TriLock: IC Protection with Tunable Corruptibility and Resilience to SAT and Removal Attacks

Yuke Zhang¹, Yinghua Hu¹, Pierluigi Nuzzo, and Peter A. Beerel

Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA
{yukezhang, yinghuahu, nuzzo, pabeerel}@usc.edu

Abstract—Sequential logic locking has been studied over the last decade as a method to protect sequential circuits from reverse engineering. However, most of the existing sequential logic locking techniques are threatened by increasingly more sophisticated SAT-based attacks, efficiently using input queries to a SAT solver to rule out incorrect keys, as well as removal attacks based on structural analysis. In this paper, we propose TriLock, a sequential logic locking method that simultaneously addresses these vulnerabilities. TriLock can achieve high, tunable functional corruptibility while still guaranteeing exponential queries to the SAT solver in a SAT-based attack. Further, it adopts a state re-encoding method to obscure the boundary between the original state registers and those inserted by the locking method, thus making it more difficult to detect and remove the locking-related components.

Index Terms—Sequential Logic Locking, SAT-Based Attacks, Hardware Security

I. INTRODUCTION

The decentralization of the integrated circuit (IC) supply chain over the past few decades has increasingly raised concerns about potential threats, such as intellectual property (IP) piracy and hardware Trojan insertion [1]. One of the most investigated IC protection schemes against these threats is logic encryption (or locking) [2]–[8], which adds programmability to the design at the gate or register-transfer level (RTL), so that the intended function is hidden from unauthorized users, and can only be accessed by a legal user by appropriately configuring the locked circuit.

Early logic locking methods have mostly focused on modifying the combinational portion of a circuit [3], [4]. On the other hand, sequential logic locking, the focus of this paper, usually involves creating new states in the finite state machine representing the original circuit and modifying its transitions [2], [6], [9]–[13]. The correct functionality is typically retrieved by either providing a key sequence, i.e., a dynamic sequence of key patterns, via the primary input ports during the first few clock cycles [2], [9] or by setting a set of key ports to fixed values throughout the circuit operation time [11]–[13]. Sequential locking shows the promise of significantly increasing the attack effort at reasonable cost by judiciously expanding a circuit’s state space. Yet, major threats to existing schemes have been posed by increasingly more sophisticated SAT-based attacks, efficiently using queries to a Boolean satisfiability (SAT) solver to rule out incorrect keys, as well as removal attacks that can exploit structural circuit signatures.

SAT-based attacks [6], [14], [15], leveraging circuit unrolling and model checking, have shown to be successful against the first versions of sequential locking [2], in that they can effectively exploit the early occurrence of output errors to dramatically decrease the number of SAT queries and accelerate the key search. This vulnerability has called for methods that can intentionally postpone the first occurrence of the output errors [9]–[11], [13]. However, SAT-based attacks can still be accelerated by leveraging functional corruptibility to help estimate the required circuit unrolling depth and further reduce the number of SAT queries [16]. Moreover, SAT- resilient methods tend to exhibit poor error rates, usually captured in terms of functional corruptibility, hence lack enough protection – a trade-off that is extensively documented in the context of combinational locking [17], [18]. Finally, the net boundary between the locking-related components and the rest of the circuit makes them vulnerable to removal attacks [19], possibly boosted by machine learning-assisted netlist analysis tools [20], [21]. A robust sequential locking scheme that can offer quantifiable protection and resilience to SAT-based and removal attacks is still elusive.

This paper proposes TriLock, an IC protection scheme that leverages the temporal dimension of sequential locking to break the well-known trade-off between SAT-attack resilience and functional corruptibility of combinational locking and simultaneously address all of the above challenges. Our contributions include:

• A cost-effective logic locking method that can exponentially increase the number of SAT queries required for a successful SAT-based attack.
• An error-generation mechanism that can strategically increase the output error rate to achieve a desired functional corruptibility without compromising SAT-attack resilience.
• A state re-encoding technique that can significantly blur the boundary between the original circuit and the logic added by the locking scheme.

To the best of our knowledge, TriLock is the first sequential locking technique that simultaneously tackles all the above security objectives. We demonstrate its effectiveness via security analysis and empirical validation on ISCAS’89 [22] and ITC’99 [23] benchmarks.

