SeqL: SAT-attack Resilient Sequential Locking

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

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

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

***Abstract*—SAT-attack is known to successfully decrypt a *functionally correct key* of a locked combinational circuit. It is possible to extend the SAT-attack to sequential circuits through the scan-chain by selectively initializing the combinational logic and analyzing the responses. Recently, sequential locking was proposed as a defense to SAT-attack, which works by locking the scan-chains of flip-flops. *ScanSAT* [1], however, showed that it is possible to convert the sequentially locked instance to a locked combinational instance, and thereby decrypt the entire sequential key using SAT-attack. In this paper, we propose *SeqL*, a secure sequential lock defense against *ScanSAT*.**

***SeqL* provides functional isolation, and also encrypts selective flip-flop inputs, thereby mitigating *ScanSAT* and other related SAT-attacks. We conduct a formal study of the sequential locking problem and demonstrate automating our proposed defense on any given sequential circuit. We show that *SeqL* hides functionally correct keys from the attacker, thereby increasing the likelihood of functional output corruption. When tested on sequential benchmarks (ITC'99) and pipelined combinational benchmarks (ISCAS'85, MCNC), *SeqL* gave 100% resilience to *ScanSAT*. We also show that *SeqL* has expoential time complexity and hence is provably secure against removal attack.**

***Index Terms*—Logic Locking, SAT-Attack, Sequential Locking**

I. Introduction

Dramatic increases in the cost of fabrication technology have shifted the semiconductor business to a contract foundry model, where leading-edge design houses outsource fabrication, assembly and testing steps to offshore facilities with lower operational costs. However, integrated circuit (IC) piracy, overbuilding, and counterfeiting in the untrusted offshore foundry have caused major concerns in electronic and defense industries [2]–[4]. United States government has recently declared trade war against loose intellectual property rights (IPR) protection laws in Asia, and declared that this has forced technology transfer and alleged IP theft [5]. Subsequent tariffs imposed by the U.S. government on semiconductor imports, will cost millions of dollars annually [6].

Reverse engineering the mask reveals the netlist, while analyzing the scan data either on an activated IC or at an outsourced tester reveals critical design information. This raises concerns regarding piracy, reverse engineering, overproduction, IPR violation, and hardware trojan insertion and counterfeit electronics [7]–[9]. Thus, it is imperative to consider trust in early phases of chip design, to secure it in the entirety of supply chain. Logic locking is one such holistic solution that was touted to address all the aforementioned threats.

A. Related work

Logic locking uses a low-overhead combinational chip-locking system to combat these issues [4]. Its purpose is to generate circuits that will reveal the correct output if and only if the secret key is entered correctly. Basically, logic locking encrypts selective nodes inside the circuit by adding a logical gate (such as XOR/XNOR/OR/AND/MUX, etc.) with one input driven by a secret key. Hence, the function of the circuit will not change if this key is entered correctly. But if the key is incorrect, the function can alter.

Several logic locking techniques have been proposed so far in literature [4], [10]–[13]. But the SAT-attack [9] is shown to successfully decrypt the encryption keys in all those cases, with over 95% success rate. Several defenses were then proposed to mitigate SAT-attack, such as *Anti-SAT* [14], *SARLock* [15] and *Cyclic Obfuscation* [16], but they have failed to address the vulnerability to *AppSAT* [17], *Double-DIP* [18] and *CycSAT* [19] attacks. Secure Function Logic Locking (SFLL) [20] was the only logic locking scheme that is resilient to most of the above attacks, but it is recently attacked by functional analysis of logic locking (FALL) attack [21] with 81% success. Moreover, all of the aforementionted techniques are limited to combinationally locked circuits or locking combinational portions of the sequential circuits.

Most of the real-world circuits are sequential in nature. It is possible to launch the SAT-attack on sequential circuits, by initializing the combinational logic and capturing/analyzing the responses using scan-chains, on an activated IC or at an outsourced tester.

To address this issue, encrypt flip-flop (*EFF*) based sequential locking [22] was recently proposed as a defense to SAT-attack on sequential circuits. In *EFF*, flip-flip outputs (FOs) are locked, and the locked FOs drive both the combinational logic as well as the next flip-flops in the scan chain. *ScanSAT* [1], however, showed that it is possible to convert the *EFF*-style sequentially locked instance to a locked combinational instance, and thereby decrypt the entire sequential key using SAT-attack. Although, existing works like *SARLock* [15] and *AppSAT* [17] consider sequential circuits, they target unrolled versions and do not consider FO locking.

