = \{1, 1, \mathit{1,0,1,0}\}$, where the key bits in italics indicate those that lock the combinational portion of the sequential circuit. The corresponding portion of the key that locks only FIs and SQs is $K_s = \{sqk_0, sqk_1, fik_0, fik_1\} = \{1, 1, \mathit{1,0}\ \}$. This corresponds to the second leaf from the left in the KAG shown in Figure 4. Since this is a functionally incorrect key, the technique is able to achieve functional output corruption. In this example, the odd against the functionally correct key is $p = \frac{3}{4} = 0.75$. The next section provides a more rigorous mathematical analysis explaining this behavior.

A. Analysis

This section formally analyzes the security of logic-locking and proves that if *SeqL* is used to lock n SFFs without feedback, then the odd 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-labeled 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} SFF from the end of the scan-out-path. All node and edge assignments 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 $\{FI, SQ\}$ key-gate combination is {XOR, XOR}, {XOR, XNOR}, {XNOR, XOR} or {XNOR, XNOR} respectively. 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, whereas 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 SFF, which is same as the parity of the corresponding child vertex. inv_k equals 0 or 1, depending on whether k^{th} SFF along the scan-chain from *SO* is locked with an XOR or $XNOR$ key-gate respectively.

Theorem IV.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,

$$(fik_i \oplus sqk_i \oplus inv_i) \oplus \bigoplus_{k=1}^{i-1} (sqk_k \oplus inv_k)$$

should equal 0. If

$$\bigoplus_{k=1}^{i-1} (sqk_k \oplus inv_k)$$

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 are 0. On the other hand, if

$$\bigoplus_{k=1}^{i-1} (sqk_k \oplus inv_k)$$

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. QED

Lemma IV.2. *KAG is a binary tree.*

Proof. The 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. QED

Theorem IV.3. *The odd against the functionally-correct-key among the scan-correct-keys is $p = 1 - \frac{1}{2^n}$*

Proof. The path from the root to a functionally correct leaf should have all 00 nodes. Applying theorem IV.1 recursively, we can show there is exactly one such leaf in KAG. Subsequently, from lemma IV.2 we know KAG is a binary tree, hence the total number of leaves in $KAG = 2^n$. This makes the odds against the functionally-correct-key $p = \frac{2^n - 1}{2^n} = 1 - \frac{1}{2^n}$, hence the proof. QED

The details of automating *SeqL* have been discussed in detail in the conference version and skipped here for brevity. The objective is to iteratively lock selective SFFs (FI-SQ pairs) without feedback such that functional output corruption is achieved, while area-overhead is minimized.

B. Limitation for designs with small R_{wof}

SeqL uses SFFs *without* feedback to improve security. For circuits which have few SFFs without feedback ($R_{wof} < 50$) but with many SFFs with feedback, it will be useful to have some other technique that can utilize the ones with feedback, so as to increase the key-length sufficient enough to defend *brute-force attack*. Since *scan-scrambling* is able to achieve this, we use *SeqL* to utilize SFFs without feedback, and *scan-scrambling* to utilize SFFs with feedback respectively, to improve security. The next section discusses in detail, our methodology to deploy *scrambling* for designs with small R_{wof}, to address this limitation.

V. SeqL+: Scrambling to enhance security of designs with small R_{wof}

In [35], Hely et al., identify for the first time the trade-off between testability and security, and hence the vulnerabilities in cryptographic implementations that use industry-standard *scan methodology*. The same paper also proposes *scan-scrambling* as a countermeasure, wherein they perform scan-chain segmentation and insert scramble-multiplexers (MUXes) at inputs of all the segments. The scramble-MUXes have one of the inputs which is correct (from the correct input scan-segment) and remaining inputs are incorrect (from the incorrect input scan-segments), and the select lines form the *secret-scrambling-key*.

Since the *secret key* is known only to the designers, the attacker is unable to extract the secrets from the cryptographic implementations, without knowing the *secret-scrambling-key*. We augment and apply the scan-scrambling technique to improve the security of circuits, which lack adequate SFFs without feedback, against brute-force attacks. We conduct a formal analysis of SAT resilience of scan-chain scrambling along with an empirical validation.

A. Benefit of scan-scrambling in the context of logic-locking attacks

The key-benefit of *scan-scrambling* is that to launch the SAT-based attack [1] on sequential circuits, the adversary needs to know the ordering of SFFs in the scan-chain in order to initialize the SFFs to known-values, and observe the corresponding next-state responses. Since *scan-scrambling* locks the ordering of SFFs in the scan-chain, the attacker is unable to achieve this, thus preventing direct applicability of SAT-based attack. This reasoning is also applicable to ScanSAT [8], where the attacker scan-unrolls the XOR/XNOR-chains. Since scan-unrolling of XOR/XNOR-chains is not possible without knowing the ordering of SFFs, *scan-scrambling* also naturally defends ScanSAT [8].

B. Limitations of existing work on scan-scrambling in the context of logic-locking attacks

In ScanSAT [28], the authors consider scan-scrambling in the context of logic-locking, where they assume that the various inputs to the scramble-MUX come from different scan-chains. Since the attacker knows that the correct input to the scramble-MUX comes from the same scan-chain (which is unique), that kind of scan-scrambling is impractical/insecure. Hence, in *SeqL+* we consider that all the inputs to the scramble-MUX come from the same scan-chain and proceed with the security analysis.

The main ideas of *scan-scrambling* [35] are: (i) partitioning of a scan-chain into multiple segments; and (ii) addition of a $2:1$ MUX at the input of each segment, for the purpose of scrambling. The select signal to the scramble-MUX corresponds to a secret key-bit. The vector of all the select signals to these MUXes constitutes the *secret-scrambling-key*. Choice of larger MUX exists [35] but may not be cost-effective in practice. Since fine-grained scramble-MUX insertion results in exponential increase in security-level with linear increase in area-overhead, we scramble only one scan-chain (security scan-chain) and insert a scramble-MUX for each SFF in this security scan-chain.

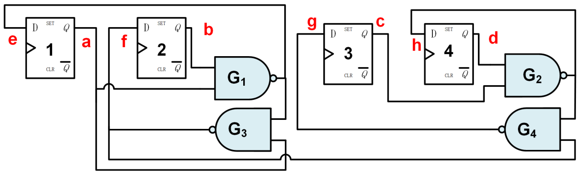

Fig. 5. Sample circuit consisting of four gates and four flip-flops.

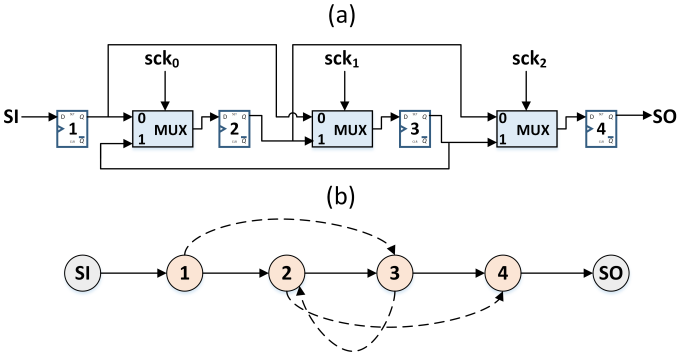

Fig. 6. (a) Sample scrambled scan-chain corresponding to Figure 5 and (b) Corresponding *scramble-digraph*

C. Formal Security Analysis of Scan-Scrambling Against SAT-attack

The key research question is: **"Would scan-scrambling form equivalence classes (ECs) like conventional, combinational logic-locking, causing a vulnerability against SAT-based attacks?"**. This subsection conducts a formal complexity analysis and formulation of scan-scrambling against such attacks, and proves crucial properties of ECs.