¹Yuke Zhang and Yinghua Hu contributed equally to this work. This work is based on research sponsored by the U.S. Government. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements of the U.S. Government.
II. BACKGROUND AND RELATED WORK

We discuss SAT-based and removal attacks as well as the methods that have been proposed to counteract them.

A. Preliminaries

In the following, we just use the term circuit to refer to a sequential circuit. For a circuit $C$, shown in Fig. 1(a) its $b$-unrolled version, denoted by $C_b$, is a combinational circuit, shown in Fig. 1(b) that represents the behavior of $C$ over the first $b$ clock cycles. We denote by $I$ and $O$ the sets of input and output ports of $C$, respectively, by $k$ and $i$ its key and input sequence, respectively, and by $\kappa$ the (cycle) length of $k$, i.e., the number of clock cycles required to provide $k$ to the circuit after reset. For a sequence $s$, $s_n$ denotes the sub-sequence of $s$ associated with the $n$-th unrolling (clock cycle) and $s_{n+m}$ the one associated with the range of unrollings from $n$ to $m$.

Let $C^e_b$ be the $b$-unrolled version of the original circuit and $C^\cap_b$, with a slight abuse of notation, the $(\kappa+b)$-unrolled version of the encrypted circuit. For simplicity, we refer to $C^e_b$ as the $b$-unrolled version of the encrypted circuit, by skipping the first $\kappa$ cycles used to input the key. Let the functions implemented by $C^\cap_b$ and $C^e_b$ be $f_b : \mathbb{B}^{|I|} \rightarrow \mathbb{B}^{|O|}$ and $f'_b : \mathbb{B}^{|I|} \times \mathbb{B}^{|I|} \rightarrow \mathbb{B}^{|O|}$, respectively. Then, the functional corruptibility (FC) of a $b$-unrolled version of an encrypted circuit is defined as [16]

$$ FC_b = \frac{1}{2^{(k+b)|I|}} \sum_{i \in \mathbb{B}^{|I|}} \sum_{k \in \mathbb{B}^{|I|}} \mathbb{I}(f_b(i) \neq f'_b(i,k)), $$

where $\mathbb{I}(\cdot)$ is the indicator function. $FC_b$ quantifies the proportion of errors over all input-key combinations for a $b$-unrolled encrypted circuit.

B. SAT-Based Attacks

The idea of formulating a SAT problem to prune out wrong keys was first adopted by the SAT attack Comb-SAT [24] to combinational logic locking. Comb-SAT assumes that the attacker has access to the netlist of the locked circuit as well as unlimited access to the correct input/output pairs from the original circuit. An iterative key elimination process is executed by searching for distinguishing input patterns (DIPs) via SAT solving. A DIP is an input pattern of the locked circuit for which there exist two different keys that lead to different outputs. When a DIP $d$ is found, it can effectively rule out a set of wrong keys $K_d$ that are detectable by $d$, expressed as

$$ K_d = \{k | f(d) \neq f'(d,k)\}, $$

where $f$ and $f'$ are the functions implemented by the original and the locked circuit, respectively. Until the correct key is obtained, more DIPs are iteratively found and used to further prune out the key search space. A trade-off exists between SAT-attack resilience, i.e., the number of DIPs required to find the correct key, and the FC of a locked circuit [17], [18]. The larger the number of errors induced by a wrong key, the higher the likelihood that the wrong key can be detected and eliminated by a DIP.

Comb-SAT cannot be directed applied to sequential circuits without scan access to their internal states. It can, however, be extended by relying on circuit unrolling to generate a $b$-unrolled version of the encrypted circuit, $C^\cap_b$, on which to apply Comb-SAT [6], [14], [15]. Once a key is found for $C^\cap_b$, model checking is performed to verify whether the key is also correct for $C^e_b$, beyond the first $b$ clock cycles. If this check fails, the above steps will be repeated for a larger $b$. Several sequential encryption methods [9]–[11], [13] boost SAT-attack resilience by increasing the minimum unrolling depth $b^*$ that is needed to rule out all the wrong keys. However, $b^*$ has been recently shown to be effectively predictable [16], thus making SAT-based attacks even more efficient.