B. Contributions

The main contributions of this paper are as follows:

1) We identify there is 100% correlation between flip-flop input (FI) locking and functional output corruption;

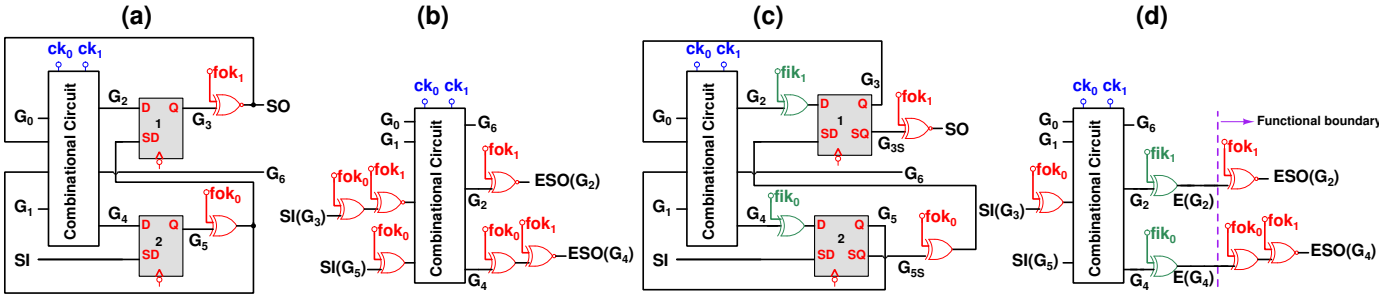


Fig. 1. (a) *EFF*-style: Sample sequential circuit with logic locking and FO locking; (b) Scan-unrolled equivalent combinational circuit for Figure 1(a); (c)*SeqL*-style: Circuit in Figure 1(a), with functional isolation and locked FIs; and (d) Scan-unrolled equivalent combinational circuit for Figure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

2) Exploiting this property, we propose *SeqL*, a defense to *ScanSAT*, that has two properties: (a) it isolates functional path from the locked scan path; (b) it locks FIs and causes functional output corruption, thus making the sequential circuit resilient to SAT-attack;
3) *SeqL* hides majority of the total number of functionally correct keys. This minimizes the fraction of functionally correct keys, among scan-correct keys, thus maximizing the likelihood of functional output corruption;
4) The small area, timing and energy-per-toggle overheads of *SeqL* and its ease of implementation makes it attractive for industry practice.

C. EFF [22] and ScanSAT [1]

The *EFF* sequential locking technique is done in addition to combinational logic locking. In *EFF*, flip-flops on the non-critical timing paths of the sequential circuits are selected, and XOR/XNOR-type key gates are added to encrypt the Q-outputs similar to how combinational circuits are locked. The encrypted Q-output of a scan-flip-flop reaches combinational logic, as well as the scan-input of the next flip-flop in the scan chain.

Figure 1(a) shows a sample sequential circuit with all flip-flops encrypted using *EFF*-style sequential locking. In Figure 1(a):

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

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

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

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

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

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

D. Threat model

The design house receives locked IPs from third-party vendors, and integrates them with in-house designed IPs, and prepares the final locked netlist. After physical design is completed, the layout is sent to the foundry for fabrication. We consider a malicious foundry that offers fabrication, assembly and testing services [23]. Thus, the attacker has access to layout and mask information, and is thus able to reverse-engineer the gate-level netlist. There are two possible instances, where the attacker at the malicious foundry is able to launch the attack:

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

the dies in EDT-bypass mode, applies desired input as well as scan patterns and collects the outputs as well as scan response data.

