SANSCrypt: Sporadic-Authentication-Based Sequential Logic Encryption

Yinghua Hu, Kaixin Yang, Shahin Nazarian, Pierluigi Nuzzo

SANSCrypt: Sporadic-Authentication-Based Sequential Logic Encryption

Yinghua Hu, Kaixin Yang, Shahin Nazarian, and Pierluigi Nuzzo

University of Southern California, Los Angeles CA, 90089, USA
{yinghuah,kaixinya,shahin.nazarian,nuzzo}@usc.edu

Abstract. Sequential logic encryption is a countermeasure against reverse engineering of sequential circuits based on modifying the original finite state machine of the circuit such that the circuit enters a wrong state upon being reset. A user must apply a certain sequence of input patterns, i.e., a key sequence, for the circuit to transition to the correct state. The circuit then remains functional unless it is powered off or reset again. Most sequential encryption methods require the correct key to be applied only once. In this paper, we propose a novel Sporadic-Authentication-Based Sequential Logic Encryption method (SANSCrypt) that circumvents the potential vulnerability associated with a single-authentication mechanism. SANSCrypt adopts a new temporal dimension to logic encryption, by requiring the user to sporadically perform multiple authentications according to a protocol based on pseudo-random number generation. We provide implementation details of SANSCrypt and present a design that is amenable to time-sensitive applications. In SANSCrypt, the authentication task does not significantly disrupt the normal circuit operation, as it can be interrupted or postponed upon request from a high-priority task with minimal impact on the overall performance. Analysis and validation results on a set of benchmark circuits show that SANSCrypt offers a substantial output corruptibility if the key sequences are applied incorrectly. Moreover, it exhibits exponential resilience to existing attacks, including SAT-based attacks, while maintaining a reasonably low overhead.

Keywords: Hardware Security, Sequential Encryption, Sporadic Authentication

1 Introduction

The ever-increasing costs for the design and manufacturing of modern VLSI systems have led to a global supply chain, where several important steps, such as verification, fabrication, testing, and packaging, are outsourced to third-party companies. As proprietary design information and intellectual property (IP) blocks inevitably get to the supply chain, an untrusted third party may gain access to a sufficient amount of critical design information to potentially reverse engineer the design and massively reproduce it for illegal profit. Another possible consequence of reverse engineering is Hardware Trojan (HT) insertion, which

can either disrupt the normal circuit operation [1] or provide the attacker with access to critical data or software running on the chip [2]. Both types of HTs can be destructive for safety-critical applications, such as autonomous driving cars and implantable medical devices.

Countermeasures for reverse engineering, such as logic encryption [3–6], integrated circuit (IC) camouflaging [7], split manufacturing [8], and watermarking [9] have been developed over the past decades to either increase the hardness of IC reverse engineering or embed unique proprietary signatures on the IC. Among these, logic encryption has received significant attention as a promising, low-overhead countermeasure. Logic encryption achieves IC protection by properly modifying the original circuit such that a user can only access the correct function after configuring the circuit with a correct key pattern. Otherwise, the circuit function remains hidden, and the output different from the correct one.

Various logic encryption techniques [3–6] and potential attacks [10–12] have appeared in the literature, as well as methods to systematically evaluate them [13, 14]. A category of techniques [3–5], referred to as *combinational encryption*, is designed to modify and protect combinational circuits or the combinational logic portions of sequential circuits. When the circuit scan chains are accessible to the attackers, one of the most successful attacks against combinational encryption is the Boolean satisfiability (SAT)-based attack [10]. Even if the scan chains are not accessible, e.g., due to scan chain encryption and obfuscation [15–17], a variant of SAT-based attacks [18,19] can still succeed, at a higher cost, by leveraging methods from bounded model checking to unroll the sequential circuit. Another possible vulnerability of combinational encryption methods stems from the correlation between the circuit structure and the correct key, as recently exposed by an increasing number of attacks [20, 21] and theoretical analyses [22]. On the other hand, *sequential logic encryption* [6, 23, 24] targets the state transitions of the original finite state machine (FSM). Sequential encryption methods typically introduce additional states and transitions in the original FSM, such that the circuit enters the *encrypted mode* upon being reset, exhibiting an incorrect function. A user must apply a certain sequence of input patterns, i.e., a key sequence, for the circuit to transition to the correct initial state and enter the *functional mode*. Then, the circuit remains functional unless it is powered off or reset again.

Recently, a set of attacks have been reported against sequential encryption schemes, aiming to retrieve the correct key sequence or the correct circuit function. Similarly to the aforementioned SAT-based attacks [18,19] to combinational encryption, sequential encryption can also be attacked via an approach based on circuit unrolling and bounded model checking [25]. Another attack based on automatic test pattern generation (ATPG) [26] uses concepts from excitation and propagation of stuck-at faults to search the key sequence among the test patterns generated by ATPG. The ATPG-based attack assumes that most stuck-at faults can only be triggered and detected in the functional mode. Therefore, the correct authentication key sequence must appear in most of the test patterns generated by ATPG tools. Furthermore, when the attackers have some knowl-

edge of the topology of the encrypted FSM, then they can extract and analyze the state transition graph and bypass the encrypted mode [25]. Overall, the continuous advances in FSM extraction and analysis tools tend to challenge any of the existing sequential encryption schemes and call for approaches that can significantly increase their robustness.

This paper presents a novel Sporadic-Authentication-based Sequential Logic Encryption scheme (SANSCrypt), which raises the attack difficulty via a *multi-authentication protocol*, whose decryption relies on *retrieving a set of correct key sequences as well as the time at which each sequence should be applied*. Our contributions can be summarized as follows:

- A robust, multi-authentication-based sequential logic encryption method that for the first time, to the best of our knowledge, systematically incorporates the robustness of multi-factor authentication (MFA) [27] in the context of hardware encryption.
- An architecture for sporadic re-authentication where key sequences must be applied at multiple times, determined by a random number generator, to access the correct circuit functionality.
- A design of the multi-authentication protocol that is suitable for time-sensitive applications, as it ensures that the real-time execution of time-critical and safety-critical tasks is not disrupted.
- Security analysis and empirical validation of SANSCrypt on a set of ISCAS'89 benchmark circuits [28], showing exponential resilience against existing attacks, including SAT-based attacks, and reasonably low overhead.

Analysis and validation results show that SANSCrypt can significantly enhance the resilience of sequential logic encryption under different attack assumptions. A preliminary version of the results of this paper appeared in our previous publication [29], where we first introduced SANSCrypt. In this paper, we present an improved architecture and protocol design that are specifically amenable to time-sensitive applications, by allowing the authentication task to be interrupted or postponed upon request from higher-priority tasks. Moreover, we extend our analysis of the brute-force attack resilience to account for the attack difficulty brought by the timing uncertainty about when to apply the correct key sequences. Finally, we offer an extensive validation of the proposed construction, showing its ability to protect time-sensitive applications without affecting the execution of time-critical tasks.

The rest of the paper is organized as follows. We provide an overview of existing sequential logic encryption methods and related attacks in Section 2. In Section 3, we present a multi-authentication protocol applicable to sequential logic encryption and introduce the basic design and implementation details of SANSCrypt. We then describe an enhanced design that is compatible with time-sensitive applications. The security level of SANSCrypt is analyzed in Section 4, while Section 5 reports the results from functional testing and the overhead after synthesis. Conclusions are drawn in Section 6.