Graph-based Formulation: Every scan-scrambled instance can be formulated as a digraph $G = (V, E)$ where:

1) Scan-input (SI), all $SFFs$ and scan-output (SO) are represented as vertices (V) in G; and
2) The connections between SI, $SFFs$ and SO in the circuit are represented as directed edges (E) between corresponding vertices in G, where the direction signifies the signal flow.

Definition V.1. *A Hamiltonian path in a digraph is a path that visits each vertex exactly once.*

Figure 5 shows a sample circuit with four 2-input `nand` gates and four flip-flops (prior to scan-insertion). Figure 6(a) shows an example of scrambled scan-chain corresponding to this circuit, and Figure 6(b) shows the corresponding *scramble-digraph*. Thus, every pair of permuted SFFs demands addition of 3 extra-edges if the vertices are adjacent in the original scan-chain, otherwise it demands 4 extra-edges. There are two possible *Hamiltonian Paths* (HPs) in Figure 6(b), $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ corresponding to $\{sck_0, sck_1, sck_2\} = (011)_2$ and $1 \rightarrow 3 \rightarrow 2 \rightarrow 4$ corresponding to $\{sck_0, sck_1, sck_2\} = (100)_2$ (equivalent to the SFF-orderings in the scrambled scan-chain in Figure 6(a)).

The scramble-key-combinations corresponding to the *HPs* in G ensure that all the $SFFs$ are connected together along

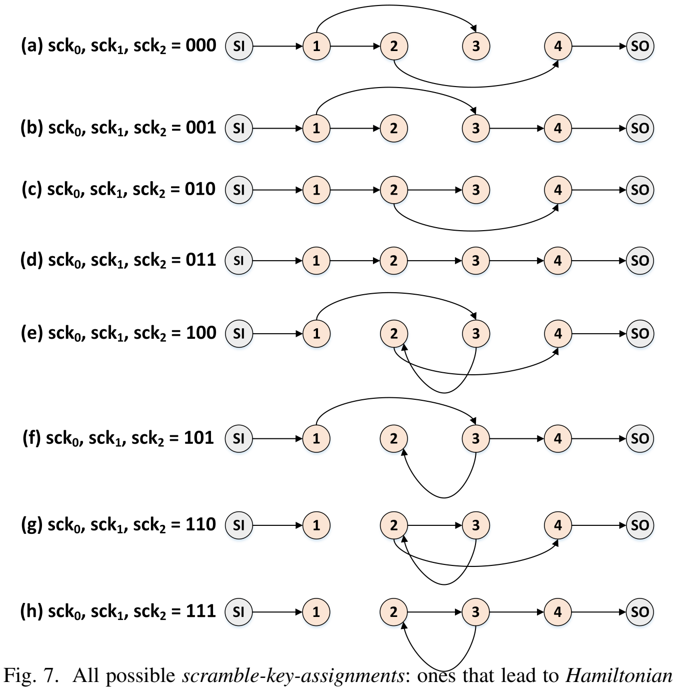
(a) sck0, sck1, sck2 = 000
(b) sck0, sck1, sck2 = 001
(c) sck0, sck1, sck2 = 010
(d) sck0, sck1, sck2 = 011
(e) sck0, sck1, sck2 = 100
(f) sck0, sck1, sck2 = 101
(g) sck0, sck1, sck2 = 110
(h) sck0, sck1, sck2 = 111
SI
1
2
3
4
SO

Fig. 7. All possible *scramble-key-assignments*: ones that lead to *Hamiltonian Paths* (HPs) i.e., $\{sck_0, sck_1, sck_2\} = \{0, 1, 1\}$ and $\{sck_0, sck_1, sck_2\} = \{1, 0, 0\}$ are valid.

with SI and SO to form the scan-chain. The remaining scramble-key-combinations disassociate some of the $SFFs$ from the scan-chain. Hence, we focus only on *HPs* in G and corresponding scramble-key-combinations. It is clear that each *valid* scramble-key-combination corresponds to exactly one *HP* in G (because application of scramble-key configures the connections, thereby creating a unique path). Next, we provide a formal proof of the converse, i.e. each *HP* in the *scramble-digraph* corresponds to a unique scramble-key-combination.

Lemma V.1. *All* HPs *in* G *correspond to* valid *scrambles.*

Figure 7 shows the resultant connections between SFFs $1, 2, 3$ and 4, for all possible *scramble-key-assignments* to sck_0, sck_1, sck_2. As we can see, only key-assignments that lead to *HPs* i.e., Figures 7(d) and 7(e) correspond to *valid* scrambles and remaining key-assignments disassociate one or more SFFs.

Theorem V.2. *Every* HP *in* G *has* injective mapping *to exactly one* valid *scramble-key-combination.*

Proof. Let H_s be a selected *HP* in G. Now, H_s corresponds to a particular ordering of vertices in G, say $\{v_1(H_s), v_2(H_s) \ldots v_N(H_s)\}$. Since each vertex in G has *injective mapping* to a unique SFF in the circuit, H_s corresponds to a unique ordering of SFFs, say $\{SFF_1(H_s), SFF_2(H_s) \ldots SFF_N(H_s)\}$.

Let $SFF_i(H_s)$ be the i^{th} scan flip-flop and let $k_i(H_s)$ be scramble-key-bit corresponding to the scramble-MUX at the input of scan flip-flop $SFF_i(H_s)$:

- **Basis step**: $SFF_1(H_s)$ is the first scan flip-flop in the scan-chain, which means SI drives $SFF_1(H_s)$. In order to achieve this, there must be a unique assignment to $k_1(H_s)$. Hence, scramble-key-bit uniqueness is true for *i=1*.

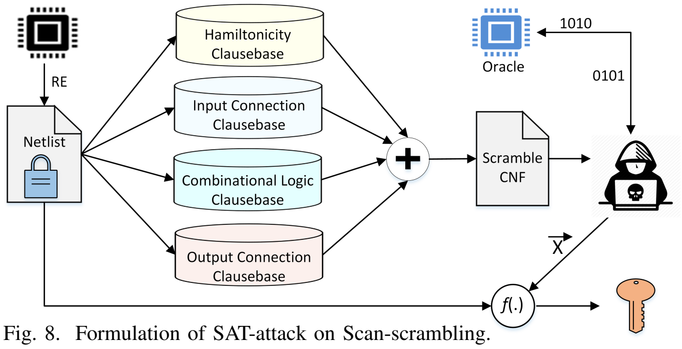
RE
Netlist
Hamiltonicity Clausebase
Input Connection Clausebase
Combinational Logic Clausebase
Output Connection Clausebase
Scramble CNF
Oracle
1010
0101
X
f(.)

Fig. 8. Formulation of SAT-attack on Scan-scrambling.

- **Induction step**: Assume scramble-key-bit uniqueness is true for *i=l*. $SFF_{l+1}(H_s)$ is the $(l+1)^{th}$ scan flip-flop, which means output of $SFF_l(H_s)$ should drive input of $SFF_{l+1}(H_s)$. In order to achieve this, there is a unique assignment to $k_{l+1}(H_s)$. Thus, scramble-key-bit uniqueness is true for *i=l+1*.

Hence, by *finite induction* we can infer that all scrambling-key-bits are unique for *HP* H_s. This indicates each *HP* in G corresponds to exactly one *scramble-key-combination*, and since the converse is also true, the mapping is *injective*, thus the proof. QED

Corollary V.2.1. *Theorem V.2 implies every* valid scrambling-key *is unique, or in other words, scramble ECs are equi-sized and of cardinality* 1.