E. Organization

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

II. Solution Insight

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

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

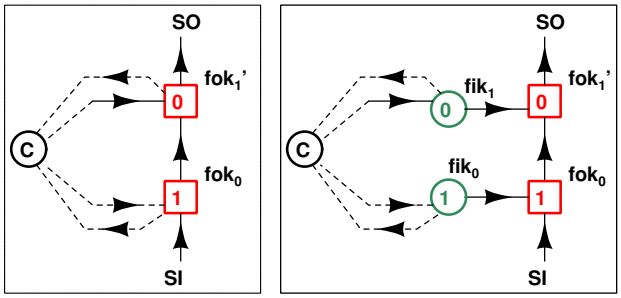
Figure 1(c) shows the proposed idea by transforming the circuit in Figure 1(a), using above principles. Figure 1(c) is different from Figure 1(a) in two ways:

- There is a separate Q and SQ, and FO key gate is added at SQ, thus leaving the functional output Q unencrypted. This is referred to as *functional isolation*;
- Extra key gates,(both of XOR type in this case) are added at FIs of both the flip-flops. fik_0 and fik_1 the FI locking key bits. These key gates are referred to as FI key gates in the rest of this paper.

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

However, the circuit's normal functional operation is purely influenced by $E(G_2)$ and $E(G_4)$, and the XOR/XNOR-chains (in red) cease to exist. As a result, the FO portion of the sequential key returned by the SAT-solver (which is correct for scan operation) will not affect the circuit's normal operation, (intuitively) can lead to corruption of the functional outputs. When the following are inputted to the formal equivalence checker

- the combinational portion of the sequential circuit shown in Figure 1c;



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

Fig. 2. Abstract models for circuits in Figures 1(a) and 1(c)

- the combinational portion of the the original unencrypted sequential circuit; and
- the combination portion of this solver key K,

the result will be either *equivalent* or *different*. In the example shown in Figure 1(d), it was found that the result was *different*, or in other words *not equivalent*. Hence, by functional isolation and FI locking, functional output corruption was achieved. This behavior is explained using an abstract model, in the next section.

A. Abstract model

Figure 2 shows an abstracted model of the sequential circuit, with the combinational logic abstracted into a source-sink circular vertex *C*, each FI key-gate abstracted into a green circular vertex, and each flip-flop key-gate abstracted into a red rectangular vertex. Figures 2(a) and 2(b) show the models corresponding to sequential circuits in Figures 1(a) and 1(c) respectively. The green circles correspond to FIs and red rectangles correspond to flip-flops. In the abstract model corresponding to the proposed flip-flop locking shown in Figure 2(b), the following are functional paths:

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

and the following are scan-out paths:

- $SP_0 : fik_0 \longrightarrow fok_0 \longrightarrow fok_1^{'} \longrightarrow SO$
- $SP_1 : fik_1 \longrightarrow fok_1^{'} \longrightarrow SO$

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

The inferences from this experiment are:

1) If the sequential circuit is applied with the key returned by SAT solver, the circuit is guaranteed to function correctly during scan operation, but not during *functional operation*; and
2) It is possible to achieve functional output corruption, by isolating the functional path from the locked scan path;
3) The FI XOR/XNOR-type key gates, cascade with the unrolled XOR/XNOR-chain of the scan path, and thereby

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

fik_1	fok_1'	fik_0	fok_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

obfuscating the SAT-solver to return the *functionally incorrect key*; and

4) The combinational portion of the returned solver key, excluding the FIs, i.e., $\{ck_0, ck_1\} = \{1, 0\}$, is correct, while the portion of the key corresponding to the flip-flops and FIs is incorrect. We shall see in later sections, that this is true in general, hence it is sufficient to lock the FIs and flip-flops (which contribute to security), thus saving area and power.

To understand this behavior more systematically, table I enumerates all possibilities for the sequential lock $\{fik_1, fok_1, fik_0, fok_0\}$ for the circuit in in Figure 1c.

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

Figure 3 shows the key assignment graph (KAG) for the circuit shown Figure 1(c). The sequential key returned by the solver, as discussed previously, corresponds to the second leaf from the left. Since this leaf is a functional incorrect key, the technique is able to achieve functional output corruption. In this example, odds against the functionally key is $p = \frac{3}{4} = 0.75$, hence the chance of functional output corruption is higher than otherwise.

There is one additional benefit. *In Table I, there are few rows in which, it is FALSE for scan-correctness, while it is TRUE for functional-correctness. This indicates that our technique* **hides** *three out of four functional-correct keys from the attacker.*

B. Analysis

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

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

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

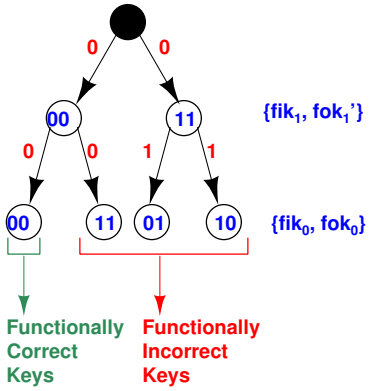


Fig. 3. Key Assignment Graph (KAG) for circuit in Figure 1(c). KAG is a binary tree, the leaves of which correspond to the rows in Table I, where the scan-correctness column is TRUE

Definition: In KAG, the children of every vertex at depth i from the root, has to satisfy scan-correctness for $(i+1)^{th}$ flip-flop from the end of the scan-out-path.