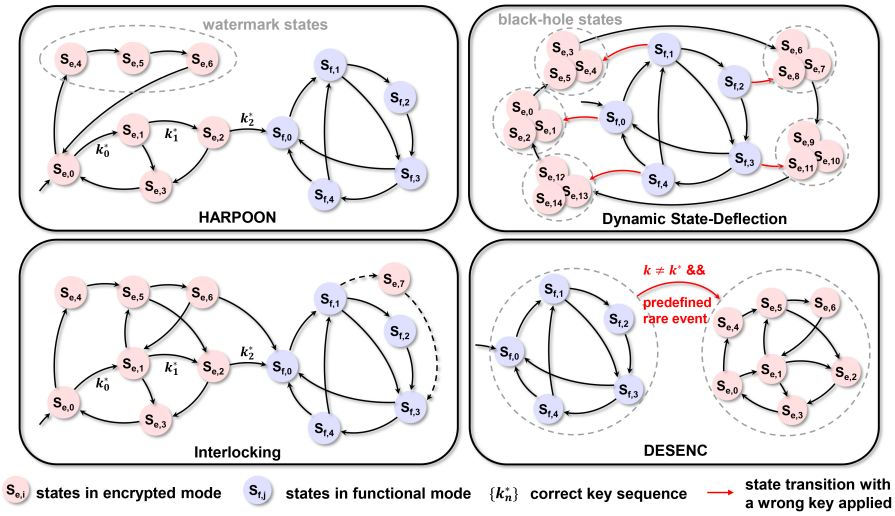


Fig. 1. State transition diagrams for different sequential encryption techniques.

2 Overview of Sequential Logic Encryption

A first sequential logic encryption method based on encrypting the finite state machine (FSM) of a circuit is *HARPOON* [6]. After encryption with HARPOON, the resulting FSM exhibits two main modes of operation, namely, an encrypted mode and a functional mode, as shown in Fig. 1 (top left). When powered on, the circuit starts in the encrypted mode and exhibits incorrect functionality. The user must apply an appropriate sequence of input patterns during the first few clock cycles to enter the functional mode, in which the correct functionality is recovered. To claim ownership of the circuit, HARPOON also creates a set of watermark states in the encrypted mode that can be entered only when another unique sequence of input patterns, known to the circuit designer, is applied. However, due to the simple mechanism of HARPOON, there is only one transition connecting the encrypted mode portion to the functional mode portion of the state transition diagram (STG) of the FSM. This distinguishable feature may help attackers locate and bypass the encrypted mode by FSM extraction and analysis methods [25].

Several tools [30–32] have been recently developed to facilitate FSM extraction by identifying the state registers from the circuit netlist. The increasing accuracy and efficiency of these methods call for encryption techniques that are more robust in the way they manipulate and obfuscate the STG of the circuit.

Interlocking [23] improves HARPOON by modifying the circuit FSM such that multiple transitions are available between the states of the encrypted mode FSM and the ones of the functional mode FSM, as shown in Fig. 1 (bottom left), making it harder for the attacker to detect the boundary between the two modes. However, in both HARPOON and Interlocking, once the circuit enters the func-

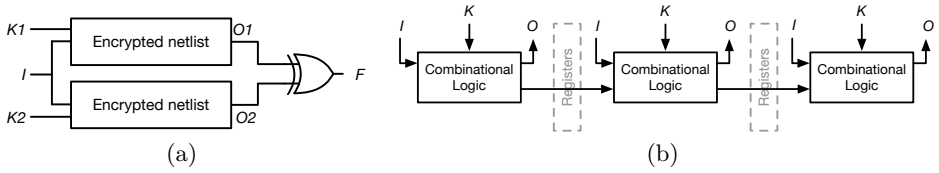


Fig. 2. (a) A miter circuit and (b) an unrolled circuit that represents the behavior of the sequential circuit for the first 3 cycles.

tional mode, it remains there unless it is powered off or reset. Moreover, because the correct circuit function can only be accessed when the correct key sequence is applied, attacks based on Automatic Test Pattern Generation (ATPG) [26] can be successfully mounted most of the times. ATPG-based attacks are based on the assumption that many stuck-at faults can only be triggered and detected when the circuit is in the functional mode. Therefore, the correct key sequence can be efficiently retrieved by analyzing common sub-sequences in the test patterns generated by ATPG tools.

Dynamic State Deflection [33] adds another level of protection by requiring an additional key input verification step in the functional mode. If the additional key input is incorrect, the FSM transitions to a black-hole state cluster which can no longer be left, as shown in Fig. 1 (top right). However, since the correct value for the extra key input is fixed over time, the scheme becomes more vulnerable to SAT-based attacks [18, 19].

The original SAT-based attack [10] has proven to be powerful on combinational logic encryption [3–5] when the circuit scan chains are accessible to attackers. The attack can efficiently prune out wrong keys by iteratively solving a set of SAT problems. At the first iteration, a miter circuit, consisting of two copies of the encrypted circuit, is assembled as shown in Fig. 2(a). The miter circuit is used to generate a SAT instance that is solved to search for a *distinguishing input pattern* (DIP). A DIP is an input pattern i, such that there exist at least two different key patterns, k_1 and k_2, leading to different outputs for the encrypted circuits, i.e., $F = 1$ in Fig. 2(a). Once a DIP is found, the attack queries an oracle, i.e., a functional circuit that is assumed to be available, to find the correct output for this DIP. The DIP and the correct output provide additional constraints for the SAT instance at the next iteration, which contributes to eliminating a group of wrong keys that do not result in the correct output when applying the DIP to the encrypted circuit. When no new DIPs are found, the SAT-based attack terminates, indicating that the remaining keys can all be used as correct keys.

When the scan chains are not available, a combinational miter circuit cannot be directly formed and SAT-based attacks [18, 19] leverage methods from bounded model checking [34] to "unroll" the logic loops in the sequential circuit and obtain a combinational circuit that represents the behavior of the original circuit over a time horizon, as pictorially shown in Fig. 2(b). The miter circuit is

then built out of the unrolled circuit to execute the SAT-based attack. However, to successfully terminate the attack, additional steps of model checking must be taken to ensure that the candidate keys are not only correct up to the current horizon but also for the original circuit. If this is not the case, the SAT-based attack must be repeated on unrolled circuit versions for increasingly longer time horizons to prune out wrong keys that were not detectable over shorter horizons. As suggested in the literature [25], a similar technique based on circuit unrolling can be used to attack sequential logic encryption methods. However, a detailed evaluation of these attacks on sequential encryption has been elusive.

While most of the encryption techniques mentioned above corrupt the circuit function immediately after reset unless the correct key sequence is applied, *DES-ENC* [24], shown in Fig. 1 (bottom right), determines the cycle for transitioning to the encrypted mode by counting the number of occurrences of a user-defined rare event in the circuit, unless the correct key sequence is applied. After the number of occurrences reaches a given threshold, the circuit enters the encrypted mode. This scheme is more resilient to sequential SAT-based attacks [35] because it requires unrolling the circuit FSM a large number of times to find the key. However, the initial transparency window may still expose critical portions of the circuit functionality.

3 Multi-Authentication-Based Sequential Encryption

We introduce the design and implementation details for SANSCrypt, starting with the underlying threat model.

3.1 Threat Model

SANSCrypt assumes a threat model that is consistent with the previous literature on sequential logic encryption [6, 19, 25]. The goal of the attack is to access the correct circuit functionality, by either finding the correct key sequence or reconstructing the correct circuit function. To achieve this goal, the attacker can leverage one or more of the following resources: (i) the encrypted netlist; (ii) a working circuit providing correct input-output pairs; (iii) knowledge of the encryption technique. In addition, we assume that the attacker has no access to the scan chain and cannot directly observe or change the state of the circuit.