The next subsection explains the first formulation of SAT-based attack on scan-scrambling. Subsequently, we exploit the property proven above in Corollary V.2.1 to exponentially increase the number of scramble ECs, so as to defeat SAT-based attack on scan-scrambling in subsection V-E.

D. Attacking scan-scrambling using SAT formulation

Figure 8 shows the SAT-formulation of the *HP* search corresponding to the correct scramble. The formulation comprises four different types of constraints:

1) Hamiltonian Path (HP) Constraints: From *Cook-Levin Theorem* [36], we know that the NP-complete *SAT* problem is polynomial-time reducible to the *HP* problem and vice-versa. Exploiting this principle, we show for the first time that scan-scrambling can be reduced to CNF, hence making it possible to launch SAT-based attack. So far we have seen how to break scrambling using HP search, next we shall see how to break using SAT-based attack.

Formulation: Given a scramble digraph G, we construct a Boolean CNF $B(G)$ such that such that $B(G)$ is satisfiable iff G has a HP. $B(G)$ has n^2 Boolean variables $\{x_{ij}\}$, $1 \leq i, j \leq n$. A satisfying truth assignment to $B(G)$ does provide us with a HP for G. Here, x_{ij} means the i^{th} position in the HP is occupied by node-j. An HP can be expressed as a permutation π of $\{1, 2, \ldots n\}$, where:

- $\pi(i) = j \Rightarrow i^{th}$ position is occupied by node-j.
- $(\pi(i), \pi(i+1)) \in G$ for $i = 1, 2, \ldots (n-1)$

Considering the example motivated thus far, $n = 4$, hence $B(G)$ has $4^2 = 16$ variables $\{x_{ij}\}$, $1 \leq i, j \leq 4$. The

Hamiltonicity Clausebase shown in Figure 8 is produced using HP constraints, which are multiple-fold:

1) Each node j must appear in the path, $1 \leq j \leq n = 4$
   - $x_{1j} \vee x_{2j} \vee x_{3j} \vee x_{4j}$

   Thus, total # constraints in this category is n.
2) No node j appears twice in the path, $1 \leq j \leq n = 4$
   - $\neg x_{1j} \vee \neg x_{2j}$, $\neg x_{1j} \vee \neg x_{3j}$, $\neg x_{1j} \vee \neg x_{4j}$
   - $\neg x_{2j} \vee \neg x_{3j}$, $\neg x_{2j} \vee \neg x_{4j}$, $\neg x_{3j} \vee \neg x_{4j}$

   Thus, total # constraints in this category is $\binom{n}{2} \times n$.
3) Every position i on the path must be occupied, $1 \leq i \leq n = 4$
   - $x_{i1} \vee x_{i2} \vee x_{i3} \vee x_{i4}$

   Thus, total # constraints in this category is n.
4) No two nodes j and k occupy the same position i in the path, $1 \leq i, j, k \leq n = 4, j \neq k$
   - $\neg x_{i1} \vee \neg x_{i2}$, $\neg x_{i1} \vee \neg x_{i3}$, $\neg x_{i1} \vee \neg x_{i4}$
   - $\neg x_{i2} \vee \neg x_{i3}$, $\neg x_{i2} \vee \neg x_{i4}$, $\neg x_{i3} \vee \neg x_{i4}$

   Thus, total # constraints in this category is $\binom{n}{2} \times n$.
5) Non-adjacent nodes i and j cannot be adjacent in the path, $1 \leq i, j \leq n = 4$
   - $\neg x_{1i} \vee \neg x_{2j}$, $\neg x_{2i} \vee \neg x_{3j}$, $\neg x_{3i} \vee \neg x_{4j}$

   Since the number of non-adjacent node-pairs depends on the scrambling algorithm, the total # constraints in this category depend on the scrambling algorithm as well.

Let's denote the set of clauses in this category as C_{HP}.

2) Constraints connecting SI bits to the SFF outputs: The *Input Clausebase* shown in Figure 8 is produced using input connection constraints. Considering the example motivated thus far, since $n = 4$, let I_1, I_2, I_3, I_4 be the input bits applied serially through SI and a, b, c, d be the outputs of SFFs $1, 2, 3, 4$ (or in other words, the inputs to the combinational circuit) respectively. The constraints connecting SI bits to the SFF outputs can be formulated as follows:

- $a = x_{11}.I_1 + x_{12}.I_2 + x_{13}.I_3 + x_{14}.I_4$
- $b = x_{21}.I_1 + x_{22}.I_2 + x_{23}.I_3 + x_{24}.I_4$
- $c = x_{31}.I_1 + x_{32}.I_2 + x_{33}.I_3 + x_{34}.I_4$
- $d = x_{41}.I_1 + x_{42}.I_2 + x_{43}.I_3 + x_{44}.I_4$

The above constraints have Boolean + and . operators on the right-hand side, which need to be converted to clauses [37], before adding to the CNF. Detailed equations are skipped for brevity and without loss of generality. Let's denote the set of clauses in this category as C_I. Each constraint corresponds to one SFF and there are n SFFs. Further, each constraint is a function of n 2-input and gates and $(n - 1)$ 2-input or gates. Since a 2-input and gate as well as a 2-input or gate translates to 3 clauses each in the CNF, there are altogether $3 \times (n + (n - 1)) = 3 \times (2n - 1)$ clauses, or in other words, $|C_I| = 3 \times (2n - 1)$.

3) Combinational Circuit Constraints: The *Combinational Logic Clausebase* shown in Figure 8 is produced by converting combinational logic gates to clauses, similar to the original SAT-based attack [1]. Figure 5 shows four 2-input nand gates G_1, G_2, G_3 and G_4 in the combinational portion of the scan-scrambled circuit. Converting these gates to clauses [37] gives:

- $G_1 \rightarrow (a + e)$, $(b + e)$, $(\overline{a} + \overline{b} + \overline{e})$
- $G_2 \rightarrow (c + h)$, $(d + h)$, $(\overline{c} + \overline{d} + \overline{h})$
- $G_3 \rightarrow (a + f)$, $(e + f)$, $(\overline{a} + \overline{e} + \overline{f})$
- $G_4 \rightarrow (f + g)$, $(h + g)$, $(\overline{f} + \overline{h} + \overline{g})$

Let's denote the set of clauses in this category as C_{Combo}.

4) Constraints connecting SFF inputs to SO bits: The *Output Connection Clausebase* shown in Figure 8 is produced using output connection constraints. Considering the example motivated thus far, since $n = 4$, let O_1, O_2, O_3, O_4 be the output bits serially scanned out through SO, and e, f, g, h be the inputs of SFFs $1, 2, 3, 4$ (or in other words, the outputs of the combinational circuit) respectively. The constraints connecting SFF inputs to SO bits can be formulated as follows:

- $O_1 = x_{11}.e + x_{12}.f + x_{13}.g + x_{14}.h$
- $O_2 = x_{21}.e + x_{22}.f + x_{23}.g + x_{24}.h$
- $O_3 = x_{31}.e + x_{32}.f + x_{33}.g + x_{34}.h$
- $O_4 = x_{41}.e + x_{42}.f + x_{43}.g + x_{44}.h$

Similar to the input constraints, the above output constraints need to be converted to clauses [37], before adding to the CNF. Let's denote the set of clauses in this category as C_O. Each constraint corresponds to one SFF and there are n SFFs. Further, each constraint is a function of n 2-input and gates and $(n - 1)$ 2-input or gates. Since a 2-input and gate as well as a 2-input or gate translates to 3 clauses each in the CNF, there are altogether $3 \times (n + (n - 1)) = 3 \times (2n - 1)$ clauses or in other words $|C_O| = 3 \times (2n - 1)$.