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

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

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

Definition: The possible weights on the edges in KAG are 0 or 1, which signifies parity. The parity of an edge is the same as the parity of the corresponding child.

Definition: inv_k equals 0 or 1, depending on whether k^{th} scan-cell along scan-chain from the scan-output is XOR or $XNOR$ respectively.

Theorem: 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,

$$(fik_i \oplus fok_i) \oplus \sum_{k=1}^{i-1} (fok_k \oplus inv_k)$$

should equal 0. If

$$\sum_{k=1}^{i-1} (fok_k \oplus inv_k)$$

equals 0, $(fik_i \oplus fok_i)$ becomes 0 (possible children of v_i are 00 and 11, in both cases parity of edge is 0).

On the other hand, if

$$\sum_{k=1}^{i-1}(fok_k \oplus inv_k)$$

equals 1, $(fik_i \oplus fok_i)$ 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: KAG is a binary tree.
Proof: Root vertex has exactly two children. Additionally, every non-root vertex has exactly two children. Since every vertex in KAG has exactly two children, KAG is a binary tree, hence the proof.

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

Theorem: There is exactly one functionally correct leaf in KAG.
Proof: *The path from the root to a functionally correct leaf should have* 00 *nodes. There is exactly one such leaf in* KAG*, whose path from the root consists of only* 00 *vertices, hence the proof.*

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

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

Proof:

1) *In the removal attack, the attacker needs to check if each key-gate removed introduces an inversion at the output of flip-flop.*
2) *Since there are* 2^n *possibilities, let* $L = \mathcal{O}(n)$ *be the logic simulation time of circuit, the time it takes to launch removal attack is* $2^n * L$*. Thus, time complexity is* $\mathcal{O}(n.2^n) = \mathcal{O}(2^n)$*, hence the proof.*

III. Automating SeqL Defense

Objective: *Lock selective FI, FO pairs such that functional output corruption is achieved, while area-overhead is minimized.*

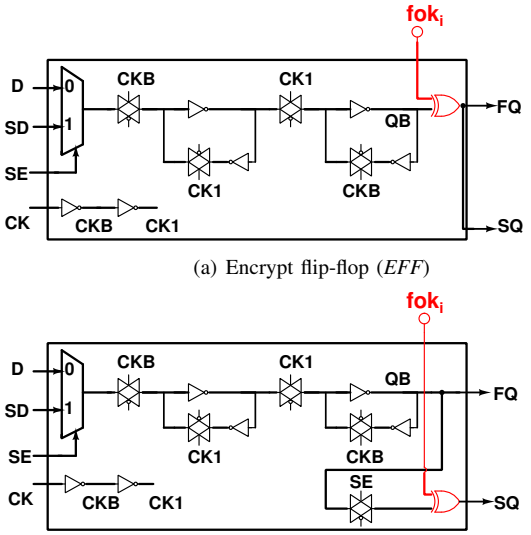
Solution: *The likelihood of functional output corruption is maximized with increase in* $p = 1 - \frac{1}{2^n}$*, whereas area-overhead increases linearly with* n*. Hence, the chances of functional output corruption increases very quickly with* n*, with minimal increase in area overhead. The proposed solution i.e.,* SeqL *exploits this principle to iteratively lock FI, FO pairs beginning at the end of the scan-chain(s), until functional output corruption is achieved.*

The proposed sequential locking, i.e., SeqL *has two parts:*

1) *An isolation-based flip-flop locking; and*
2) *An iterative FI locking algorithm.*

A. Isolation-based flip-flop locking

Define the sequential key to be $K = \{K_c, K_{fi}, K_{fo}\}$*, where* K_c*,* K_{fi} *and* K_{fo} *are portions of the key that lock the combinational logic (excluding the FIs), the FIs and the flip-flops respectively. In* EFF *technique, all these components influence the sequential circuit's normal functional operation. Figure 4(a) shows the* EFF*-style flip-flop 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 flip-flop locking is shown in Figure 4(b), which isolates the functional path from the locked scan path. Hence, the FO 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_{fo} *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.*