3.2 Authentication Protocol

As shown in Fig. 3(a), existing sequential logic encryption techniques are mostly based on a single-authentication protocol, requiring users to be authenticated only once before accessing the correct circuit function. After the authentication, the circuit remains functional unless it is powered off or reset. To attack the circuit, therefore, it is sufficient to discover the correct key sequence that must be applied to the encrypted circuit upon reset.

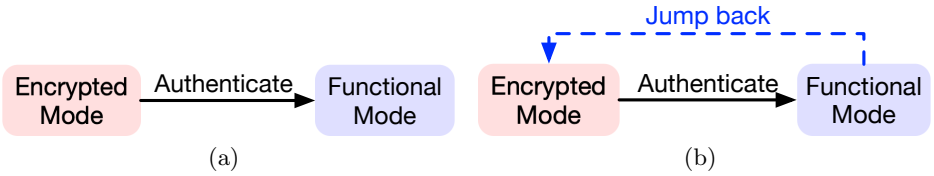


Fig. 3. Conventional (a) and proposed (b) authentication protocols for logic encryption.

We adopt, instead, the authentication protocol in Fig. 3(b), where the circuit can "jump" back to the encrypted mode from the functional mode. Once the back-jumping occurs, another round of authentication is required to resume the normal operation. The back-jumping can be triggered multiple times and involves a different key sequence for each round of re-authentication. The hardness of attacking this protocol stems from both the increased number of the key sequences to be applied and the uncertainty on the time at which each sequence should be applied. A new temporal dimension adds to the difficulty of the decryption procedure, which poses a significantly higher threshold to the attackers.

3.3 Overview of the Encryption Scheme

SANSCrypt is a sequential logic encryption scheme which supports random back-jumping, as represented in Fig. 4. When the circuit is powered or reset, the circuit falls into the reset state $E0$ of the encrypted mode. Like other sequential logic encryption schemes, the user must apply at startup the correct key sequence at the primary input ports for the circuit to transition to the initial (or reset) state $N0$ of the functional mode.

Once the circuit enters the functional mode, it can deliberately, but randomly, jump back, as denoted by the blue edges in Fig. 4, to a state s_{bj} in the encrypted mode, called *back-jumping state*, after a designated number of clock cycles t_{bj}, called *back-jumping period*. The user then needs to apply another key sequence to return to the state right before the back-jumping operation and resume normal operations, as shown by the red arrows in Fig. 4. Both the back-jumping state s_{bj} and the back-jumping period t_{bj} are determined by a pseudo-random number generator (PRNG) embedded in the circuit. Therefore, when and where the back-jumping operation happens is unpredictable unless the attacker is able to break the PRNG given the resources described in Section 3.1. An in-package key management circuit will be in charge of automatically applying the key sequences from a tamper-proof memory at the right time, as computed from a hard-coded replica of the PRNG. The schematic of SANSCrypt is shown in Fig. 5 and consists of two additional blocks, that is, a back-jumping module and an encryption finite state machine (ENC-FSM), besides the original circuit. We discuss each of these blocks in the following subsections.

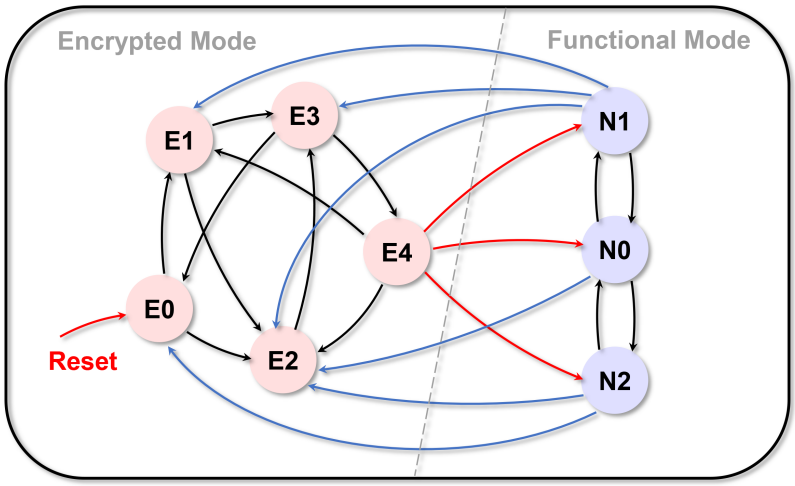


Fig. 4. State transition diagram of SANSCrypt.

3.4 Back-Jumping Module

The back-jumping module consists of an n-bit *PRNG*, an n-bit *Counter*, and a *Back-Jumping Finite State Machine* (BJ-FSM). BJ-FSM continually checks the output from the PRNG and the counter, and determines the back-jumping operations, as summarized by the flowchart in Fig. 6. Upon circuit reset, BJ-FSM keeps checking the authentication status. Once the authentication is successful and the circuit enters the functional mode, BJ-FSM samples the current PRNG output and stores this value as the back-jumping period t_{bj}. At the same time, the counter is set to zero.

The counter increments its output at each clock cycle until it reaches t_{bj}, when BJ-FSM samples again the PRNG output r. By taking the PRNG outputs at different clock cycles, r and t_{bj} are generally not the same. The BJ-FSM then implements a function of r to determine the back-jumping state, i.e.,

$$s_{bj} = f(r).$$

For example, if s_{bj} is an l-bit binary number, BJ-FSM can arbitrarily select l bits from r and assign the value to s_{bj}. If the first l bits of r are selected, we have

$$f(r) = r[0 : l - 1].$$

Meanwhile, BJ-FSM sends a back-jumping request to the other blocks of the circuit, such that the circuit back-jumps to s_{bj} in the encrypted mode, where it keeps checking the authentication status of the circuit. SANSCrypt does not set any specific requirement on the PRNG. Any PRNG architecture can be used based on the design budget and the desired security level. For example,

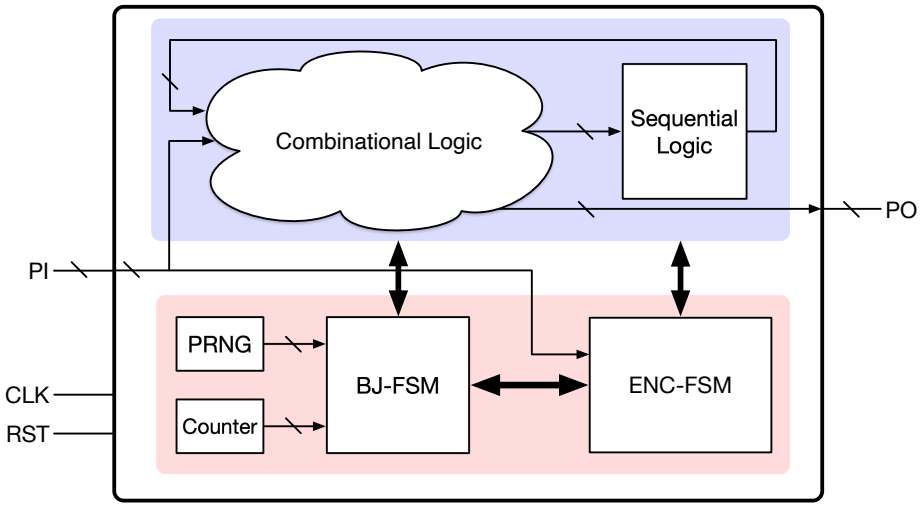


Fig. 5. Schematic view of SANSCrypt.

linear PRNGs, such as Linear Feedback Shift Registers (LFSRs), provide higher speed and lower area overhead but tend to be more vulnerable than cipher-based PRNGs, such as AES, which are, however, more expensive.

