Security Assessment of Dynamically Obfuscated Scan Chain Against Oracle-guided Attacks

M SAZADUR RAHMAN, ADIB NAHIYAN, and FAHIM RAHMAN, University of Florida, USA
SAVERIO FAZZARI, Booz Allen Hamilton, USA
KENNETH PLAKS, Defense Advanced Research Projects Agency, USA
FARIMAH FARAHMANDI, DOMENIC FORTE, and MARK TEHRANIPOOR, University of Florida, USA

Logic locking has emerged as a promising solution to protect integrated circuits against piracy and tampering. However, the security provided by existing logic locking techniques is often thwarted by Boolean satisfiability (SAT)-based oracle-guided attacks. Criteria for successful SAT attacks on locked circuits include: (i) the circuit under attack is fully combinational, or (ii) the attacker has scan chain access. To address the threat posed by SAT-based attacks, we adopt the dynamically obfuscated scan chain (DOSC) architecture and illustrate its resiliency against the SAT attacks when inserted into the scan chain of an obfuscated design. We demonstrate, both mathematically and experimentally, that DOSC exponentially increases the resiliency against key extraction by SAT attack and its variants. Our results show that the mathematical estimation of attack complexity correlates to the experimental results with an accuracy of 95% or better. Along with the formal proof, we model DOSC architecture to its equivalent combinational circuit and perform SAT attack to evaluate its resiliency empirically. Our experiments demonstrate that SAT attack on DOSC-inserted benchmark circuits timeout at minimal test time overhead, and while DOSC requires less than 1% area and power overhead.

CCS Concepts: • **Security and privacy → Security in hardware**; **Hardware security implementation**; **Hardware-based security protocols**;

Additional Key Words and Phrases: SAT attack, logic locking, scan obfuscation, mathematical model for satisfiability, hardware obfuscation

ACM Reference format:
M Sazadur Rahman, Adib Nahiyan, Fahim Rahman, Saverio Fazzari, Kenneth Plaks, Farimah Farahmandi, Domenic Forte, and Mark Tehranipoor. 2021. Security Assessment of Dynamically Obfuscated Scan Chain Against Oracle-guided Attacks. *ACM Trans. Des. Autom. Electron. Syst.* 26, 4, Article 29 (March 2021), 27 pages.
https://doi.org/10.1145/3444960

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) (Grant No. FA8650-18-1-7821). Disclaimer: The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.
Authors' addresses: M S. Rahman, A. Nahiyan, and F. Rahman, University of Florida, Gainesville, Florida; emails: {mohammad.rahman, adib1991}@ufl.edu, fahimrahman@ece.ufl.edu; S. Fazzari, Booz Allen Hamilton, Arlington, Virginia; email: saverio.fazzari.ctr@darpa.mil; K. Plaks, Defense Advanced Research Projects Agency, Arlington, Virginia; email: kenneth.plaks@darpa.mil; F. Farahmandi, D. Forte, and M. Tehranipoor, University of Florida, Gainesville, Florida; emails: {farimah, dforte, tehranipoor}@ece.ufl.edu.

1084-4309/2021/03-ART29 $15.00
https://doi.org/10.1145/3444960

1 INTRODUCTION

Due to the ever-increasing costs of maintaining a cutting-edge semiconductor fabrication facility, there has been a shift in the integrated circuit (IC) design flow giving rise to fabless semiconductor companies, third party design houses, and contract foundries. This trend has accelerated innovation, lowered cost, and reduced time-to-market. However, with many entities involved in design, manufacturing, integration, and distribution located across the globe, original intellectual property (IP) owners can no longer monitor the entire process. As a result, they are now facing threats of IP theft/piracy, tampering, counterfeiting, reverse engineering, and IC overproduction [1]. From a global perspective, where IP protection laws (and the degree of their enforcement) vastly vary from one country to another, IP protection can no longer be limited to passive methods such as patents, copyrights, and watermarking [2] that merely deter these threats.

To address these concerns, researchers have proposed several *design-for-trust* techniques, e.g., IP encryption [3], logic locking [4, 5], state-space obfuscation [6], IC camouflaging [7], split manufacturing [8], and split testing [9]. In addition, the IEEE IP encryption standard (IEEE-P1735) was developed to protect the confidentiality of IPs and has been widely adopted by semiconductor industry [3]. However, logic-locking [4, 12, 13] has been studied most widely, since the others possess several critical vulnerabilities [10, 11].