C. Removal Attacks

Sequential logic encryption methods may be vulnerable to removal attacks based on structural analysis of the circuit netlist [12], [19]–[21]. Unwanted signatures may be detected in the state transition graph (STG) of the encrypted circuit, for example, when there is only one edge from the set of states added by the locking logic to the states in the original STG [2]. Graph analysis methods can then be applied to the STG to recognize the boundary between the two sets of states [19]. State-Deflection [10] adds several sink state clusters in the STG to trap illegal users. However, because a sink cluster does not have any outgoing edge, it can be easily identified by a strongly connected component (SCC) algorithm.

Several papers [20], [21] view the recognition of state registers as the first step for reverse-engineering finite state machines, and propose accurate tools for this task. After the state registers are recognized, the original registers must be separated from the additional registers associated with the encryption logic. In this paper, we assume that all the state registers of a circuit can be successfully recognized. The aim of TriLock is to make it more difficult to separate and remove the additional registers associated with the encryption.

III. TriLock

We first discuss the trade-off between SAT-attack resilience and FC and present in Section III-A a naive implementation of TriLock that achieves high resilience at the cost of low FC. We show how to overcome the trade-off in Section III-B.
allowing independent configuration of FC without compromising SAT-attack resilience. We finally detail our strategy to mitigate removal attacks in Section III-C. The encryption flow of TriLock is shown in Fig. 3.

### A. Trade-off Between SAT-Attack Resilience and FC

Combinatorial logic encryption techniques, such as SARLock [4] and Anti-SAT [5], adopt point functions to achieve exponential SAT-attack resilience, quantified in terms of the required number of DIPs ($n_{dip}$). In these methods, each DIP can only rule out a limited number of wrong keys at each iteration of the SAT attack. We apply a similar concept in TriLock to achieve exponential $n_{dip}$ in the key length $\kappa$. We assume that an attacker can efficiently estimate the minimum required unrolling depth $b^*$ [16] and perform COMB-SAT directly on $C_b^\kappa$, with $b \geq b^*$. We then focus on guaranteeing an exponential $n_{dip}$ for $C_b^\kappa$.

We define an error function, $E_b^N : \mathbb{B}^{[|I|]} \times \mathbb{B}^{[|I|]} \rightarrow \{1, 0\}$, for $C_b^\kappa$, as a function that takes as arguments a $|I|$-bit input sequence $i$ and a $|I|$-bit key sequence $k$, and returns 1 if and only if an error occurs at the output of $C_b^\kappa$, i.e., if and only if $f_b^N(k, i) \neq f_b^N(i)$. A naive error function that achieves exponential $n_{dip}$ can then be obtained by setting $b^* = \kappa$ and by implementing a point function, as done, for example, in SARLock [4]. We would therefore obtain

$$E_b^N(i, k) = \mathbb{I}[(k \neq k^*) \land (k = i^{1+\kappa})], \quad (3)$$

where $k^*$, the correct key sequence, is a fixed sequence of length $\kappa$. For an arbitrary wrong key $k^w$, there exists a set of input sequences $IS_k^w$ for which $E_b^N$ evaluates to 1, expressed as follows,

$$IS_k^w = \{i \in \mathbb{B}^{[|I|]} | k^w = i^{1+\kappa}\}, \quad (4)$$

that is, all the input sequences having $k^w$ as a prefix. Based on the mechanism of COMB-SAT, any input sequence in $IS_k^w$ can then be selected as a DIP to rule out the wrong key $k^w$. However, an input sequence in $IS_k^w$ cannot detect any other wrong key, that is,

$$\forall i \in IS_k^w, \forall k \in \mathbb{B}^{[|I|]} \setminus \{k^w\}, E_b^N(i, k) = 0. \quad (5)$$