5) Running SAT-based attack on Scan-Scrambling: The clauses corresponding to these HP constraints, connection constraints, and combinational circuit constraints together produce the original scramble CNF $B(G)$ (shown earlier in Figure 8) needed to attack scan-scrambling.

$$B(G) = C_{HP} \cup C_I \cup C_O \cup C_{Combo} \tag{1}$$

The adversary uses this original scramble CNF $B(G)$, sends serial scan inputs to the oracle, observes the serial scan responses and produces the final CNF $B^{'}(G)$. The adversary runs the SAT-solver on $B^{'}(G)$ to solve for $\vec{X}$ shown in Figure 8.

Considering the example motivated thus far, the above formulation when provided as input to the SAT-solver, returns $x_{11} = x_{23} = x_{32} = x_{44} = 1$ and $x_{ij} = 0$ otherwise. This corresponds to $\pi(1) = 1$, $\pi(2) = 3$, $\pi(3) = 2$, $\pi(4) = 4$, or in other words the HP ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4$). We have seen how to calculate $\vec{X}$ using the above formulation. This $\vec{X}$ vector space can be mapped to the scrambling key space $\vec{K}$ and hence *decrypt* the key by looking at the reverse-engineered netlist as shown in Figure 8. Although so far, we have explained the attack on using a sample circuit consisting of four gates and four SFFs, the concept is generic and hence can be extended to any arbitrary scan-scrambled sequential circuit. The next section discusses how to defend SAT-based attack on scrambling.

E. Defending SAT-based attack on Scan-Scrambling

It is well-known that SAT-based attack is a brute-force search on the ECs [1]. The goal of our defense is to increase the number of scramble ECs, so as to make the attack computationally infeasible. Based on Corollary V.2.1, this translates to increasing the number of *HPs* in G. Thus, we

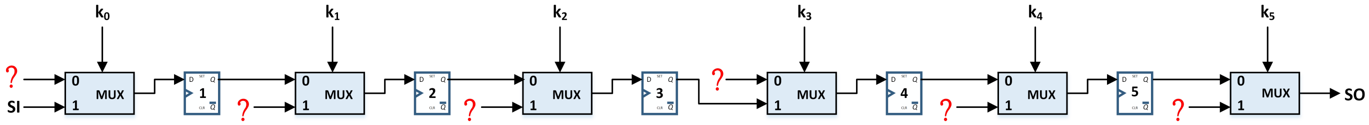

Fig. 9. One of the inputs of each scramble-MUX is known. The second input is unknown and has to be decided in such a way, so as to maximize the number of *HPs* in the resultant scramble-graph. The correct EC is $\{(k_0 = 1, k_1 = 0, k_2 = 0, k_3 = 1, k_4 = 0, k_5 = 0)\}$, and the objective is decide "?" connections, such that the number of incorrect ECs is maximized.

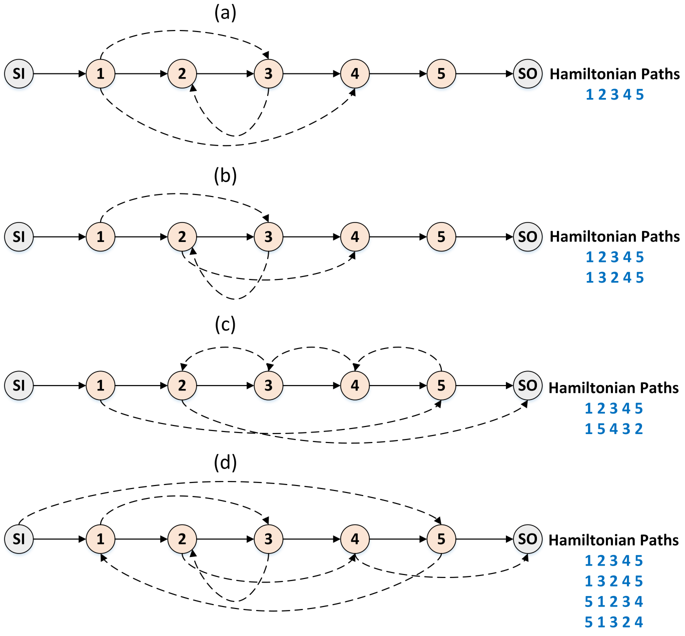

Fig. 10. Scramble Graphs

arrive at the following:

Objective: Connect the second input to the scramble-MUXes so as to produce a *scramble-digraph* G with maximum number of *HPs*.

1) *Search Space Exploration:* We assume only *security scan-chain* is scrambled, whose length is n. We assume a scramble-MUX at the input of each SFF as well as the scan-out port, thus there is a total of $(n+1)$ scramble-MUXes, as shown in Figure 9. Since the first input to each scramble-MUX is fixed corresponding to the correct scramble, and the second input available for exploration, the designer needs to evaluate the search space and decide the best choice. Avoiding self-loops and repetition, the second input of each scramble-MUX can be connected in $(n-1)$ ways. Thus, size of the scrambling search space is $(n-1)^{(n+1)}$.

Figure 10 shows the *scramble digraphs* exploring different possibilities of scan-scrambling the circuit in Figure 9 and the possible *HPs* in each case. Figure 10(a) has only 1 *HP* (least secure), Figures 10(b) and (c) have 2 *HPs* each and Figure 10(d) has 4 *HPs* (most secure). Thus, the way we connect the second input of the scramble-MUXes determines the # *HPs*, or in other words, the number of scramble ECs, hence the security level of the scrambled circuit.

2) *Disturbance analysis:* Prior to adding the second input to all the scramble-MUXes, there are n edges in the initial version of the scramble-graph, let's call this initial version as G_0. After adding the second input to all the scramble-MUXes, we know it is denoted as G. Let's define $G_1 = G - G_0$, which is essentially the collection of the newly added edges. Since the scan-chain in the final design has to be connected, G_0 has only 1 path and by definition Hamiltonian. Coming to G_1, since there are only n edges, it contains no more than 1 *HP*. This translates to at most 2 *HPs* which contain edges either purely from G_0 or purely from G_1. Thus, by definition, the additional *HPs* in G (at least $\kappa - 2$) will contain edges from both G_0 and G_1.

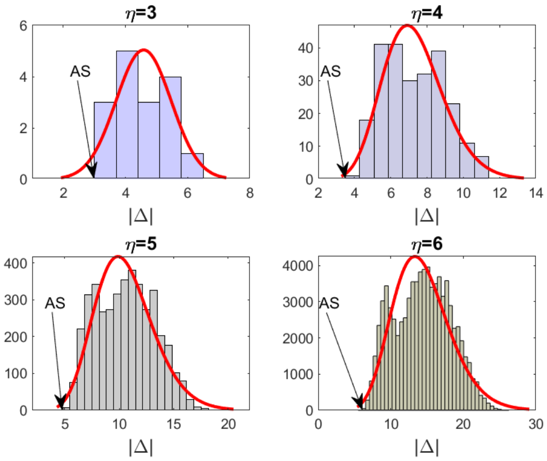

Fig. 11. $|\Delta|$ distribution with normal fit for $\eta = 3, 4, 5, 6$.