Logic locking, also known as logic obfuscation, inserts additional gates that are controlled by a key into the design to hide original functionality and prevent reverse engineering, piracy, overproduction, and tampering. The proper function of the design is ensured once correct unlocking key inputs are provided from a tamper-proof memory. In search of an unbreakable logic obfuscation technique, researchers proposed several key insertion methods [4, 12, 13]. The prospect of logic obfuscation and emerging threats of tampering [14] have propelled government agencies like Defense Advanced Research Project Agency (DARPA) to call for programs like "Automatic Implementation of Secure Silicon" (AISS) [15] under "Electronics Resurgence Initiative" [16] to automatically include logic locking in chip designs for IP protection. Nevertheless, logic locking possesses critical vulnerabilities. Most notably, recent work has demonstrated that all existing logic obfuscation techniques are susceptible to *Boolean satisfiability* (SAT)-based attacks that extract the locking key quickly [17].

A conceptual representation of the SAT attack is shown in Figure 1. The SAT attack constitutes a miter-like circuit using two copies of the locked netlist to identify distinguishing-input-patterns (DIPs) and applies that DIP to the unlocked chip to rule out incorrect keys. To mitigate SAT attacks, several SAT-resistant logic obfuscation techniques have been recently proposed such as SARLock [18], Anti-SAT [19], and SFLL [20]. However, logic locking techniques that claim to be highly resistant to SAT attacks, tend to have low output corruptibility and structural traces that are more vulnerable to other attacks such as bypass attack [21], Signal Probability Skew (SPS) attack [22], removal attack [23], AppSAT attack [24], and FALL attack [25]. Hence, there remains a critical need to develop countermeasures to SAT attacks.

SAT and its variant attacks, such as AppSAT [24] and ScanSAT [26], have been carried out on combinational designs. For a sequential design, it is assumed that an attacker can exploit the design-for-test (DFT) infrastructure (i.e., scan chain) to divide the sequential design into smaller combinational circuits that SAT solvers can handle. Hence, the approach taken in this article concentrates on limiting the data collection capability required for a successful SAT attack. We adopt one of our prior schemes that was aimed to secure a design against scan-based side-channel attacks by dynamically obfuscating scan chains [27]. We take inspiration from this scan chain protection technique to develop *dynamically obfuscated scan chain* (DOSC) for protection of obfuscated circuits and perform a comprehensive security assessment. Our major contributions are

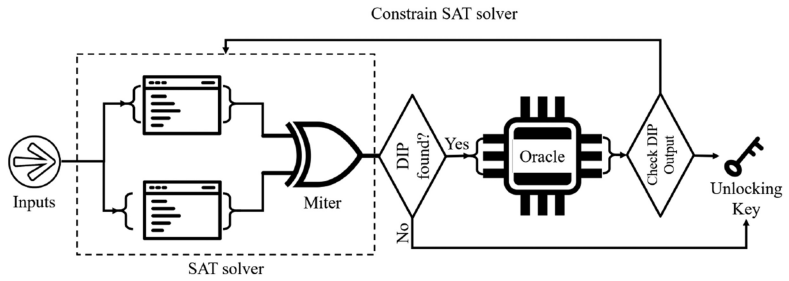

Fig. 1. Conceptual overview of SAT attack. SAT attack requires two copies of the locked netlist to identify DIPs and an unlocked chip to verify key-guesses.

- We investigate dynamic obfuscation of the scan chain and propose its effectiveness in resisting oracle-guided attacks, e.g., SAT and its variants (AppSAT, ScanSAT, TimingSAT, etc.), key sensitization, SPS, functional and structural analysis, bypass, and scan-based attacks. We provide a mathematical model of DOSC-inserted design and a formal notion of its security. Our analysis shows that DOSC exponentially increases the resiliency against key extraction by SAT attack and its variants.
- We mathematically and experimentally demonstrate the optimum configuration parameters of DOSC (e.g., seed length, permutation rate, XOR gate placement policy throughout the scan chain, etc.) to achieve maximum security. Our analysis is similar to defining the length of the key for encryption algorithms to ensure their unbreakability in a reasonable amount of time. Our investigation concludes that attack complexity in a DOSC-inserted logic-locked design is maximized with the permutation rate of "1" and XOR gates placed uniformly throughout the scan chain. All our experiments are performed using advanced SAT solvers that correlate with our mathematical estimation of attack complexity with 95% accuracy.
- We have enhanced the shadow chain and obfuscated scan chain structure from previous work to achieve maximum security. This enhancement improved scan-based attack complexity to $O(3^N)$ in case of DOSC-inserted logic locked design.
- We implement an obfuscation tool flow that takes DOSC design parameters as input and automatically produces the DOSC IP with minimal area and test time overhead.
- DOSC facilitated complete design flow has been proposed in this article along with secure test flow (test generation and application) and post-test activation that ensures maximum security and fault coverage.