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

(b) *SeqL* flip-flop

Fig. 4. Flip-Flop variants for Sequential Locking

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

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

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

Figure 5 shows the ScanSAT *resilience verification flow used iteratively in* SeqL*. Algorithm 1 generates* C_3 *shown*

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

Input: C

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

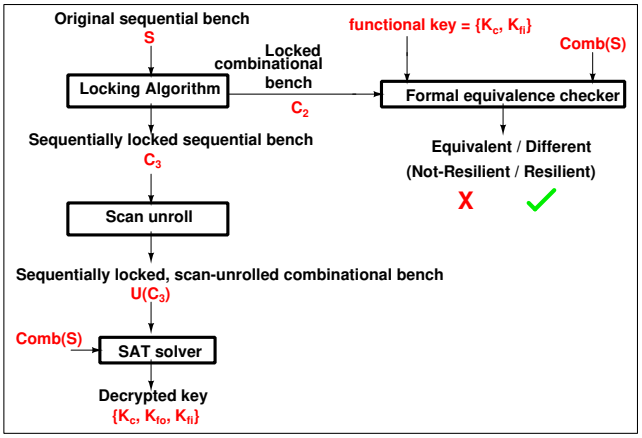


Fig. 5. *ScanSAT* resilience verification flow

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

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

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

Algorithm 2: Iterative boundary locking algorithm for sequential circuits

Input: C, K_{fo}

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

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

IV. Experimental Evaluation

In order to perform ScanSAT *resilience evaluation, we have used* sld *key generation solver and* lcmp *formal-equivalence checkers provided by [9]. Both the solvers use the* .bench *format for the input benchmark circuit. In all the subsequent experiments, we have used the open-source* .bench *designs for ITC'99 sequential benchmarks available at [24] and logic-encrypted ISCAS'85, MCNC combinational benchmarks provided by [9]. Algorithms 2 (*IBLA*) and 1 were used for sequentially locking the sequential benchmarks and pipelined combinational benchmarks, respectively. Both the locking algorithm were implemented in* Perl. *Since the locking algorithm execution times across all the benchmarks is very small (matter of seconds), the execution times were not reported.*

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

Table III shows the results of applying the procedure shown in Figure 5 on 4 different encryption schemes validated in [9], and compared against EFF *[22]. This table shows that* SeqL *secured all sequential circuits against* ScanSAT *in* 100% *of the cases. As explained in Section II,*

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

Results on $IOLTS'14$ *gave 0% resilience in* EFF *case and 100% resilience in* SeqL *case, across all benchmarks, hence not reported in Table III for shortage of space. On the other*

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

	EFF [22]			*SeqL*				
Bench.	Res.	Ov.	DT.	Res.	n	p	γ	DT.
b01	✖	9%	$10ms$	✔	4	0.93	20%	$10ms$
b02	✖	12%	$10ms$	✔	3	0.88	20 %	$10ms$
b03	✖	14%	$2.1s$	✔	5	0.97	20 %	$4.4s$
b04	✖	8%	$52s$	✔	4	0.93	10 %	$57s$
b05	✖	3%	8.2s	✔	3	0.88	5%	15s
b06	✖	14%	$10ms$	✔	2	0.75	20 %	$0.1s$
b07	✖	9%	$31s$	✔	3	0.88	10 %	$38s$
b08	✖	10%	$0.5s$	✔	3	0.88	15%	$1.5s$
b09	✖	13%	$0.9s$	✔	2	0.75	15 %	$1.0s$
b10	✖	8%	$0.3s$	✔	3	0.88	10%	0.5
b11	✖	4%	$5.5s$	✔	2	0.75	5%	$9.5s$
b12	✖	10%	$67s$	✔	2	0.75	10%	$163s$
b13	✖	12%	$24s$	✔	4	0.93	15%	$37s$
b14	✖	1.0%	$109s$	✔	8	0.99	1%	$14m$
b15	✖	0.7%	$79s$	✔	9	0.99	1%	$35m$
b17	✖	0.3%	$187s$	✔	16	0.99	0.5%	$37m$