Consequently, one DIP can only rule out one wrong key at a time and the $n_{dip}$ will equal the number of wrong keys, i.e.,

$$n_{dip} = 2^{\kappa|I|} - 1. \quad (6)$$

The effect of $E_b^N$ is pictorially represented by the colored error table in Fig. 3(a) for a 2-input circuit with $\kappa = b^* = b = 2$. The row and the column indexes correspond to the values of the input and the key sequences, respectively. If $E_b^N(i, k) = 1$, the corresponding square is red. By definition, $FC_b^\kappa$ can be computed as

$$FC_b^\kappa = \frac{(2^{\kappa|I|} - 1) \cdot 2^{(b-\kappa)|I|}}{2^{(\kappa+b)|I|}} \approx \frac{1}{2^{(b-\kappa)|I|}} \approx \frac{1}{n_{dip} + 1}. \quad (7)$$

which is, unsurprisingly, low for $E_b^N$, as low as 0.06 in the scenario of Fig. 3(a). In the following, we detail how this trade-off between $n_{dip}$ and $FC_b^\kappa$ can be computed.

### B. Circumventing the SAT-Attack Resilience vs. FC Trade-Off

For better clarity, we use $\kappa_s$ to denote the key cycle length and rewrite the error function in (3) as follows:

$$E_b^S(i, k) = \mathbb{I}[(k \neq k^* \lor (k_{1+\kappa_s} = i_{1+\kappa_s})]. \quad (8)$$

We can now increase FC without compromising the attack resilience achieved by $E_b^S$ by strategically redesigning the error function over a larger key cycle length $\kappa = \kappa_s + \kappa_f$, leading to an extended error table, as shown in Fig. 3(b) for a 2-input circuit with $\kappa_s = b_s = b = 2$ and $\kappa_f = 1$. The red squares represent the errors defined by $E_b^S$; we denote their number by $n_{dip}^S$, given by

$$n_{dip}^S = 2^{\kappa|I|} - 1. \quad (9)$$

Similarly to (4), for any wrong key with prefix $k_{1+\kappa_s}^w$, there exists a set of input sequences $IS_{k^w} = \{i \in \mathbb{B}^{[|I|]} | k_{1+\kappa_s}^w = i_{1+\kappa_s}\}$ that can be used as DIPs to eliminate only wrong keys with the same prefix $k_{1+\kappa_s}^w$. There are, in total, $2^{\kappa|I|}$ possible values for the prefix $k_{1+\kappa_s}^w$ of a wrong key, so the SAT-attack resilience corresponding to $E_b^S$ is

$$n_{dip}^S = 2^{\kappa|I|}. \quad (10)$$

Besides the errors defined by $E_b^S$, we look for a set of additional input-key pairs in $C_b^\kappa$ such that, if an error is added at each pair, it will not decrease the SAT-attack resilience achieved by $E_b^S$. For a fixed sequence $k^{**}$ of length $\kappa_f$, specified by the designer and such that $k^{**} \neq k_{(\kappa_s-\kappa_f)+\kappa_s}$, one such set can be defined as follows:

$$P_b^{k^{**}} = \{(i, k) | (k_{(\kappa_s-\kappa_f)+\kappa_s} \neq k^{**}) \land (k \neq k^*)\}, \quad (11)$$

where $k^*$ is the correct key sequence of length $\kappa$. The blue squares in Fig. 3(b) pictorially represent $P_b^{k^{**}}$, when $k^* = 100101$ and $k^{**} = 11$. The following result states the property of $P_b^{k^{**}}$. 

![Fig. 3. Error tables resulting from the application of (a) $E_b^N$ as in (1), with $|I| = \kappa = b^* = b = 2$ and (b) $E_b^S$ as in (10), with $|I| = \kappa = b = 2$ and $\kappa_f = 1$.](image)

![Fig. 4. Relations between $n_{dip}^S$ and $FC_b^\kappa$ of a 4-input circuit implemented by $E_b^S$ and $E_b^S^\kappa_f$ with $\kappa_f = 1$.](image)
Theorem 1. Given two sequences $k^{**}$ and $k^*$ of length $\kappa_f$ and $\kappa$, respectively, with $\kappa_f < \kappa$, let the unrolled version of the encrypted circuit $C^w_b$ implement the error function $E_b^S$ in (8), with $\kappa_w = \kappa - \kappa_f$. We assume that errors are inserted at all input-key pairs in $P_b^{k^*,k^{**}}$ defined as in (11), i.e., $f_b(i,k) \neq f_b(i,k)$, $\forall (i,k) \in P_b^{k^*,k^{**}}$. Then, COMB-SAT on $C^w_b$ will require at least $2^{\kappa_w|I|}$ DIPs.