The rest of the article is organized as follows: Section 2 presents some preliminaries on logic obfuscation with static keys, their vulnerabilities, threat model, SAT attack exploiting scan chain, and limitations of existing SAT-resistant techniques. Section 3 describes the DOSC architecture, its operation, testability, and security establishment. Section 4 presents the mathematical model of DOSC-inserted design and the formal notion of security. Section 5 analyzes the security of DOSC-inserted design against SAT attack and experimental results based on the analysis. Section 6 represents results and discussions on how DOSC protects against other oracle-guided attacks. Finally, Section 7 concludes the article.

2 PRELIMINARIES

2.1 Logic Obfuscation: Static Obfuscation Key

Logic obfuscation aims at locking and, to some extent, concealing underlying hardware IPs to protect them against reverse engineering, cloning, and overproduction. The locking key gates and the

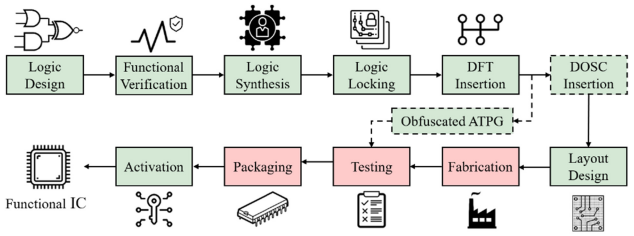

Fig. 2. Various stages of electronic hardware supply chain. Green boxes are trusted, and red boxes are untrusted.

keys are assumed to be chosen in such a way that an adversary cannot guess (or extract by other means) them within a practically feasible time. Without knowing the correct unlocking (secret) key, the correct functionality can never be retrieved. Once fabricated, packaged, and delivered to a trusted facility, the locked ICs are activated by burning the unlocking key into a tamper-proof memory. Existing schemes consider static obfuscation [28] where the unlocking key remains fixed throughout the lifetime of the chip. Over the years, several key recovery attacks have been proposed such as key-sensitization attack [13] and EPIC attack [29]. The former utilizes an automatic test pattern generation (ATPG) tool to propagate the effect of a key gate to a primary output based on the location of the key gates. EPIC attack [29] uses a hill-climbing search-based algorithm that monitors test responses to guess the secret key. In 2015, a SAT-based attack [17] was proposed that broke most of the combinational circuit logic obfuscation techniques within minutes. Concurrently, decamouflaging attack [30] was proposed that reverse-engineered camouflaged ICs within hours.

2.2 Threat Model

In this article, we assume that a trusted entity such as the design house performs logic design, verification, and synthesis based on a target technology library as shown in Figure 2. It also inserts additional key gates in the synthesized design to perform logic locking. Once the re-synthesis of obfuscated design is complete, DFT structures are integrated to improve testability. The design house generates the obfuscated physical layout and GDSII version of the netlist and sends the obfuscated GDSII to an offshore (untrusted) foundry for fabrication. After the fabrication, the die goes through wafer sorting, dicing, and manufacturing test. The manufacturing test is a structural test that does not check if the circuit's overall functionality is correct; instead, it tries to confirm that the individual building blocks (known as standard cells or logic gates) are working correctly. The assumption is that if the design is correct and structural test verifies the proper integration and functionality of individual gates; then the overall functionality should be held. Researchers have explored how manufacturing test can be done without using the correct chip unlocking key [63]. Therefore, leveraging the structural test definition, we propose post-test activation for DOSC-inserted logic-locked design to ensure maximum security and fault coverage [62]. Once the manufacturing test differentiates the defect-free ICs, they can be packaged and shipped to a trusted facility (e.g., IP owner) for activation and distribution. We adopt the typical assumptions for the SAT attack [17] where the attacker has access to the following:

(1) **Locked Netlist:** The attacker is either an untrusted foundry or an end user. In the former case, the locked, i.e., key gate inserted, gate-level netlist can be derived from the GDSII in the foundry's possession. For the latter, it is assumed that the netlist is generated from images of the layout captured from a de-processed chip.