3.5 Encryption Finite State Machine (ENC-FSM)

The Encryption Finite State Machine (ENC-FSM) determines whether the user's key sequence is correct and, if it is not correct, takes actions to corrupt the functionality of the original circuit. Without creating extra input ports for the authentication, the input of the ENC-FSM is provided via the primary input ports. The output *enc_out* of ENC-FSM is an n-bit-long array, which can be used, together with a set of XOR gates, to corrupt the circuit function [3]. For example, in Fig. 7, a 3-bit array *enc_out* is connected to six nodes in the original circuit via XOR gates. In this paper, XOR gates are inserted at randomly selected nodes. However, any other combinational logic encryption technique is also applicable. As a design parameter, we denote by *node coverage* the ratio between the number of inserted XOR gates and the total number of combinational logic gates in the circuit.

Only one state of ENC-FSM, termed *auth*, is used in the functional mode. In *auth*, all bits in *enc_out* are set to zero and the original circuit functionality is not corrupted. In the other states, the value of *enc_out* changes based on the state, but at least one bit is set to one to guarantee that the circuit output is incorrect. A sample truth table for a 3-bit *enc_out* array is shown in Table 1. When the circuit is not in *auth*, i.e., in the encrypted mode, *enc_out* changes its value based on the state of the encryption FSM. Such an approach makes it difficult for signal analysis attacks, aiming to locate signals with low switching

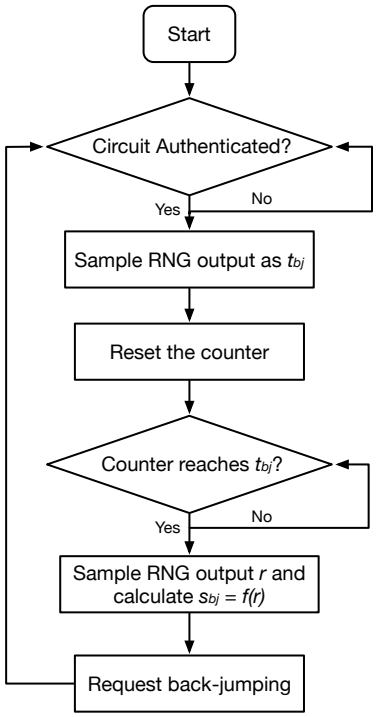


Fig. 6. Flowchart of BJ-FSM.

activity in the encrypted mode, to find *enc_out* and bypass ENC-FSM. After a successful authentication, the circuit resumes its normal operation. Additional registers are, therefore, required in the ENC-FSM to store the circuit state before back-jumping so that it can be resumed after authentication.

3.6 Guaranteeing Real-Time Operation

Unlike previous sequential logic encryption methods, SANSCrypt requires the user to sporadically be re-authenticated amid the circuit's normal operation. As discussed in Section 4, this feature can significantly raise the attack difficulty. However, it can also cause timing overhead and impact the performance in time-sensitive applications that require prompt, real-time response, or guarantees that a time-critical or safety-critical task meets a pre-defined deadline. For example, upon detection of a vehicle collision, authentication tasks should be preempted by the airbag control in the attempt to protect passengers. In this Section, we present an enhanced back-jumping FSM (EBJ-FSM) design that delivers precise, guaranteed, and predictable timing for real-time operation.

We denote by V_I and V_S the set of all possible primary input patterns and states of a circuit, respectively. We then assume without loss of generality that

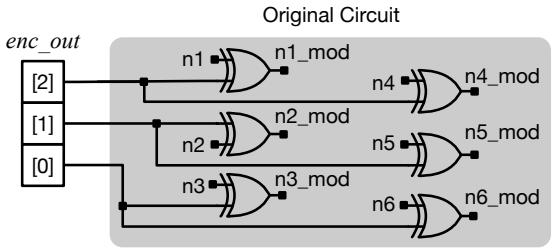


Fig. 7. *enc_out* controls the original circuit via XOR gates.

Table 1. Truth Table for a 3-Bit *enc_out* Array

State	E0	E1	E2	E3	E4	Auth
enc_out[0]	0	1	1	1	1	0
enc_out[1]	1	0	1	1	0	0
enc_out[2]	1	1	1	0	0	0

any input pattern in a set

$$I_p = \{i_1, i_2, ..., i_n\},$$

where $I_p \subseteq V_I$, can trigger a time-critical task whose execution should immediately start or be queued after the ongoing time-critical task. Such tasks should be completed, without interruption, within a number of clock cycles given by a function *deadline*. We assume that a task deadline depends, in general, on the triggering input and the current state of the circuit, i.e.,

$$deadline : V_I \times V_S \rightarrow \mathbb{N}.$$

When a time-critical task is requested by an input pattern in I_p, the deadline function returns the remaining number of clock cycles required without interruption to finish this task and the ongoing time-critical tasks, if any, based on the current state s. When the circuit is in an idle state, i.e., no time-critical task is being executed, and the current input pattern i satisfies $i \notin I_p$, the *deadline* function returns zero. The *deadline* function models the scheduling algorithm determining the priority among tasks and can be customized based on the desired application. Fig. 8 shows the flowchart of EBJ-FSM for time-sensitive applications. On top of the basic back-jumping feature, two locations in the flow chart handle task prioritization in the encrypted mode (①) and the functional mode (②), respectively.

High-Priority Task Triggered in the Encrypted Mode (Case ①). The enhanced BJ-FSM (EBJ-FSM) allows the circuit to enter the functional mode in two scenarios: (1) upon a successful authentication, and (2) when a high-priority

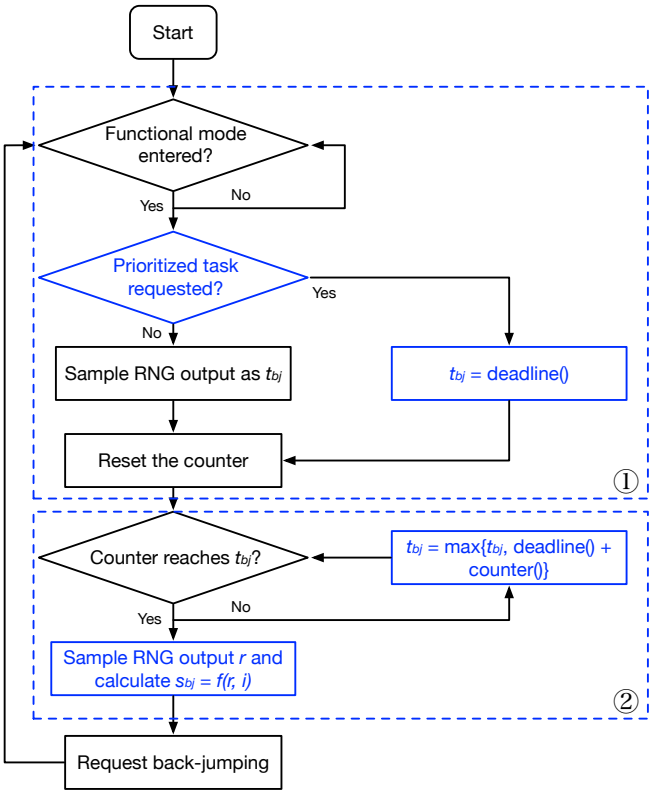


Fig. 8. Flowchart of EBJ-FSM.

task is triggered. EBJ-FSM monitors these two events at each clock cycle. Once the functional mode is entered, EBJ-FSM checks whether the second scenario occurred, i.e., a high-priority task was triggered. If this is the case, meaning that the authentication process was interrupted, EBJ-FSM sets the back-jumping time t_{bj} to the time required to complete the task execution as computed by the *deadline* function. When the high-priority task terminates and there are no new task requests with high priority, the circuit back-jumps to the encrypted mode and the authentication procedure is resumed.