Definition V.2. *Let $\delta_i = |c_i^1 - c_i^2|$ be the defined as the disturbance produced on vertex-i, where c_i^1 and c_i^2 be the indices of the vertices whose outputs are connected to the first and second inputs the corresponding scramble-MUX.*

Since the majority of *HPs* ($\geq (\kappa - 2)$) reuse edges from both G_0 and G_1, the more the disturbance produced by δ_i on vertex-i, the more the number of edges from G_1 that provide the recovery to simultaneously satisfy δ_i as well complete the *HP*. On the contrary, the lesser the disturbance δ_i, the more localized its effect, and the fewer the number of edges from G_1 that provide the recovery to simultaneously satisfy δ_i as well complete the *HP*. For ease of explanation, let's define the disturbance vector $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$, resulting in

$$|\Delta| = \sqrt{\sum_i \delta_i^2}$$

Figure 11 shows the distribution of $|\Delta|$ for $\eta = 3, 4, 5, 6$ when running a *brute-force search*. We have observed that in all the cases, the lowest value of $|\Delta|$ is 6, 10, 11 and 12 for $\eta = 3, 4, 5, 6$ respectively. We have verified that this corresponds to the adjacent-scrambling (AS) case as shown

```
Algorithm 1: Iterative Swapping-based Scrambling
Input: $C$, $\eta$
Create a scramble-digraph $G = (V, E)$ with SFFs as
    vertices, and directed edges corresponding to signal
    flow in $C$;
$C^{'} = C, n_s \rightarrow 0$;
$G^{'} = G$;
while $n_s \leq \eta$ do
    Mark $\{v_{n_s}, v_{n_s+1}, v_{n_s+2}\}$ as visited ;
    Scramble SFFs $\{v_{n_s}, v_{n_s+1}\}$ and add directed
        edges to $G^{'}$ corresponding to the additional signal
        flow in $C^{'}$;
    $n_s \rightarrow n_s + 3$;
Result: $C^{'}$
```

in Figure 11 and also observed a general reduction in # *HP* with increase in $|\Delta|$. We have observed similar pattern for higher values of η, thus demonstrating the power of adjacent-scrambling. Since it is not possible to perform *brute-force search* for higher values of η, we exploit this observation and propose adjacent-scrambling algorithm explained in the next section and its security guarantees will be provided later in Section VI.

F. Adjacent-scrambling algorithm

Algorithm 1 shows the proposed iterative swapping-based scan-scrambling algorithm (or adjacent-scrambling in short), where C is the circuit and η is the user-defined cost/area constraint ($0 < \eta \leq n$). As discussed earlier, due to cost constraints we restrict ourselves to $2:1$ scramble-MUXes. Thus, every pair of permuted SFFs demands addition of 3 extra-edges if the vertices are adjacent in the corresponding *scramble digraph* of the original scan-chain while it demands 4 extra-edges if otherwise. From *pareto-optimality* perspective, we chose *adjacent-scrambling*. Thus, SFFs are allowed to be permuted only once, and it is also not allowed to permute their fanout SFFs as well, once permuted. Algorithm 1 thus picks one adjacent SFF pair at a time, and performs *iterative swapping*. This eliminates 3 SFFs from the exploration-space for future iterations, depending on whether the chosen SFF pair is adjacent or otherwise, respectively. Since it is a single loop iterating over the SFFs until the cost constraint η is met, the algorithm time-complexity is $\mathcal{O}(\eta)$.

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+*. In all the experiments, we have used the open-source *.bench* designs for sequential benchmarks (ITC'99 [38]), logic-locked combinational benchmarks (ISCAS'85, MCNC [1]) and a synthesized `RISC-V CPU`, along with *sld* solver and *lcmp* formal-equivalence checker provided by [1]. Since the locking algorithm execution times across all the benchmarks were in the order of few seconds, they were not reported. Further details on scan-locking experimental evaluation are available in the conference version [19].

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

	Oracle-guided			Oracle-less	
Bench.	DDIP [2]	SS [8]	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**	✔	✔

A. Resilience of SeqL vs. EFF [18] against SAT-Attacks on pipelined combinational benchmarks

Table I shows the results of applying *SeqL* on 4 different encryption schemes validated in [1], and compared against *EFF* [18]. This table shows that *SeqL* secured all sequential circuits against SAT-based attack in 100% of the cases. 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 our experiments, across all benchmarks, (1) K_c was successfully decrypted, while (2) K_{fi} was incorrect, hence causing functional output corruption, thus achieving resilience. Results on $IOLTS'14$ encryption scheme [1], [12] gave 0% resilience in the *EFF* case and 100% resilience in the *SeqL* case, across all benchmarks, hence not reported in Table I for shortage of space.

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

Table II shows the resilience of *SeqL*-locked design to state-of-the-art attacks on logic-locking like *Double-DIP* (DDIP) [2], *ScanSAT* (SS) [8], *HackTest* (HT)-attack [3],

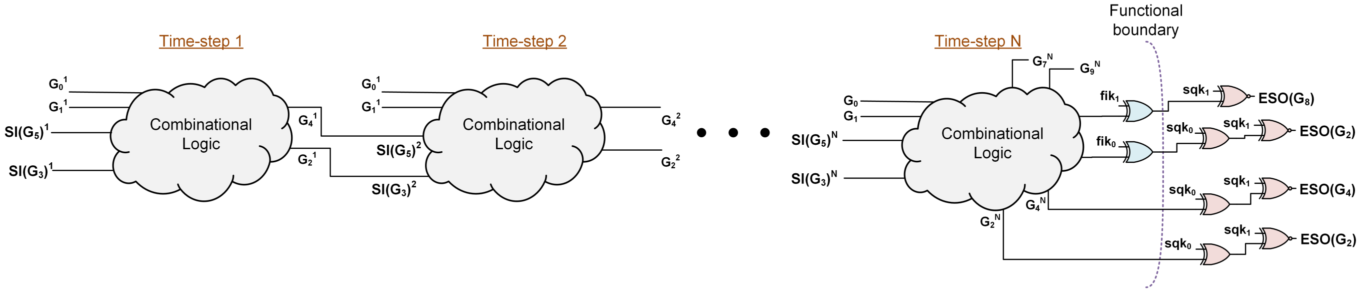

Fig. 12. Time-unrolled locked netlist used for multi-cycle-attack. Primary inputs are same for all time-frames, because input-register R_i (shown earlier in Figure 1) is scanned-in only once before first capture cycle. Primary outputs of intermediate stages are not observable because output-register R_o is scanned-out only once after the last (N^{th}) capture cycle.

TABLE III
RESILIENCE OF *SeqL* FOR ITC'99 SEQUENTIAL BENCHMARK CIRCUITS AND RISC-V. 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. ALL EXPERIMENTS ARE RUN ON *IBM BladeCenter®* *Cluster* WITH ABORT LIMIT OF 1 WEEK. *'✔' is secure and '✖' is insecure.*

Bench.	*#Gates*	*#SFFs*	*#SCs*	$\lvert R_{wof}\rvert$	*EFF* [18]		*SeqL*									
										Resilience			Decryption Time		
					Res.	Ov.	n	p	Ov.	N=1	N=2	N=5	N=1	N=2	N=5
b01	45	5	1	2	✖	9%	4	0.93	14 %	✔	✔	✔	22 *ms*	26 *ms*	72 *ms*
b02	26	4	1	1	✖	12%	3	0.88	18 %	✔	✔	✔	15 *ms*	13 *ms*	27 *ms*
b03	152	30	1	4	✖	14%	5	0.97	5 %	✔	✔	✔	4.2 *s*	0.14 *s*	0.23 *s*
b04	718	66	1	8	✖	8%	4	0.93	1.3 %	✔	✔	✔	50.9 *s*	1.9 *s*	0.4 *s*
b05	961	34	1	36	✖	3%	3	0.88	1 %	✔	✔	✔	15.3 *s*	1.2 *s*	0.42 *s*
b06	48	9	1	6	✖	14%	2	0.75	9 %	✔	✔	✔	0.1 *s*	24 *ms*	17 *ms*
b07	432	49	1	8	✖	9%	3	0.88	1.3 %	✔	✔	✔	34 *s*	1.2 *s*	1.5 *s*
b08	170	21	1	4	✖	10%	3	0.88	4.3 %	✔	✔	✔	1.8 *s*	78 *ms*	0.1 *s*
b09	168	28	1	1	✖	13%	2	0.75	2 %	✔	✔	✔	1.1 *s*	0.2 *s*	0.1 *s*
b10	189	17	1	6	✖	8%	3	0.88	1.5 %	✔	✔	✔	0.6 *s*	0.2 *s*	0.1 *s*
b11	757	31	1	6	✖	4%	2	0.75	0.3 %	✔	✔	✔	10.4 *s*	0.8 *s*	0.4 *s*
b12	1,065	121	2	6	✖	10%	2	0.75	0.35 %	✔	✔	✔	173 *s*	150 *s*	180 *s*
b13	342	53	1	10	✖	12%	4	0.93	1.9 %	✔	✔	✔	43 *s*	0.9 *s*	1.2 *s*
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	25,096	2,031	20	226	✖	7.9 %	10	0.99	0.09 %	✔	✔	✔	2 *min.*	13 *min.*	6 *hrs.*