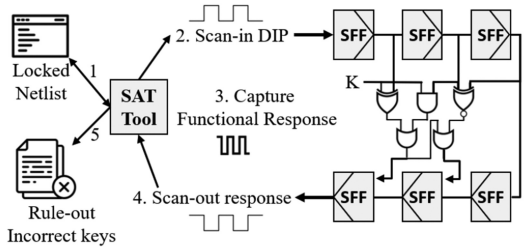

Fig. 3. SAT attack using scan chain for sequential design to extract logic locking key, K. SAT tool determines DIPs by applying input sequences in the locked netlist and verifies the key guesses by exploiting the scan chain of the oracle.

(2) **Unlocked IC:** At a minimum, one unlocked IC (i.e., oracle) can return correct outputs for any input pattern. Such an IC can be obtained from the open market, a rogue insider in the trusted supply chain, or from an on-field system.
(3) **Scan Chain:** Scan chain access is required to convert a sequential circuit into a combinational circuit, since the basic SAT attack only runs on combinational circuits.

Details on how testing is performed in a DOSC-inserted design is later discussed in Section 3 once DOSC architecture is introduced.

2.3 SAT Attack: Exploiting Scan Chain

DFT infrastructures transform the sequential elements in the design to a chain of shift registers (scan chain) to provide additional observability and controllability during the post-fabrication structural test. Figure 3 shows how the scan structure can be exploited to perform the SAT attack in a sequential design. The attacker performs these steps:

(1) The attacker derives a DIP for each combinational cone of the scan chain using the locked netlist and two key guesses.
(2) Next, the attacker runs the unlocked chip (oracle) in test mode and shifts the DIPs into the oracle.
(3) Then, the attacker switches to the functional mode and runs the circuit to capture functional response for the DIPs.
(4) The attacker switches back to test mode to scan out the obtained functional response.
(5) Comparing the output of steps 1 and 4, the SAT solver rules-out keys that produce incorrect output by adding a constraint to the SAT solver.
(6) Return to step 1 and repeat until no DIPs are found.

It has been shown that SAT attacks are much faster than traditional brute force attack, because each iteration typically rules out a large number of incorrect keys [17]. The efficiency of SAT attack can be expressed as [19]

$$T = \sum_{i=1}^{M} t_i, \tag{1}$$

where T is the total execution time, M refers to the total number of SAT iterations, and time t_i refers to the required time to solve ith iteration. M depends on the number of DIPs, length of key bits, and location of the key gates in the obfuscated netlist. t_i relies on the complexity of solving each iteration. SAT attack can be thwarted if either M or t_i is large enough to make the SAT attack infeasible within a reasonable length of time or with available resources. Since the scan chain

provides access to circuit internals, it makes the SAT attack extremely efficient. Massad et al. [48] showed that scan access provides high controllability and observability to the attacker. Having scan access, an attacker can run the sequential circuit to any desired state that differentiates the properties of the locking gates. Nevertheless, without scan access, an attacker must apply a set of input sequences that steer the sequential circuit to the target state. Massad et al. [48] have attempted to break camouflaged sequential circuits, which are conceptually similar to obfuscated sequential circuits, without scan access by using model checker NuSMV [52]. However, the authors were only successful for very small sequential benchmarks where the number of sequential elements was less than 250 because of the fact that model checker tool cannot handle large sequential circuits. As one can see, the DFT technique that helps test engineers for finding faults, also enables the attacker to convert a sequential design to a set of combinational cones and perform SAT attack. Researchers have recently proposed *Encrypt Flip-Flop* by obfuscating scan chains to prevent scan-based attacks [31] by inserting multiplexers at the end of registers and selecting Q or $\bar{Q}$ based on an obfuscation key. However, due to the obfuscation type being static (i.e., the key is fixed), such a scan obfuscated circuit can be modeled as a traditional logic locking problem, and the adversary can perform the SAT attack to reveal the secret [26]. An improved version of *Encrypt Flip-Flop* [57] has been proposed that dynamically changes the selection strategy of the MUXes by encrypting the scan enable input of some scan cells using a test key. A similar approach has been proposed in Reference [58], where a scan controller locks all the scan enable pins at a time using a scan access key. The difference between these two scan obfuscation techniques is that, improved *Encrypt Flip-Flop* [57] individually encrypts scan flip-flops while secure scan controller [58] locks all the scan enable pins of the scan flip-flops in the scan chain at a time. However, recently proposed DynUnlock [54] modeled the dynamic selection nature of scan flip-flops into a combinational circuit and recovered the secret test key. Similar attack could be possible in References [57, 58] as well. Furthermore, both of these techniques require the use of a comparator that weakens these methods to bypass attack [21].