Proof. Given a wrong key $k^w$ with $k^{**}$ as a suffix, i.e., such that $(k^w \neq k^*) \land (k^w |_{(\kappa_f-\kappa)} = k^{**})$, by the definition of $P_b^{k^*,k^{**}}$, we have that $(i,k^w) \notin P_b$, $\forall i \in B^{|I|}$. Let $i^w$ be the DIP capable of detecting $k^w$. Since $i^w$ cannot be in $P_b^{k^*,k^{**}}$, then it must satisfy $E_b^S(i^w,k^w) = 1$. Let us now assume that $k^w$ is another wrong key with $k^{**}$ as a suffix. In total, there are $2^{\kappa_w|I|}$ wrong keys with a suffix of $k^{**}$. Therefore, the DIP that allows ruling out $k^w$ cannot exclude any other wrong key having $k^{**}$ as a suffix. In total, there are $2^{\kappa_w|I|}$ wrong keys with a suffix of $k^{**}$. Therefore, $2^{\kappa_w|I|}$ DIPs are at least required to exclude those wrong keys.

Algorithm 1 State register selection

Input: $C^w$ and $S$.
Output: $R$.
1: $RCG = create\_graph(C^w)$; $R = []$; $count = 0$
2: $E, O, M = run\_scce(RCG)$
3: while $(E \neq \emptyset \text{ or } O \neq \emptyset \text{ and } count < S)$ do
4:  if $(E \neq \emptyset \text{ and } O \neq \emptyset)$ then
5:     $SCC_1, SCC_2 = get\_largest\_scce(E, O)$
6:     else
7:         temp = get\_non\_empty\_set($E, O$)
8:         $SCC_1, SCC_2 = get\_largest\_scce(temp, M)$
9:     end if
10:    $r_1, r_2 = get\_max\_edge\_node(SCC_1, SCC_2)$
11:    $R.append([r_1, r_2])$; $count = count + 1$
12: $RCG = update\_graph(RCG, [r_1, r_2])$
13: $E, O, M = run\_scce(RCG)$
14: end while
15: return $R$

C. Enhancing Removal Attack Resilience: State Re-encoding

As shown in Fig. 2(a), we can distinguish the original state registers of an encrypted circuit (in blue) from the extra state registers added by the encryption (in green). Identifying the type of registers is an essential step toward removal attacks. Specifically, an attacker can leverage the SCC algorithm in a register connection graph (RCG), where a register is represented by a node and the existence of a path between two registers is denoted by a directed edge between the corresponding nodes. The output of the SCC algorithm on an RCG is one or more clusters of nodes, called SCCs. For any two nodes in the same SCC, they are reachable from each other. We denote by O-SCC, E-SCC, and M-SCC, an SCC containing only the original registers, only the extra registers, and a mix of the two types of registers, respectively.

When no SCC in an RCG is an M-SCC, the identification of the set of original or extra registers is expected to be easy, as each SCC is already a congregation of either original or extra registers. In the best case, an attacker could expect only two SCCs, an O-SCC and an E-SCC, with all the original registers and all the extra registers, respectively, as the algorithm output. Such a successful clustering of the registers would be due to the insufficient connections between original and extra registers. In contrast, if the connections are dense, one or more M-SCCs will exist and it will be harder to classify the type of registers in those M-SCCs. We then propose a state re-encoding method, implemented on the encrypted sequential circuit, that intentionally creates new edges between O-SCCs and E-SCCs, resulting in more registers being clustered in one or several M-SCCs. As shown in Fig. 2(b), the state re-encoding method selects a configurable number of registers, and inserts state encoders and decoders after adding the error generator and error handlers. We introduce below the register selection procedure and the encoder/decoder mechanism adopted in state re-encoding.