High-Priority Task Triggered in the Functional Mode (Case ②). In the functional mode, EBJ-FSM continually checks whether the counter output has reached the threshold t_{bj}. If so, it will request back-jumping as is the case for the BJ-FSM design in Fig. 6. However, in this case, the back-jumping time t_{bj} is updated at each clock cycle via the following formula,

$$t_{bj,k} := \max\{t_{bj,k-1}, deadline() + counter()\},$$

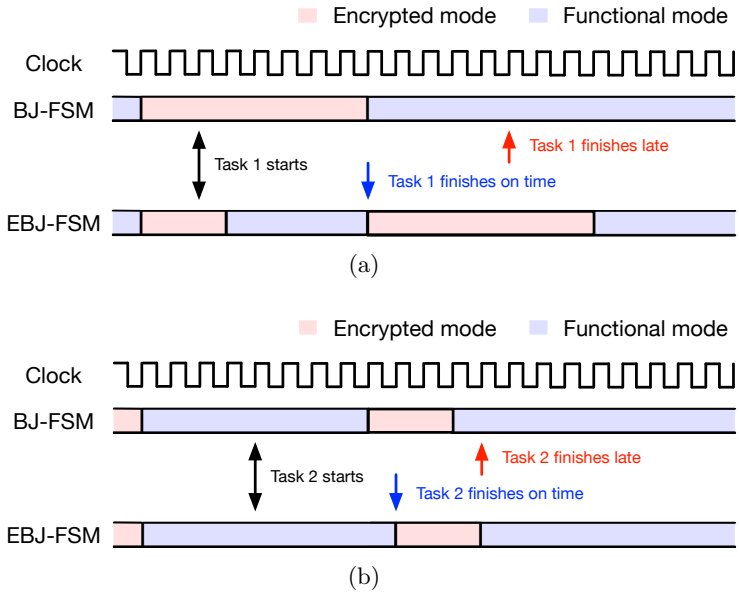


Fig. 9. Examples of high-priority tasks triggered in (a) the encrypted mode and (b) the functional mode. Both tasks require 5 clock cycles to finish with BJ-FSM and EBJ-FSM.

where $counter()$ returns the current counter output and $t_{bj,k}$ refers to the stored back-jumping time at time k. Effectively, the back-jumping time t_{bj} is prolonged if a high-priority task cannot be completed before the next designated back-jumping operation.

Fig. 9 shows the timing diagrams for two high-priority tasks triggered in two different modes. While the basic SANSCrypt protocol causes delays to the task execution, the EBJ-FSM guarantees that there is no delay in the execution of the critical tasks and no impact on the real-time performance of the circuit.

As a further enhancement, we observe that the function that determines the back-jumping state s_{bj} can also be modified by adding the current input value i as an argument. We propose this modification to mitigate a potential vulnerability associated with FSM structural analysis of the basic SANSCrypt architecture, as further illustrated in Section 4.

4 Security and Performance Analysis

We analyze SANSCrypt's resilience against existing attacks and estimate its timing overhead due to the multi-authentication protocol.

4.1 Brute-Force Attack

We assume that the number of primary inputs used as key inputs is $|i|$ and a round of authentication requires c clock cycles to apply the correct key sequence. If the attacker has no *a priori* knowledge of the correct key sequence, then the average number of attempts needed to find the correct key sequence for each authentication step, $\overline{\tau}$, can be computed as follows:

$$\overline{\tau} = (2^{|i|\cdot c} + 1)/2 \approx 2^{|i|\cdot c-1},$$

where we use $\overline{\tau}$ to represent the expected value of the random variable τ. This amounts to the same brute-force attack complexity of HARPOON, where the encrypted circuit needs only one round of authentication. Due to the multi-authentication protocol implemented in SANSCrypt, the attacker needs to find the correct key sequences for more than one round of authentication. Each correct key sequence depends on the back-jumping states that are determined by the PRNG output. To achieve maximum protection, a designer can associate each PRNG output value with a unique back-jumping state, hence a unique key sequence for the authentication. Therefore, the average brute-force effort to guess all the correct key sequences $\overline{T}$ is

$$\overline{T} = N_r \cdot \overline{\tau} = N_r \cdot 2^{|i|\cdot c-1},$$

where N_r is the number of possible values of the PRNG output. For a 10-bit PRNG, if $|i| = 32$ and $c = 8$, this average attack effort reaches 5.8×10^{79}.

Even if all the key sequences are known, it still remains challenging to infer when each key sequence should be applied, as the attacker should find the back-jumping time associated with the sequence, and this is independent of the sequence itself. To account for the time uncertainty, we first estimate the effort for guessing the back-jumping time for one authentication round. The back-jumping time ranges from one to N_r cycles following a uniform distribution. Therefore, the average brute-force effort to correctly guess the time is

$$\bar{t}_{bj} = \frac{N_r}{2},$$

while the average effort to correctly find both the key sequence and the time at which to apply it becomes

$$\bar{t}_{bf} = E[\tau t_{bj}] = \overline{\tau}\bar{t}_{bj} = \frac{N_r}{2} \cdot 2^{|i|\cdot c-1}.$$

Suppose the attacker needs to perform at least m rounds of authentication, where $m \geq N_r$ and all N_r key sequences are used for the authentication at least once. The expected value for m can be calculated as follows, using a result from the coupon collector's problem [36]:

$$\overline{m} = N_r \cdot \left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{N_r - 1} + \frac{1}{N_r}\right).$$

In our previous example, where $N_r = 1024$, we have $\overline{m} = 7689$. The average brute-force attack effort to find the back-jumping times and the key sequences for m authentication steps would then be

$$\overline{T}_{bf} = E\left[(t_{bj}\tau)^m\right] = E_m\left[E[(t_{bj}\tau)^m \,|\, m]\right] = E_m\left[\left(\frac{N_r}{2}2^{|i|\cdot c-1}\right)^m \middle| m\right].$$

For simplicity, we provide a lower bound for the expectation above. Since $m \geq N_r$ and $\frac{N_r}{2} \cdot 2^{|i|\cdot c-1} > 1$, we have the following lower bound for $\overline{T}_{bf}$:

$$\overline{T}_{bf} \geq \left(\frac{N_r}{2} \cdot 2^{|i|\cdot c-1}\right)^{N_r}.$$

In our example, the average brute-force effort will be lower bounded by 1.8×10^{81379}, which makes a brute-force attack infeasible and exponentially harder than in previous sequential obfuscation methods.

4.2 Sequential SAT-Based Attack

A SAT-based attack can be carried out on existing sequential logic encryption methods by unrolling the sequential circuit [25]. In this paper, we implement such an attack to validate the resilience of methods such as HARPOON and SANSCrypt by adapting previously proposed attack strategies [18, 19] to a setting in which a dynamic key, i.e., a sequence of keys applied at different clock cycles, is presented via the primary input ports of the circuit.

Fig. 10 shows the schematic of an unrolled circuit under the assumption that the number of clock cycles, n, required by the encrypted circuit to enter the functional mode after reset is known. The primary input ports of the first n replicas of the encrypted circuit, marked in red, act as the key ports K of the unrolled circuit. Starting with the $(n+1)^{th}$ circuit replica, the primary input and output ports of the encrypted circuit, marked in blue and magenta, act, instead, as the primary input ports I and the primary output ports O of the unrolled circuit, respectively. A combinational miter circuit can then be assembled using this unrolled circuit to mount a combinational SAT-based attack and find the correct key. If the SAT-based attack fails to find the correct key with $(n + 1)$ circuit replicas, the circuit will be unrolled once more to repeat the attack.