2.4 Existing SAT-resistant Techniques and Their Limitations

To resist SAT attack, several SAT-resistant logic obfuscation techniques have been proposed, such as SARLock [18], Anti-SAT [19], SFLL [20], and SFLL-fault [55]. SARLock and Anti-SAT resist the SAT attack by increasing the number of required DIPs by exploiting a one-point function to flip the output of the design for each incorrect key. While these two techniques are strong enough to withstand the oracle-guided SAT attack, they are vulnerable to removal attack [23], bypass attack [21], SPS attack [22], and approximate SAT (AppSAT) attack [24] due to their low corruptibility; hence, they fail to provide the desired strength in functional obfuscation. Stripped-functionality logic locking (SFLL) [20] was proposed to withstand the aforementioned attacks by stripping some of the functionality of the original design and hiding it in the form of a secret key. Once the correct secret key is applied, the original functionality of the design is restored. However, the major limitation of SFLL is that it embeds the secret key or some properties of the key into the stripped functionality unit. Fall attack [25] exploited this vulnerability by performing structural and functional analyses of locked design to identify the locking key that is hardcoded in the stripping circuitry. In extension of Reference [20], SFLL-fault [55] was proposed to efficiently identify the functional cubes to be stripped from the original design and to automate this step by utilizing the fault modeling of the VLSI testing principles. However, an ATPG-guided fault injection attack [56] has been recently proposed, that can successfully recover the entire key within an attack complexity that grows linearly with key size. In addition to the functional logic locking that corrupts the design functionality for incorrect keys, several parametric locking methods have been proposed [61] that modifies the design parameters, e.g., power, performance, reliability, and

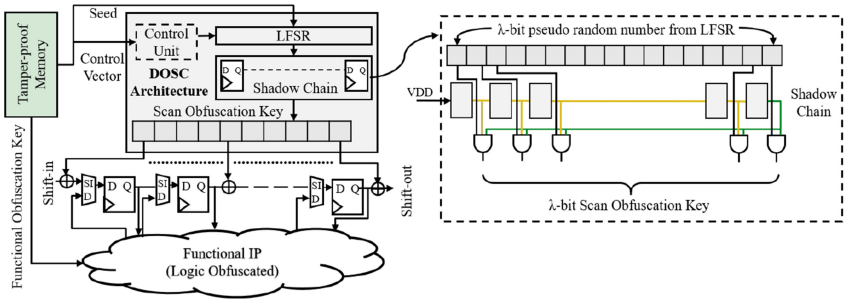

Fig. 4. DOSC-inserted in a functional logic locked IP. Both logic locking key and DOSC seed are protected in a tamper-proof memory.

so on, for incorrect key application. Nevertheless, these parametric locking methods are susceptible to FALL attack [25] and a variant of SAT (TimingSAT) [60] that utilizes timing profile of all the gates to embed into SAT formulation. Hence, there remains a need to develop a SAT-resistant logic obfuscation technique that can combat all the known attacks and new oracle-guided attacks that could arise.

3 DYNAMICALLY OBFUSCATED SCAN CHAIN

3.1 DOSC Architecture

A high-level overview of the DOSC architecture, inserted in the scan chain of a logic-obfuscated functional IP, is shown in Figure 4. The logic obfuscation of functional IP can be done by existing logic obfuscation schemes [4, 7, 12, 32]. However, the main security advantage comes from the DOSC architecture itself, which obfuscates the values extracted from scan chains. The DOSC architecture [27] is composed of three major components:

(1) Linear feedback shift register (LFSR): In our proposed DOSC architecture, the scan obfuscation key changes randomly. A polynomial primitive LFSR is adopted to generate λ-bit pseudo-random numbers, which are later passed through the shadow chain (to be discussed next) and connected to the scan chain by XOR gates to one-time pad (OTP) scan chain contents. The LFSR reads control signals from the control unit to generate pseudo-random numbers. The seed of the LFSR is kept in a secured non-volatile tamper-proof memory along with the functional obfuscation key. The LFSR clock and scan clock are related by the permutation rate,