functional-analysis-attacks on logic-locking (FALL) [5], and SMT-attack [4]. The codes for the FALL- and SMT-attacks are obtained from [39] and [40] repositories respectively. 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 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 *SeqL* automation [19]. For the oracle-less attacks, the resilience verification flow is slightly different because of the absence of oracle, however, *lcmp* verifier is still used for formal-equivalence-checking.

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

Table III shows the results of applying *SeqL* automation on ITC'99 open-source sequential gate-level benchmarks [38] and flattened `RISC-V CPU` netlist. The `RISC-V CPU` RTL is obtained from [41], and gate-level synthesis synthesis is performed at *Nangate 45nm* node [42] 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 columns *#SFFs*, *#SCs*, *Res.* and *Ov.* indicate number of SFFs, 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 [31]

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 modeled by time-unrolling the reverse-engineered netlist as well as the oracle in Figure 3, as shown in Figure 12.

Coming to test-time, since scan-in and scan-out phases span hundreds of clock cycles and N is in general relatively very small, running at slow-speed (considering challenges with at-speed test) will not significantly affect the attack time.

Empirical results are shown in Table III, where N denotes the number of capture-cycles in the multi-cycle scan based test. Similar to single-cycle attack ($N = 1$), *SeqL* was resilient to multi-cycle attack ($N = 2$, 5) across all benchmarks. For EFF, since key is successfully recovered for $N = 1$ itself, resilience results for $N > 1$ were not shown.

E. Resilience to Removal attack [9]

From Table III, we note that the number of locked flip-flops, $n = |K_{fi}| = |K_{sq}|$ in all cases is $<= 10$. 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 increase the user-configurable parameter γ in Algorithm *IBLA* [19], which results in exponential increase in sequential key search space, with linear increase in area overhead.

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

In *RDFS* [7], [31], special secure cells (SCs) are inserted into scan-chains to drive the key-gates. Unlike *RDFS*, in *SeqL* the 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 the 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 the scan-chain is itself locked in *SeqL*, it is mandatory to know the scan-key up front to run the automatic test pattern generation (ATPG) tool to be able to 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*.

G. Resilience to Scan-flushing attack [8]

Scan-flushing attack [8] corresponds to placing the design in scan-shift mode and flushing-in/out 1's and 0's with the expectation to reveal the K_{sq} portion of the *secret key* used to lock the scan-chain. Let's assume the locked scan-chain has n key-bits. If we shift-in 0-bit, after traversing through all the flip-flops in the scan-chain, the corresponding transformed shift-out bit is

$$\bigoplus_{k=1}^{k=n} sqk_k$$

On the other hand, if we shift-in 1-bit, after traversing through all the SFFs in the scan-chain, the corresponding transformed shift-out bit is

$$(\bigoplus_{k=1}^{k=n} sqk_k)'$$

In both these cases, the attacker is only able to derive the inversion parity of the locked scan-chain and not the exact key-bits. Hence, *SeqL* is resilient to *scan-flushing attack.*

H. Resilience of adjacent scrambling to SAT-Attack

Theorem VI.1. *The number of* scramble ECs *produced using the* adjacent scrambling *algorithm is* $2^{\lfloor \frac{\eta+1}{3} \rfloor}$.

Proof. From *Steinhaus-Johnson–Trotter algorithm* [43], we know that different vertices of a permutohedron can be visited through iterative swapping of the entries within the permutations. Inspired by this approach, Algorithm 1 swaps/scrambles two vertices per iteration in the graph G consisting of $(n+1)$ vertices in G (including the scan-out vertex).

- If these two vertices are adjacent, number of vertices that need to be eliminated is 3;
- If these two vertices are not adjacent, number of vertices that need to be eliminated is 4;

For every scramble-pair, multiple vertices get eliminated but only two possible *valid* paths exist. To estimate the upper bound, we need to consider the best-case scenario, which is the case when all scramble-pairs are adjacent because only 3 vertices get eliminated from each iteration. In this case, each iteration eliminates 3 vertices and creates 2 *valid paths* ($u \rightarrow v$ and $v \rightarrow u$ in Algorithm 1). Thus, the number of times Algorithm 1 iterates is $\lfloor \frac{\eta+1}{3} \rfloor$, defined by a cost constraint η.

Since each iteration decides 3 successive positions in the permutation, and the positions-under-scrutiny are mutually exclusive across iterations, the number of *HPs* multiply *geometrically*. For e.g. iteration-(1) scrambles vertices 1 and 2, iteration-(2) scrambles vertices 4 and 5, then:

- after iteration-(1), the scramble permutations are $\{\mathbf{1}, \mathbf{2}, 3, ...\}$ and $\{\mathbf{2}, \mathbf{1}, 3, ...\}$.
- after iteration-(2), the scramble permutations are $\{\mathbf{1}, \mathbf{2}, 3, \mathbf{4}, \mathbf{5}, 6, ...\}$, $\{\mathbf{1}, \mathbf{2}, 3, \mathbf{5}, \mathbf{4}, 6, ...\}$, $\{\mathbf{2}, \mathbf{1}, 3, \mathbf{4}, \mathbf{5}, 6, ...\}$ and $\{\mathbf{2}, \mathbf{1}, 3, \mathbf{5}, \mathbf{4}, 6, ...\}$.
- ...

This translates to the number of *HPs* produced after iteration-(1) being 2, after iteration-(2) being 4, after iteration-(3) being 8, and so on. In general, number of *HPs* produced after iteration-(i) is 2^i. Since the number of times Algorithm 1 iterates, prior to termination, is $\lfloor \frac{\eta+1}{3} \rfloor$, the number of *HPs* in the scramble graph produced through *adjacent scrambling* is $2^{\#iterations} = 2^{\lfloor \frac{\eta+1}{3} \rfloor}$, thus the proof. QED

I. Complexity Analysis

The SAT-based attack algorithm iteratively eliminates *invalid scramble ECs* using oracle input/response pairs, until it finds the correct scrambling key. As discussed in *Corollary V.2.1*, each *scramble EC* is of cardinality 1, hence the number of iterations the while loop in SAT-based attack [1] executes is equal to the number of *scramble ECs*$= 2^{\lfloor \frac{\eta+1}{3} \rfloor}$, thus ensuring $\mathcal{O}(2^{\lfloor \frac{\eta+1}{3} \rfloor})$ SAT-decryption time-complexity. In industry practice, for large processors, typically maximum scan-chain-length (n) is typically around $500 - 1000$, thus the SAT-attack complexity can be made arbitrarily large as shown in Table IV, making it practically impossible to decrypt the scrambling-key. Hence adjacent scrambling is computationally-secure against SAT-based attack.