The attack described above would still be ineffective on SANSCrypt, since it can retrieve the first key sequence but would fail to discover when the next back-jumping occurs and what would be the next key sequence. Even if the attacker knows when the next back-jumping occurs, the attack will fail due to the large number of circuit replicas needed to find all the key sequences, as empirically observed in Section 5.

4.3 FSM Extraction and Structural Analysis

As discussed in Section 2, a common shortcoming of previous sequential encryption schemes is the easy separation of states between the encrypted mode and

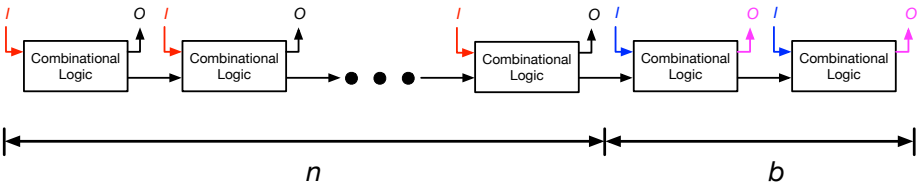


Fig. 10. An unrolled version of the encrypted circuit which requires n clock cycles to find the key sequence.

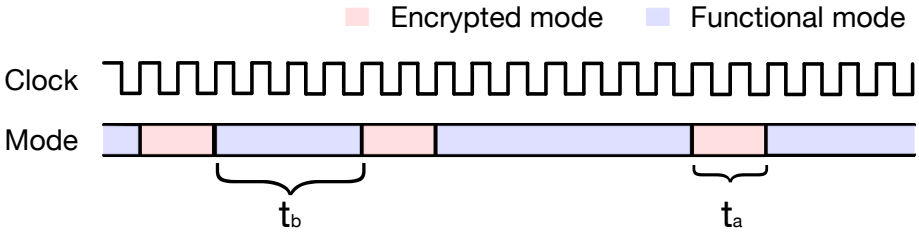


Fig. 11. Circuit mode switching for an authenticated user.

the functional mode due to the fact that only one transition goes though the two modes. SANSCrypt addresses this issue by designing more than one transition between the two modes, as shown in Fig. 4. In a basic BJ-FSM design, shown in Fig. 6, the back-jumping state is solely determined by the PRNG. Because transitions in FSM are typically determined also by the primary input, attackers can potentially identify all the back-jumping transitions by analyzing the transition conditions. The enhanced BJ-FSM design (EBJ-FSM), shown in Fig. 8, circumvents this vulnerability by also using the primary inputs to determine the back-jumping state.

Without extracting the FSM, an attacker may also try to locate and isolate the output of ENC-FSM by looking for low signal switching activities when the circuit is in the encrypted mode. SANSCrypt addresses this risk by expanding the output of ENC-FSM from one bit to an array. The value of each bit changes frequently with state changing in the encrypted mode, which makes it difficult for attackers to find them based only on signal switching activities.

4.4 Cycle Delay Analysis

Due to multiple back-jumping and authentication operations in SANSCrypt, additional clock cycles will be required. Suppose that each authentication requires t_a clock cycles and the circuit stays in the functional mode for t_b clock cycles before the next back-jumping occurs, as shown in Fig. 11. Assuming that no higher-priority tasks are triggered, the cycle delay overhead can be computed as the ratio $O_{cd} = t_a/t_b$.

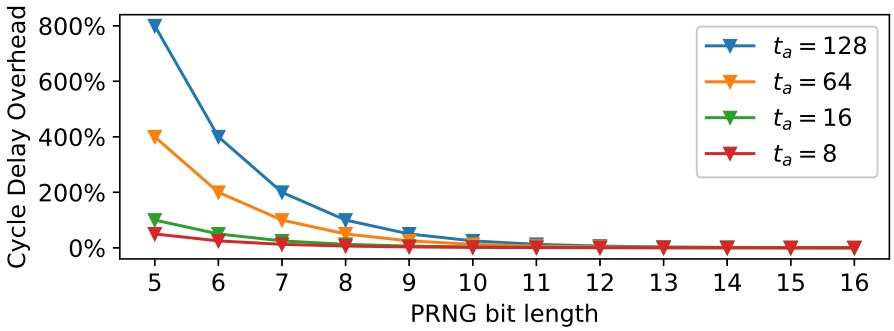


Fig. 12. Average cycle delay as a function of PRNG bit length when the key sequence cycle length t_a is 8, 16, 64, and 128.

Table 2. Overview of the Selected Benchmark Circuits

Circuit	**s27**	**s298**	**s1238**	**s9234**	**s15850**	**s35932**	**s38584**
Input	4	3	14	36	77	35	38
Output	1	6	14	39	150	320	304
DFF	3	14	18	211	534	1728	1426
Gate	10	119	508	5597	9772	16065	19253

Specifically, for an n-bit PRNG, the average t_b is equal to the average output value, i.e., 2^{n-1}. To illustrate how the cycle delay overhead is influenced by this encryption, Fig. 12 shows the relation between average cycle delay overhead and PRNG bit length. The clock cycles (t_a) required for each authentication are set to 8, 16, 64, and 128. When the PRNG bit length is small, the average cycle delay increases significantly as t_a increases. However, the cycle delay can be reduced by increasing the PRNG bit length. For example, the average cycle delay overhead becomes negligible ($\leq 1\%$) for all the four cases when the PRNG bit length is 14 or larger.

5 Simulation Results

We first evaluate the effectiveness of SANSCrypt on seven ISCAS'89 sequential benchmark circuits of different sizes, as summarized in Table 2. All the experiments are executed on a Linux server with 48 2.1-GHz processor cores and 500-GB memory. We implement our technique on the selected circuits with different configurations and use a 45-nm Nangate Open Cell Library [37] to synthesize the encrypted netlists for area optimization under a critical-path delay constraint that targets the same performance as for the original netlists. For the purpose of illustration, we realize the PRNG using Linear Feedback Shift Registers (LFSRs) with different sizes, ranging from 5 to 15 bits. An LFSR provides an area-efficient implementation and has often been used in other logic

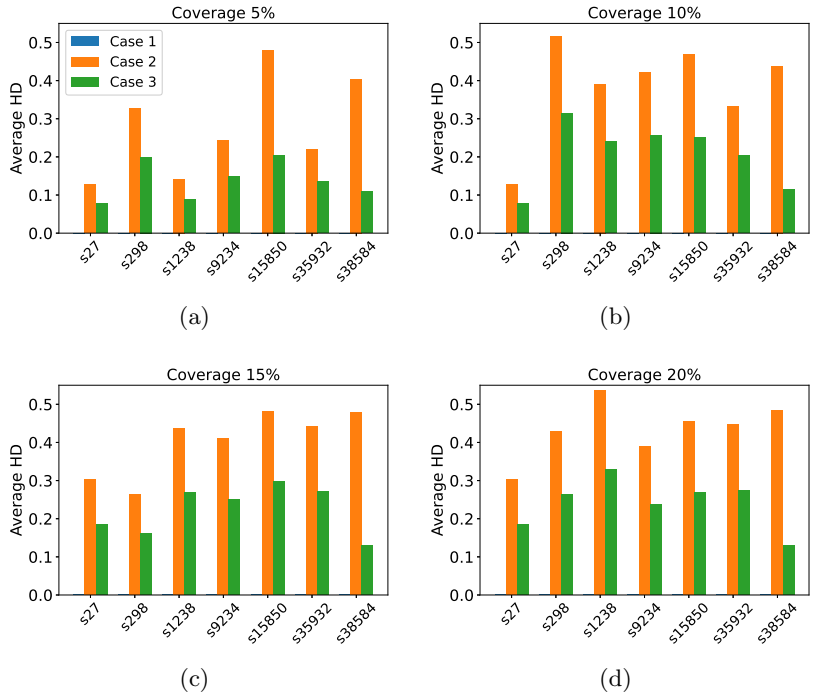


Fig. 13. The average HD for different node coverage: (a) 5%, (b) 10%, (c) 15%, and (d) 20%.

encryption schemes in the literature [8, 38]. We choose a random 8-cycle-long key sequence as the correct key, and select 5%, 10%, 15%, and 20% as node coverage levels. Finally, we use the Hamming distance (HD) between the correct and the corrupted output values as a metric for the output corruptibility. If the HD is 0.5, the effort spent to identify the incorrect bits is maximum.