State Register Selection. We adopt a greedy method to iteratively select and encode pairs of original and extra reg-
Algorithm 1 shows the pair selection process, which takes as inputs an encrypted netlist \( C^e \) and the desired number of register pairs \( S \), and returns a list of register pairs \( R \) as output. After creating the RCG from \( C^e \) (line 1) and running the SCC algorithm (line 2), three sets, i.e., \( E \), \( O \), and \( M \), are generated that contain all the E-SCCs, O-SCCs, and M-SCCs, respectively. To maximize the impact of state re-encoding for a single pair of original and extra registers, we first identify the largest O-SCC and E-SCC as \( SCC_1 \) and \( SCC_2 \), respectively (line 5). In case there does not exist an E-SCC or O-SCC, we choose the largest M-SCC as the replacement \((\text{line 7-8})\). In \( SCC_1 \) and \( SCC_2 \), we then select the nodes connected by the largest number of edges, denoted as \( r_1 \) and \( r_2 \), respectively (line 10). We record \((r_1, r_2)\) in \( R \) (line 11) and update the RCG (line 12-13) as a result of re-encoding \((r_1, r_2)\). The above register pair selection process is iterated until theodesign-specific number of pairs is reached or no E-SCC and O-SCC exist.

**Encoder/Decoder Mechanism.** For each pair of registers \((r_1, r_2)\) in \( R \), a state encoder and a state decoder are inserted between the combinational logic and the two registers, \( r_1 \) and \( r_2 \), as shown in Fig. 5(a) to merge the SCC containing \( r_1 \) (\( SCC_1 \)) and the one containing \( r_2 \) (\( SCC_2 \)). We denote by \( enc(\cdot) \) and \( dec(\cdot) \) the functions of the encoder and the decoder, respectively. Between the encoder and the decoder, \( r_1 \) and \( r_2 \) are replaced by a set of encoded state registers \( r_{zi} \), where \( 1 \leq i \leq m \). We denote by \( s_1 \) (\( s'_1 \)) the net connecting from (to) the combinational logic to (from) \( r_1 \). Similar notations are used for \( r_2 \).

To prevent the encoder/decoder structure from affecting the circuit function, a fixed-point condition \( dec(enc(a)) = a \) should hold for any 2-bit sequence \( a \). Moreover, state re-encoding should achieve successful merging of the two SCCs into one M-SCC, which requires the existence of the following looped signal propagation path (abbreviated as a path):

\[
\exists x, y \in [1, m] : s_1 \rightarrow r_{ex} \rightarrow s'_2 \rightarrow s_2 \rightarrow r_{ey} \rightarrow s'_1 \rightarrow s_1. 
\tag{17}
\]

When such a looped path exists, any register \( r_0 \) in \( SCC_1 \) can connect to any register \( r_0 \) in \( SCC_2 \) via the path: \( r_0 \rightarrow s_1 \rightarrow r_{ex} \rightarrow s'_2 \rightarrow r_{in} \), as shown in Fig. 5(b). Similarly, \( r_b \) can also reach \( r_a \) via the path: \( r_b \rightarrow s_2 \rightarrow r_{ey} \rightarrow s'_1 \rightarrow r_0 \). On the RCG, the above paths construct a bidirectional edge between \( r_a \) and \( r_b \), which merges \( SCC_1 \) and \( SCC_2 \) into an M-SCC.

In this paper, we implement the encoder with two arithmetic operations, namely, \( e_1 = s_1 + s_2 \) and \( e_2 = s_1 - s_2 \), where \( e_1 \) and \( e_2 \) are the encoded states. The decoder also executes two arithmetic operations, \( s'_1 = \frac{1}{2}(e'_1 + e'_2) \) and \( s'_2 = \frac{1}{2}(e'_1 - e'_2) \), which satisfies the fixed-point condition while creating a looped path as in (17). The associated two SCCs are, thus, merged into an M-SCC. To mitigate the structural signature produced by repeatedly implementing the same encoder/decoder, various \( enc(\cdot) \) and \( dec(\cdot) \) can be applied to different register pairs, which can be subject of future work.