We have discussed earlier in section V-D1 that the # of non-adjacent node constraints depend on the scrambling algorithm. Since by definition, *adjacent scrambling algorithm* swaps adjacent nodes, there are altogether $(n-1)+(\eta-1) = n+\eta-2$ adjacent node-pairs in the scramble-graph. All the remaining node-pairs in the complete digraph are non-adjacent, which equals $2 \times \binom{n}{2} - (n + \eta - 2) = n^2 - 2n - \eta + 2$. For each non-adjacent node-pair, there are $(n - 1)$ possible ways to be placed adjacent to the path, so altogether the number of non-adjacent node constraints are:

$$\begin{aligned}(n^2-2n-\eta+2)\times(n-1) &= n^3-2n^2-\eta.n+2n-n^2+2n+\eta-2 \\ &= n^3 - 3n^2 + 4n - \eta.n + \eta - 2 \quad (2)\end{aligned}$$

TABLE IV

CNF AND SAT-BASED ATTACK STATISTICS FOR *SeqL+* OBFUSCATED CIRCUITS WITH $\#SFFs > 50$ BUT $R_{wof} < 50$. FOR SUCH DESIGNS, WE ASSUME ONLY SCRAMBLING IS USED, AND SCAN-LOCKING IS NOT USED. ALGORITHM 1 WAS USED FOR SCRAMBLING THE SCAN-CHAINS. PLEASE THAT HERE, $\eta = n$ IS USED I.E. ALL THE SCAN FLIP-FLOPS WERE USED FOR SCRAMBLING. THUS, THE OVERHEADS WILL BE FURTHER LESS FOR SMALLER VALUES OF η.

Bench.	# Gates (g)	#*SFFs*	n	R_{wof}	Ite. Dec. time (τ_0)	# *Literals*	# *Clauses*			#*Iters.* (# eq. cls.)	Est. Tot. Dec. time (τ)	Ov.
						$(2n^2+8n+g)$	HP Cons. ($\lvert C_{HP}\rvert$) $(2n^3-5n^2+7n-2)$	Conn. Cons. ($\lvert C_I \cup C_O\rvert$) $(12n-6)$	Ckt. Cons. ($\lvert C_{Combo}\rvert$) ($\mathcal{O}(g)$)	$(2^{\lfloor\frac{n+1}{3}\rfloor})$	$\tau_0 * 2^{\lfloor\frac{n+1}{3}\rfloor}$	
b04	718	66	66	8	51 s	$10^{4.0}$	$10^{5.7}$	$10^{2.9}$	$10^{3.3}$	$10^{6.62}$	6.7 **years**	27.3 %
b12	1,065	121	121	6	173 s	$10^{4.5}$	$10^{6.5}$	$10^{3.2}$	$10^{3.5}$	$10^{12.04}$	$10^{6.8}$ **years**	17.5 %
b13	342	53	53	10	43 s	$10^{3.8}$	$10^{5.5}$	$10^{2.8}$	$10^{3.0}$	$10^{5.42}$	131 **days**	37 %
b18	114,561	3,320	1,000	23	53 $hrs.$	$10^{6.3}$	$10^{9.3}$	$10^{4.1}$	$10^{5.5}$	$10^{100.2}$	$10^{97.9}$ **years**	0.3 %
b19	231,266	6,642	1,000	30	91 $hrs.$	$10^{6.4}$	$10^{9.3}$	$10^{4.1}$	$10^{5.8}$	$10^{100.2}$	$10^{98.2}$ **years**	0.1 %
b20	20,172	490	490	22	7 $min.$	$10^{5.7}$	$10^{8.4}$	$10^{3.8}$	$10^{4.8}$	$10^{49.1}$	$10^{44.2}$ **years**	1.5%
b21	20,517	490	490	22	6 $min.$	$10^{5.7}$	$10^{8.4}$	$10^{3.8}$	$10^{4.8}$	$10^{49.1}$	$10^{44.1}$ **years**	1.5 %
b22	29,897	735	735	22	11 $min.$	$10^{6.1}$	$10^{8.9}$	$10^{4.0}$	$10^{5.0}$	$10^{73.8}$	$10^{69.1}$ **years**	1.0 %

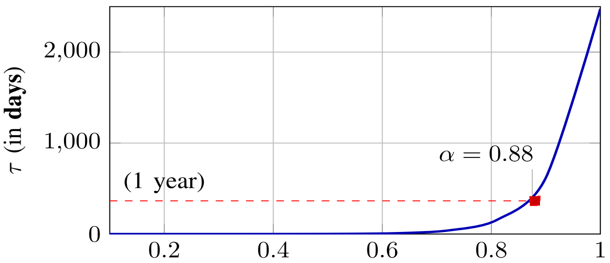

Fig. 13. Estimated decryption time ($\tau = \tau_0 * 2^{\lfloor\frac{\alpha.n+1}{3}\rfloor}$) for b04 (the smallest circuit under consideration for scrambling), as a function of area-cost constraint $\alpha = \frac{\eta}{n}$ ($0 < \alpha \leq 1$). Please note the Y-axis range.

Substituting this in the results from section V-D1, we get

$$|C_{HP}| = 2n \times \left(1 + \binom{n}{2}\right) + (n^3 - 3n^2 + 4n - \eta.n - \eta - 2)$$
$$= 2n + n^2 \times (n-1) + (n^3 - 3n^2 + 4n - \eta.n + \eta - 2)$$
$$= 2n^3 - n^2(4+\alpha) + n(6+\alpha) - 2,\ 0 < \alpha = \frac{\eta}{n} \leq 1 \quad (3)$$

This suggests the worst-case *HP constraint* complexity is $\mathcal{O}(n^3)$ (because $\alpha \leq 1$). We have seen earlier that the connection constraint complexity is $\mathcal{O}(n)$ and combinational circuit constraint complexity is $\mathcal{O}(g) = \mathcal{O}(n)$ (because the ratio of flip-flops to gates lies in a restricted range [44]. Table IV also reflects this observation.) Thus, the worst-case total CNF reduction complexity is $\mathcal{O}(n^3) + \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n^3)$.

The last-but-one column in Table IV shows the practical impossibility to launch the SAT-based attack on the scrambled instances, hence we report the decryption time per iteration in the sixth column of this table. For b19 processor [38] when scrambled with $\eta = n$ results in only 0.1% overhead, but we notice 91 $hrs.$ decryption time per iteration and a total of $10^{100.2}$ iterations needed to decrypt the scrambling key. This causes the estimated decryption time to be $10^{98.2}$ years, thus demonstrating the power of the proposed technique. Further, Figure 13 shows exponential increase in the estimated decryption time as a function of the $\alpha = \frac{\eta}{n}$ ($0 < \alpha \leq 1$), providing an opportunity to trade-off area with decryption time.

J. Testing

The testing of key-gates in combinational logic (fik) is straightforward, as the faults in the output of these key-gates are appended to the fault list before ATPG is invoked. Coming to key-gates along scan-chain (sqk), there is no need for additional patterns because we consider only XOR/XNOR type gates and chain-test will automatically test faults on sqk-type gates (fault-equivalence).

K. Cost Evaluation

The cost evaluation of scan-locking is provided in the conference version [19]. Coming to scrambling, there are K 2 : 1 MUXes, each costing two 2-input and gates and one 2-input or gate. So, the total transistor cost is $\mathcal{O}(K)$. Further, each 2 : 1 MUX demands two additional inputs (one scrambling input signal and one select signal), and there are K such MUXes. So, the total routing cost is also $\mathcal{O}(K)$. Since with linear increase in cost (transistors + routing), an exponential increase in security level is achieved, *scan-scrambling* looks attractive. Moreover, it is sufficient to *scramble* only one scan-chain, making it cost-effective. The last column in Table IV shows the overheads of scrambling. The overhead decreases with an increase in circuit complexity, demonstrating the scalability of the proposed technique. Please that here, $\alpha = 1$ is used i.e. all the SFFs were used for scrambling, yet the area overhead is acceptable for large designs. Thus, the overheads will be further less for smaller values of α.