Functional Verification. First, we simulate all the encrypted circuits with (case 1) and without (case 2) the correct key sequences, by applying a randomly generated input vector that is 1000-cycle long. We then compare the circuit output with the golden output from the original netlist and calculate the HD between the two. Each simulation is repeated for 1000 times to obtain the average HD. Moreover, we demonstrate the additional robustness of SANSCrypt by simulating a scenario (case 3) in which the attacker assumes that the encryption is based on a single-authentication protocol and, thus, provides only the first correct key sequence upon reset. Fig. 13 shows the average HD in these three cases. For all the circuits, the average HD is zero only in case 1, when all the correct key sequences are applied at the right clock cycles. Otherwise, in case 2 (orange) and case 3 (green), we observe a significant increase in the average HD.

Table 3. SAT-based attack runtime for finding the first 7 key sequences

Key Seq. Index	1 (HARPOON)	2	3	4	5	6	7
Runtime [s]	4	123	229	1941	1301	2202	25571

Table 4. ADP Overhead Results for Full Encryption

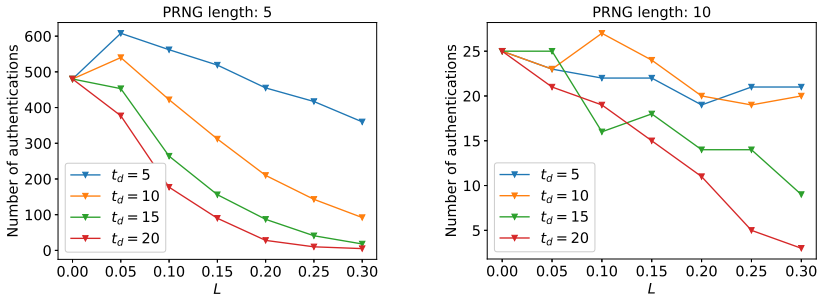
Circuit	**s27**				**s298**				**s1238**				**s9234**			
Node Coverage	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
Area [%]	1418.5	1418.5	1403.2	1403.2	413.0	427.3	425.2	453.8	144.8	165.7	176.0	189.2	114.6	131.7	144.5	160.1
Power [%]	1627.7	1627.7	1627.5	1627.5	385.7	390.6	389.9	402.8	217.8	232.1	235.0	249.8	179.8	197.5	188.0	190.6
Delay [%]	0.0	0.0	1.4	1.4	0.0	0.0	0.0	0.5	0.0	0.0	0.0	5.8	0.0	0.0	0.9	3.6
Circuit	**s15850**				**s35932**				**s38584**				**Average***			
Node Coverage	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
Area [%]	92.9	112.1	120.1	133.9	116.3	129.5	139.4	151.6	133.5	140.9	158.7	165.6	120.4	136.0	147.8	160.1
Power [%]	127.4	142.3	153.2	163.0	98.4	101.9	101.2	103.0	123.9	128.8	142.0	140.3	149.5	160.5	163.9	169.4
Delay [%]	-0.3	0.0	0.1	0.6	-0.4	0.0	4.3	5.3	0.6	2.0	0.4	4.9	0.0	0.4	1.1	4.0

*Excluding s27 and s298.

The average HD in case 3 is always smaller than that of case 2 because, in case 3, the correct functionality is recovered for a short period of time, after which the circuit jumps back to the encrypted mode. The longer the overall runtime, the smaller will be the impact of this transparency window in which the circuit exhibits the correct functionality.

Sequential SAT-Based Attacks. We apply the sequential SAT-based attack in Section 4 to circuit *s1238* with a 5-bit LFSR and 20% node coverage, under a stronger attack model, in which the attacker knows when to apply the key sequences. Table 3 shows the runtime to find the first set of 7 key sequences. The runtime remains exponential in the number of key sequences, which makes sequential SAT-based attacks impractical for large designs.

Impact of High-Priority Tasks. We further characterize the behavior of SANSCrypt in the presence of high-priority tasks. We consider the largest ISCAS benchmark *s38584* and assume, without loss of generality, that all the high-priority tasks to be executed on the encrypted circuit have the same deadline t_d. For a sequence of input patterns, we define the high-priority task load L as the ratio between the number of high-priority task requests in the sequence and the sequence length. Fig. 14(a) and Fig. 14(b) show simulation results under different task loads and deadlines for $10,000$ clock cycles, when the PRNG length is 5 and 10, respectively. L ranges from 0 to 0.3, while the task deadline takes four different values within 5 and 20. When $L = 0$, no high-priority tasks are requested and the numbers of authentications within $10,000$ clock cycles are 480 and 25 for the two different PRNG lengths, respectively. When L or t_d increases, it is more likely for a high-priority task to either interrupt or postpone the authentication step, leading to a decreasing number of authentications, as shown



(a) (b)

Fig. 14. Number of authentications required within 10,000 clock cycles on *s38584* for different priority task loads L. The PRNG length is (a) 5 and (b) 20.

in Fig. 14(a). However, in a scenario in which the number of authentications is already as low as 25 without execution of high-priority tasks, as in Fig. 14(b), the likelihood that a task needs to interrupt or postpone the authentication process decreases. As a result, increasing L or t_d does not significantly affect the number of authentications as in the scenario of Fig. 14(a). When $L = 0.05$ and $t_d = 5$ or 10, the number of authentications becomes larger than in the absence of high-priority tasks in Fig. 14(a), an artifact due to the non-ideality of the LFSR used in the design, which disappears when using a higher-quality PRNG. On the other hand, when many time-consuming high-priority tasks need to be executed, i.e., when the task load L is 0.3 and the deadline t_d is 20, we observe that 5 and 3 authentications are still required per 10,000 clock cycles in Fig. 14(a) and Fig. 14(b), respectively, which keeps the multi-authentication protocol effective. Overall, SANSCrypt is capable of delivering security as well as precise, guaranteed, and predictable timing in the execution of time-critical tasks.