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

Bench.	RND		DAC'12		ToC'13/xor		ToC'13/mux	
	EFF	*SeqL*	*EFF*	*SeqL*	*EFF*	*SeqL*	*EFF*	*SeqL*
apex2	✖	✔	✖	✔	✔	✔	✖	✔
apex4	✖	✔	✖	✔	✔	✔	✔	✔
i4	✔	✔	✖	✔	✔	✔	✖	✔
i7	✔	✔	✖	✔	✔	✔	✖	✔
i8	✔	✔	✖	✔	✔	✔	✖	✔
i9	✔	✔	✖	✔	✖	✔	✖	✔
seq	✔	✔	✖	✔	✔	✔	✖	✔
k2	✔	✔	✖	✔	✔	✔	✖	✔
ex5	✔	✔	✖	✔	✔	✔	✖	✔
ex1010	✔	✔	✖	✔	✔	✔	✖	✔
dalu	✔	✔	✖	✔	✔	✔	✖	✔
des	✔	✔	✖	✔	✔	✔	✖	✔
c432	✖	✔	✖	✔	✖	✔	✖	✔
c499	✖	✔	✖	✔	✖	✔	✖	✔
c880	✔	✔	✔	✔	✖	✔	✖	✔
c1355	✔	✔	✖	✔	✔	✔	✖	✔
c1908	✔	✔	✖	✔	✖	✔	✖	✔
c3540	✔	✔	✖	✔	✖	✔	✖	✔
c5315	✔	✔	✔	✔	✖	✔	✖	✔
c7552	✔	✔	✖	✔	✖	✔	✖	✔

hand, results on $DTC'10/LUT$ were not reported because it is LUT-based.

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

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

A. Overheads

Table IV shows the comparison of area, timing and energy for original, EFF-type and proposed $SeqL$-type locked 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 IV, it is evident that the proposed $SeqL$-style locked scan flip-flop has 22% and 19% reduction in $T_{CK-to-Q}$ and Energy-Per-Toggle (EPT) respectively, with only 4% area overhead as compared to EFF-style locked scan flip-flop.

B. Decryption time

Comparison of decryption times for ITC'99 circuits were shown earlier in Figure II. Figure 6 shows the comparison of key decryption times of SeqL *(red) with* EFF *(blue) techniques for pipelined combinational benchmarks, on the 4 different encryption schemes validated in [9]. The benchmarks are shown on X-axis and the key decryption time is shown on a logarithmic scale on Y-axis. Clearly,* SeqL *is superior with regard to decryption time, as compared to* EFF*, across all the benchmarks, across all the encryption categories.*

V. DISCUSSION

A. Comparison to Timing-Driven SAT Defenses

In [25], delay-locking *was proposed to mitigate SAT-attack. In* delay-locking*, the key not only determines the functionality of the circuit but also its timing profile. A functionally-correct but timing-incorrect key will result in timing violations and thus make the circuit malfunction. In [26], a novel SAT formulation based approach called* TimingSAT *was used to deobfuscate the functionalities of such delay locked designs within a reasonable amount of time. Since delay locking is orthogonal to* EFF *in defending SAT-attack, we do not compare with* delay-locking *or* TimingSAT*.*

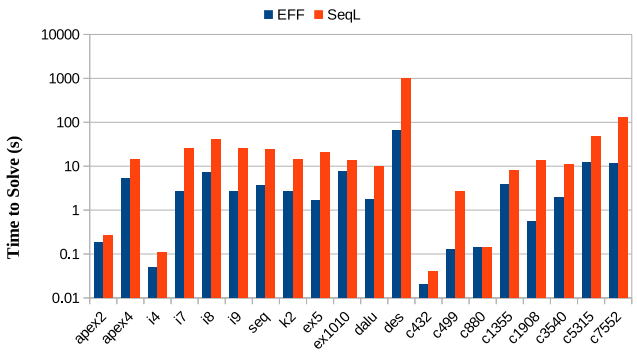
B. Limitations of SeqL

While ScanSAT *[1] uses* EFF *as a representative example, there are other sequential locking variants. While* EFF *uses statically obfuscated scan-chains, a dynamically-obfuscated-scan (DOS) architecture is proposed in [27] and design-for-security (DFS) architecture is proposed in [28]. We do not cover those extensions in this work.*