VII. CONCLUSIONS

We have proposed *SeqL+*, which performs functional isolation, FI-SQ locking, and scan-scrambling. *SeqL+* hides a major fraction of the functionally correct keys, thus maximizing functional output corruption, and also embeds exponentially many number of *Hamiltonian Paths* into the *scramble digraph* thereby thwarting *brute-force attacks*. We have shown both the theoretical and empirical improvements in the security of *SeqL+* compared to the state-of-the-art. The results have shown 100% resilience to state-of-the-art oracle-guided as well as oracle-less attacks. Furthermore, since the combinational key (excluding FIs) is completely recovered, it is sufficient to lock FI-SQ pairs and scramble only a single scan-chain containing an adequate number of flip-flops, making *SeqL+* efficient in terms of overheads. Moreover, we have demonstrated implementation on large-scale designs such as RISC-V CPU and b19, demonstrating its applicability in mainstream industry practice.

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

[9] M. Yasin et al, “Removal attacks on logic locking and camouflaging techniques,” in *IEEE TETC*, 2017.

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

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

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

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

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

[15] Y. Xie and A. Srivastava, “Anti-SAT: Mitigating SAT attack on logic locking,” *IEEE TCAD*, vol. 38, no. 2, pp. 199–207, 2018.

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

[17] 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.

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

[19] S. Potluri, A. Aysu, and A. Kumar, “SeqL: Secure Scan-Locking for IP Protection,” in *ISQED*, 2020, pp. 7–13.

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

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

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

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

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

[25] Y. Zhang et al, “TGA: An Oracle-Less and Topology-Guided Attack on Logic Locking,” in *ASHES*, 2019, p. 75–83.

[26] P. Chakraborty et al, “SAIL: Machine learning guided structural analysis attack on hardware obfuscation,” in *IEEE AsianHOST*, 2018, pp. 56–61.

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

[28] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Alqutayri, and O. Sinanoglu, “Scansat: Unlocking static and dynamic scan obfuscation,” *IEEE TETC*, pp. 1–1, 2019.

[29] “ASE Technology Holding Revenue 2006-2020,” 2020. [Online]. Available: https://www.macrotrends.net/stocks/charts/ASX/ase-technology-holding/revenue/

[30] “ASE Group Test Service,” 2021. [Online]. Available: https://ase.aseglobal.com/en/products/test

[31] U. Guin et al, “A novel design-for-security architecture to prevent unauthorized IC overproduction,” in *IEEE VTS*, 2017, pp. 1–6.

[32] “Intel Set to Outsource Select CPU Production to TSMC’s 5nm Process,” 2021. [Online]. Available: https://www.allaboutcircuits.com/news/intel-set-to-outsource-select-cpu-production-tsmcs-5nm-process/

[33] M. E. Massad et al, “Reverse engineering camouflaged sequential circuits without scan access,” in *ICCAD*, 2017, pp. 33–40.

[34] X. Wang et al, “Secure scan and test using obfuscation throughout supply chain,” *IEEE TCAD*, vol. 37, no. 9, pp. 1867–1880, 2018.

[35] D. Hely et al, “Scan design and secure chip,” in *IEEE IOLTS*, 2004, pp. 219–224.

[36] S. A. Cook, “The complexity of theorem-proving procedures,” in *IN STOC*. ACM, 1971, pp. 151–158.

[37] T. Qinhan et al, “Efficacy of sat-based attacks in the presence of circuit reverse-engineering errors,” in *IEEE ISCAS*, 2020, pp. 1–5.

[38] “ITC’99 benchmarks.” [Online]. Available: https://www.cerc.utexas.edu/itc99-benchmarks/bench.html

[39] D. Sirone et al. Functional analysis attacks on logic locking benchmark circuits, benchmark circuits and codes. [Online]. Available: https://bitbucket.org/spramod/fall-attacks/src/master/

[40] K. Z. Azar et al. SMT decryption tool binaries, benchmarks, and codes. [Online]. Available: https://github.com/gate-lab/SMTAttack

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

[42] The Silvaco 45nm Open Cell Library. [Online]. Available: https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library/

[43] L. M. Surhone et al, *Steinhaus-Johnson-Trotter Algorithm*. Beau Bassin, MUS: Betascript Publishing, 2010.

[44] S. Bhunia et al, “Low-power scan design using first-level supply gating,” *IEEE TVLSI*, vol. 13, no. 3, pp. 384–395, 2005.

Seetal Potluri received his Ph.D.from the Department of Electrical Engineering, Indian Institute of Technology Madras, on ”Power : Its Manifestations in Digital Systems Testing” in 2015. He has worked at Xilinx between 2016-2018, and his work on ”Delta-IDDq” is currently in production on ”Automotive-grade” Zynq ICs. Currently, he is pursuing a Post-Doc at North Carolina State University, USA. He has served on the Technical Program Committees of IEEE Asia South Pacific Design Automation Conference (2018, 2019, 2020), IEEE European Test Symposium (2016, 2017, 2018), IEEE Asian Test Symposium (2017), and IEEE International Test Conference Asia (2017 and 2020). He has published 22 research papers, an approved WIPO patent, and is an IEEE member.

Shamik Kundu is a doctoral student in the Department of Electrical and Computer Engineering at the University of Texas, Dallas. He received his B.Tech degree in Electronics and Communications Engineering from Heritage Institute of Technology in 2018. His research interests include hardware and system security, fault detection, and modeling in hardware architectures.

Akash Kumar (SM’13) received the joint Ph.D. degree in electrical engineering and embedded systems from the Eindhoven University of Technology, Eindhoven, The Netherlands, and the National University of Singapore (NUS), Singapore, in 2009. From 2009 to 2015, he was with NUS. He is currently a Professor with Technische Universität Dresden, Dresden, Germany, where he is directing the Chair for Processor Design. His current research interests include the design, analysis, and resource management of low-power and fault-tolerant embedded multiprocessor systems.

Kanad Basu received his Ph.D. from the Department of Computer and Information Science and Engineering, University of Florida. His thesis was focused on improving signal observability for post-silicon validation. Post-Ph.D., Kanad worked in various semiconductor companies like IBM and Synopsys. During his Ph.D. days, Kanad interned at Intel. Currently, Kanad is an Assistant Professor at the Department of Electrical and Computer Engineering of the University of Texas at Dallas. Prior to this, Kanad was an Assistant Research Professor at the Electrical and Computer Engineering Department of NYU. He has authored 2 US patents, 2 book chapters, and several peer-reviewed journal and conference articles. Kanad was awarded the ”Best Paper Award” at the International Conference on VLSI Design 2011. Kanad’s current research interests are hardware and systems security.

Aydin Aysu (SM’19) received his Ph.D. degree in Computer Engineering from Virginia Tech in 2016. He was a post-doctoral research fellow at the University of Texas at Austin from 2016 to 2018. He is currently an assistant professor and Bennet faculty fellow in the Electrical and Computer Engineering Department of North Carolina State University. Dr. Aysu’s research focuses on the development of secure systems that prevent cyberattacks targeting hardware vulnerabilities. His research interests lie at the intersection of applied cryptography, digital hardware design, and computer architectures. He received the 2020 NSF CAREER award, 2020 DATE best paper award, 2019 GLSI-VLSI best paper award, 2019 NC State FRPD award, 2018 NSF CRII award, and is an IEEE senior member.