### IV. Experimental Results

We implement the encryption flow of TriLock in Python, using Synopsys Design Compiler and a 45nm Nangate Open Cell Library [25] as the synthesis tool and the target library, respectively. FC is simulated with 800 random inputs and keys using Synopsys VCS, while the SAT-attack resilience is evaluated via an implementation of a state-of-the-art SAT-based attack [16], which can effectively predict the minimum required unrolling depth \( b^* \). In the case of TriLock, \( b^* = \kappa_s \).

We select ten benchmark circuits from ISCAS’89 [22] and ITC’99 [23], as shown in Table I. All experiments are executed on an Intel(R) Xeon(R) E5-2450 2.5-GHz CPU with 126-GB memory.

**SAT-Attack Resilience.** Table I shows \( n_{dip} \) and the runtime resulting from applying the attack on the selected benchmark circuits when \( \kappa_s \) ranges from 1 to 3, and \( \kappa_f \), \( \alpha \) and \( S \) are fixed to 1, 0.6, and 10, respectively. With a two-day time-out threshold, four experiments terminated successfully. The results show that the achieved SAT-attack resilience is consistent with (10).

For the rest of the experiments, denoted in blue, we show \( n_{dip} \) as computed by (10) and extrapolate the runtime by conservatively assuming a constant ratio between the runtime and \( n_{dip} \) that can be acquired from the finished experiments. According to Table I 76.6\% of the attack experiments are expected to require more than one year to finish.

**Functional Corruptibility.** Fig. 7 reports the simulated FC\(_b\) for different \( \alpha \) and \( \kappa_f \). We set \( \kappa_s = 4 \) to achieve high SAT-attack resilience, since \( \kappa_s = 3 \) can already achieve high resilience for most circuits in Table I. For each locking configuration, we plot the average of the simulated FC\(_b\) for \( b \) ranging from \( \kappa_s \) to \( \kappa_s + 5 \). Our results show that FC\(_b\) is close to its estimate in (15), with an absolute error within \( \pm0.05 \), which illustrates TriLock’s ability to configure FC with high SAT-attack resilience.

**Removal Attack Resilience.** For each benchmark circuit, we perform state re-encoding with \( S = 10 \) and \( S = 30 \). In addition, we generate a reference design with no state re-encoding, i.e., \( S = 0 \). Table II shows the results of the SCC algorithm. In addition to the number of different types of
We synthesize the locked netlists with the original and extra registers are densely connected. P eliminates the existence of O-SCCs or E-SCCs for most cases on-chip scenarios, it is possible to implement 40% increase in the area, delay, and power, respectively, incurred. of area, delay, and power (ADP) is computed as percentage average, the numbers of O-SCCs and E-SCCs are reduced by 71%. \( \kappa \) achieves reasonably high SAT-attack resilience with 50%, where the overhead is less than 40%. However, as shown in Table II, these circuits can already achieve reasonably high SAT-attack resilience with \( \kappa = 2 \), where the overhead is less than 40%. In system-on-chip scenarios, it is possible to implement TriLock only on the sensitive portions of the design, making the overhead even smaller.

V. CONCLUSIONS

In this paper, we propose a cost-effective sequential logic locking technique, TriLock, to achieve both high SAT-attack resilience and high functional corruptibility, which circumvents, for the first time, the trade-off between the two security concerns that exists in combinational locking. We also present a state re-encoding technique that can significantly improve the removal attack resilience of TriLock and, potentially, other sequential locking techniques. Future work includes investigating other attack vectors [26], e.g., signature analysis on the STG, to further improve the robustness of TriLock.