Implementation Overhead. Finally, Table 4 reports the synthesized area, power, and delay (ADP) overhead due to the implementation of our technique. In more than 70% of the circuits, the delay overhead is less than 1%, and exceeds the required clock cycle by at most 5.8%. Except for *s27* and *s298*, characterized by a small gate count, all the other circuits show average area and power overhead of 141.1% and 160.8%, respectively, which is expected due to the additional number of registers required in ENC-FSM to guarantee that the correct state is entered upon re-authentication. However, because critical modules in large SoCs may only account for a small portion of the area, this overhead becomes affordable under partial obfuscation. For example, we encrypted a portion of state registers in *s38584*, the largest ISCAS'89 benchmark, using SANSCrypt. We then randomly inserted additional XOR gates to achieve the same HD as in the case of full encryption. Table 5 reports the overhead results after synthesis,

Table 5. ADP Overhead Results for Partial Encryption

Encrypted registers/Total registers	100%	50%	25%	10%	5%	2.5%	1%
Area [%]	133.5	71.6	49.1	33.4	27.8	23.5	22.4
Power [%]	123.9	40.2	9.6	-12.8	-20.5	-22.1	-25.0
Delay [%]	0.6	1.8	2.1	4.2	5.4	3.9	4.6

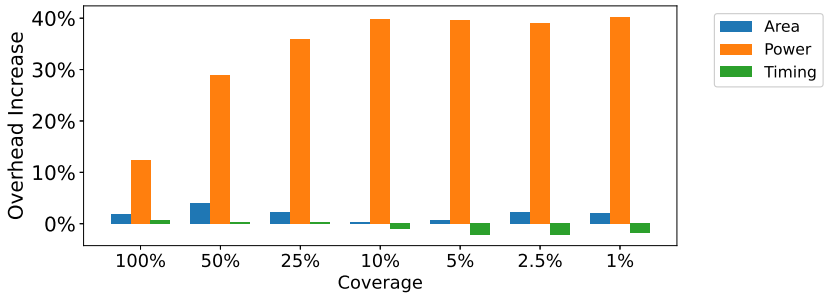


Fig. 15. Area, timing, power overhead increase, compared with the original SANSCrypt scheme, after the implementation of EBJ-FSM under different coverage ratios on *s38584*.

when the ratio between the encrypted state registers and the total number of state registers decreases from 100% to 1%. Encrypting 10% of the registers will only cost 33.4% of the area while incurring negative power overhead and 4.2% delay overhead. On the other hand, implementing the enhanced design based on the EBJ-FSM on *s38584*, while using the same settings as in Table 5, causes an increase in the ADP overhead with respect to the basic SANSCrypt architecture, as shown in Fig. 15. Yet, the increase in both the area and timing overhead is below 4%, with the timing overhead often being lower than in the baseline. The increase in power overhead is substantial, but it is partially compensated by the negative power overhead of the baseline design in Table 5, and therefore still acceptable.

6 Conclusion

We proposed SANSCrypt, a robust sequential logic encryption technique relying on a sporadic authentication protocol, in which re-authentications are carried out at pseudo-randomly selected time slots to significantly increase the attack effort. By allowing flexible interruption and postponement of authentication tasks upon requests from high-priority tasks, SANSCrypt is capable of guaranteeing reliable timing and seamless operation in real-time and time-sensitive applications. Future work includes optimizing the implementation to further reduce the overhead, and investigating key manager architectures to guarantee reliable key delivery in large systems on chip.

Acknowledgments

This work was supported in part by the Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-1-7817.

References

1. R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.
2. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.
3. J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Trans. Computers*, vol. 64, no. 2, pp. 410–424, 2013.
4. M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. SIGSAC Conf. Computer and Communications Security*, pp. 1601–1618, 2017.
5. M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.
6. R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.
7. M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "CamoPerturb: Secure IC camouflaging for minterm protection," in *2016 IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, pp. 1–8, 2016.
8. K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 14–19, 2015.
9. E. Charbon, "Hierarchical watermarking in IC design," in *IEEE Proc. Custom Integrated Circuits Conf.*, pp. 295–298, 1998.
10. P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 137–143, 2015.
11. P. Chakraborty, J. Cruz, and S. Bhunia, "SURF: Joint structural functional attack on logic locking," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 181–190, 2019.
12. Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "SigAttack: New high-level sat-based attack on logic encryptions," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 940–943, 2019.
13. V. V. Menon, G. Kolhe, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "System-level framework for logic obfuscation with quantified metrics for evaluation," in *Secure Development Conf. (SecDev)*, pp. 89–100, 2019.
14. Y. Hu, V. V. Menon, A. Schmidt, J. Monson, M. French, and P. Nuzzo, "Security-driven metrics and models for efficient evaluation of logic encryption schemes," in *ACM-IEEE Int. Conf. Formal Methods and Models for System Design (MEMOCODE)*, pp. 1–5, 2019.

15. G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, "Secured flipped scan-chain model for crypto-architecture," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2080–2084, 2007.
16. S. Paul, R. S. Chakraborty, and S. Bhunia, "Vim-scan: A low overhead scan design approach for protection of secret key in scan-based secure chips," in *IEEE VLSI Test Symp. (VTS)*, pp. 455–460, 2007.
17. X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure scan and test using obfuscation throughout supply chain," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, 2017.
18. M. El Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 33–40, IEEE, 2017.
19. K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-condition crunching for fast sequential circuit deobfuscation," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 534–539, 2019.
20. P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *IEEE Asian Hardware Oriented Security and Trust Symp. (AsianHOST)*, pp. 56–61, 2018.
21. D. Sisejkovic, F. Merchant, L. M. Reimann, H. Srivastava, A. Hallawa, and R. Leupers, "Challenging the security of logic locking schemes in the era of deep learning: A neuroevolutionary approach," *arXiv preprint arXiv:2011.10389*, 2020.
22. Y. Hu, K. Yang, S. Dutta Chowdhury, and P. Nuzzo, "Risk-aware cost-effective design methodology for integrated circuit locking," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 1182–1185, IEEE, 2021.
23. A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall, "Interlocking obfuscation for anti-tamper hardware," in *Proc. Cyber Security and Information Intelligence Research Workshop*, pp. 1–4, 2013.
24. Y. Kasarabada, S. R. T. Raman, and R. Vemuri, "Deep state encryption for sequential logic circuits," in *IEEE Computer Society Annual Symp. VLSI (ISVLSI)*, pp. 338–343, 2019.
25. T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit sequential logic obfuscation: Attacks and defenses," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1–4, 2017.
26. D. Duvalsaint, Z. Liu, A. Ravikumar, and R. Blanton, "Characterization of locked sequential circuits via ATPG," in *IEEE Int. Test Conf. in Asia (ITC-Asia)*, pp. 97–102, 2019.
27. A. Bhargav-Spantzel, A. C. Squicciarini, S. Modi, M. Young, E. Bertino, and S. J. Elliott, "Privacy preserving multi-factor authentication with biometrics," *Journal of Computer Security*, vol. 15, no. 5, pp. 529–560, 2007.
28. F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1929–1934, 1989.
29. Y. Hu, K. Yang, S. Nazarian, and P. Nuzzo, "SANSCrypt: A sporadic-authentication-based sequential logic encryption scheme," in *IFIP/IEEE Int. Conf. Very Large Scale Integration (VLSI-SoC)*, pp. 129–134, 2020.
30. T. Meade, Y. Jin, M. Tehranipoor, and S. Zhang, "Gate-level netlist reverse engineering for hardware security: Control logic register identification," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1334–1337, 2016.
31. M. Brunner, J. Baehr, and G. Sigl, "Improving on state register identification in sequential hardware reverse engineering," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, 2019.

32. J. Geist *et al.*, "RELIC-FUN: Logic identification through functional signal comparisons," in *Proc. Design Automation Conf. (DAC)*, 2020.
33. J. Dofe and Q. Yu, "Novel dynamic state-deflection method for gate-level design obfuscation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 273–285, Feb. 2018.
34. A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," 2003.
35. Y. Kasarabada, S. Chen, and R. Vemuri, "On SAT-based attacks on encrypted sequential logic circuits," in *Int. Symp. Quality Electronic Design (ISQED)*, pp. 204–211, 2019.
36. P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992.
37. Silvaco, "45nm open cell library," 2019.
38. M. S. Rahman, A. Nahiyan, S. Amir, F. Rahman, F. Farahmandi, D. Forte, and M. Tehranipoor, "Dynamically obfuscated scan chain to resist oracle-guided attacks on logic locked design.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 946, 2019.