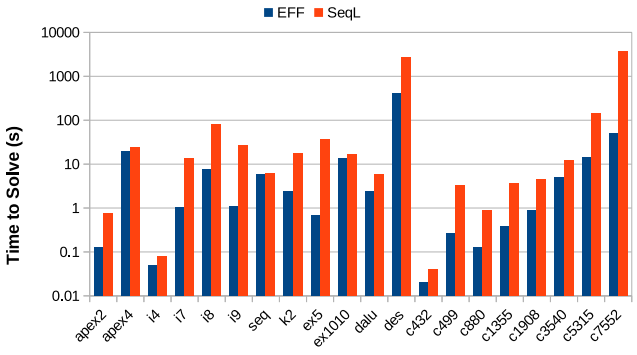
From Table II, we note that value of $n = |K_{fi}| = |K_{fo}|$ in most cases is less than 10. So, the total number of possible sequential key bits is $|K_{fi}| + |K_{fo}| < 20$, hence it possible to find the functionally correct key using brute-force-search. Solution to address this issue is to lock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

VI. CONCLUSIONS AND FUTURE WORK

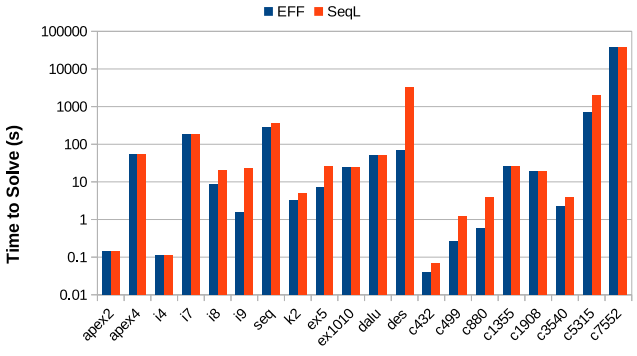
We have proposed SeqL*, that performs functional isolation and FI locking.* SeqL *hides a major fraction of the functionally correct keys, thus maximizing functional output corruption. We have shown both the theoretical and emprical improvements in the security of sequential locking. The results have shown 100% resilience to* ScanSAT *and furthermore* SeqL *has lower*



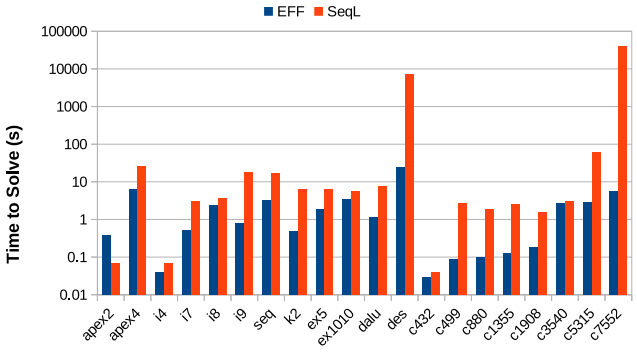
(a) RND [4]



(b) DAC'12 [10]



(c) ToC'13/xor [11]



(d) ToC'13/mux [11]

Fig. 6. Pipelined combinational benchmarks: key decryption time comparison between *SeqL* and *EFF*

area, timing and power overheads. Additionally, it takes time exponential in number of locked flip-flops, to launch removal attack on SeqL*, and is hence provably secure.*

REFERENCES

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

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

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

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

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

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

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

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

[9] *P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in* IEEE HOST*, 2015, pp. 137–143.*

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

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

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

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

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

[15] *M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: Sat attack resistant logic locking," in* IEEE International Symposium on Hardware Oriented Security and Trust*, May 2016, pp. 236–241.*

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

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

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

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

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

[21] *D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking,"* CoRR*, vol. abs/1811.12088, 2019. [Online]. Available: https://arxiv.org/abs/1811.12088*

[22] *R. Karmakar, S. Chattopadhyay, and R. Kapur, "Encrypt Flip-Flop: A novel logic encryption technique for sequential circuits,"* CoRR, *vol. abs/1801.04961, 2018. [Online]. Available: http://arxiv.org/abs/1801.04961*

[23] *T. I. B. Services. [Online]. Available: www.tsmc.com/english/dedicatedFoundry/services/integrated_backend.htm*

[24] *I. benchmarks. [Online]. Available: https://www.cerc.utexas.edu/itc99-benchmarks/bench.html*

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

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

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

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