REFERENCES

[1] M. Tehranipoor et al., “A survey of hardware trojan taxonomy and detection,” IEEE Design & Test of Comput., vol. 27, no. 1, pp. 10–25, 2010.
[2] R. S. Chakraborty et al., “HARPION: An oblivious-based SoC design methodology for hardware protection,” IEEE Trans. Comput.-Aided Design of Integrated Circ. and Syst., vol. 28, no. 10, pp. 1493–1502, 2009.
[3] J. Rajendran et al., “Fault analysis-based logic encryption,” IEEE Trans. Computers, vol. 64, no. 2, pp. 410–424, 2015.
[4] M. Yamin et al., “SARLock: SAT attack resistant logic locking,” in IEEE Int. Symp.Hardware Oriented Secur. and Trust (HOST), pp. 236–241, 2016.
[5] Y. Xu et al., “Anti-SAT: Mitigating SAT attack on logic locking,” in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 2, pp. 199–207, 2018.
[6] Y. Hu et al., “SANSCrypt: Spontaneous-authentication-based sequential logic encryption,” in VLSI-SoCDesign Trends (A. Calimera et al., eds.), (Cham), pp. 255–278, Springer International Publishing, 2021.
[7] C. Pilato, A. B. Chorduly, D. Sciuto, S. Garg, and R. Katti, “ASSURE: RTL locking against an untrusted foundry,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021.
[8] S. D. Choudhury et al., “Enhancing SAT-attack resilience and cost-effectiveness of reconfigurable-logic-based circuit obfuscation,” in Int. Symp. Circuits and Systems (ISCAS), pp. 1–5, IEEE, 2021.
[9] A. Desai et al., “Interlocking obfuscation for anti-tamper hardware,” in Proc. Cyber Secur. and Info. Intrust. Research Workshop, pp. 1–4, 2013.
[10] J. Dolle et al., “Novel dynamic state-deletion method for gate-level design obfuscation,” IEEE Trans. Comput.-Aided Design of Integrated Circ. and Syst., vol. 37, no. 2, pp. 273–285, 2018.
[11] Y. Kasarabada et al., “Deep state encryption for sequential logic circuits,” in IEEE Comput. Society Annual Symp. VLSI, pp. 338–341, 2019.
[12] R. S. Choudhury et al., “DFSSD: Deep faults and shallow state duality, a provably strong obfuscation solution for circuits with restricted access to scan chain,” in IEEE VLSI Test Symp., pp. 1–6, 2020.
[13] A. Rezaei et al., “Sequential logic encryption against model checking attack,” in Design, Auto. and Test in Europe Conf. and Exh., pp. 178–181, 2021.
[14] M. E. Massoud et al., “Reverse engineering camouflaged sequential circuits without scan access,” in 2017 IEEE/ACM Int. Conf. on Comput.-Aided Design, pp. 34–40, 2017.
[15] K. Shams et al., “RC2: Key-conditioning for fast sequential circuit de-obfuscation,” in Design, Auto. and Test in Europe Conf. and Exh., pp. 534–539, 2019.
[16] Y. Hu et al., “Fun-SAT: Functional compatibility-guided SAT-based attack on sequential logic encryption,” arXiv preprint 2108.04912, 2021.
[17] Y. Hu et al., “Security-driven metrics and models for efficient evaluation of logic encryption schemes,” in ACM-IEEE MEMOCODE, pp. 1–5, 2019.
[18] Y. Hu et al., “Risk-aware cost-effective design methodology for integrated circuit locking,” in Design, Auto. and Test in Europe Conf. and Exhibition (DATE), pp. 1182–1185, IEEE, 2021.
[19] T. Maede et al., “Revise sequential logic obfuscation: Attacks and defenses,” in IEEE Int. Symp. Circuits and Systems, pp. 1–4, 2017.
[20] T. Maede et al., “Gate-level netlist reverse engineering for hardware security: Control logic register identification,” in IEEE Int. Symp. Circuits and Systems (ISCAS), pp. 1334–1337, 2016.
[21] J. Geist et al., “RELIC-FUN: Logic identification through functional signal comparisons,” in Proc. Design Auto. Conf., pp. 1–6, 2020.
[22] F. Bretz et al., “Combinational profiles of sequential benchmark circuits,” in IEEE Int. Symp. Circ. and Syst., pp. 1928–1934, 1989.
[23] F. Corono et al., “RI-level ITC’99 benchmarks and first ATPG results,” in Design & Test of Computers, vol. 17, no. 5, pp. 44–53, 2000.
[24] P. Subramanyan et al., “Evaluating the security of logic encryption algorithms,” in IEEE Int. Symp. Hardware Oriented Secur. and Trust, pp. 137–143, 2015.
[25] Silvaco, “45nm open cell library,” 2019.
[26] S. Engels et al., “The end of logic locking? A critical view on the security of logic locking,” Cryptology ePrint Archive, Report 2019/796, 2019.