$$\alpha = \frac{Scan\ Clock\ Frequency}{LFSR\ Clock\ Frequency}. \tag{2}$$

For example, a permutation rate of $\alpha = 4$ means that LFSR generates pseudo-random numbers once in every four scan clock cycles. The definition of permutation rate has been changed from our previous work [27]. In the proposed DOSC structure, we take $\alpha = 1$, i.e., LFSR is driven by the same clock frequency as that of the shadow chain and scan chain. This configuration ensures maximum security as we shall discuss in Section 4. It should be noted that, in case of an XOR feedback, a seed of all zeros is forbidden as the LFSR would remain in a locked-up state and continues providing all zero scan obfuscation keys. However, an LFSR with XNOR feedback would have a forbidden state of all ones. An LFSR with XOR feedback can be tampered with and forced to be stuck into a forbidden state by lowering the power supply during the scan mode when the seed is being loaded. To mitigate this issue, DOSC uses XNOR feedback-based LFSR.

(2) Shadow Chain: The shadow chain protects LFSR generated pseudo-random scan obfuscation keys from ScanSAT [26] attack (detail discussed in Section 6), and other oracle-guided

scan-based attacks. The length of the shadow chain is the same as the LFSR. It takes the λ-bit pseudo-random number generated by the LFSR as input and generates a λ-bit scan obfuscation key. Shadow chain consists of a chain of flip-flops, driven by the scan clock, with the first flip-flop input connected to logic "1." The flip-flops' outputs are connected to the XOR gates inserted into the scan chain through a series of AND gates. These are three-input AND gates where one of the input is coming from the last flip-flop of the shift registers of the shadow chain (the green interconnect in Figure 4). Upon reset or when a new seed is loaded, at first, all the flip-flops in the shadow chain are reset and forced to logic "1" serially (the yellow interconnects in Figure 4) with scan clock frequency. When the last flip-flop of the shadow chain becomes "1" at the λth clock cycle after reset or seed being loaded, only then is the scan obfuscation key applied to the XOR gates and scrambled responses start showing at the scan-out port. The shadow chain structure has been modified since our previous work [27] (as shown in Figure 4) to enhance the scan chain blocking capability.

(3) Obfuscated Scan Chain: Obfuscated scan chain is the scan chain of the (logic locked) functional IP with λ number of XOR gates placed throughout the chain. In enhancement to our previous work [27], here we propose XOR gates to be placed uniformly throughout the scan chain to ensure maximum security against SAT attack (the proof behind uniform placement is presented in Section 4.2). One of the inputs of the XOR gates comes from the λ-bit scan obfuscation key while the other input comes from the scan chain. When the scan obfuscation key is applied to the scan chain, the XOR gates OTP scan chain contents.

A control unit reads a control vector from the tamper-proof memory in addition to the above three components. The control unit's role has been enhanced from Reference [27] to comply with the threat model of logic obfuscation for functional logic-locked IP and DOSC seed. The control unit of DOSC controls memory loading, LFSR activity (based on the applied control vector and permutation rate), resets shadow chain during a system reset, or seed is loaded.

Requirements for Inserting DOSC: There are two requirements for using DOSC to resist SAT attacks and supply chain threats: (i) the target design needs to be sequential with scan-chain inserted within it (this is always the case for any large-scale functional design); (ii) the combinational circuit of the design needs to be logic locked by some logic locking scheme.

DOSC Security Operation: During functional mode, DOSC is turned off. When switched to the test mode for scan access (a requirement for practical SAT attack), DOSC starts up in the following steps: (1) DOSC powers up along with scan obfuscation circuitry, and the seed from the tamper-proof non-volatile memory is loaded into the LFSR. (2) After the seed is loaded, the LFSR starts generating pseudo-random numbers. (3) Individual flip-flop outputs within the shadow chain start becoming logic "1" one by one in each clock cycle. Any intended scan-in pattern during the first λ clock cycles after reset or seed being loaded will result in λ-bit zeros at the scan out. This protects against reset attack [33], which extracts secrets by exploiting the initial reset condition of sequential elements. After λ clock cycles, flip-flops of the shadow chain will settle down, and any scanned-in pattern will get obfuscated by the scan obfuscation key. Now, if the attacker tries flushing the scan chain, the scan output is shuffled and they are the original or inverted response. This way, the shadow chain protects both the scan obfuscation key as well as unscrambled data from leaking through the scan out. Further, as XOR gates of DOSC are inserted into the scan chain, DOSC is not prone to bypass attack either (details in Section 5-F).

Testability of DOSC-integrated Chip: DOSC establishes a secure test, along with preserving the required test coverage [27]. In development from the previous work [27], in this subsection, we propose a comprehensive method of manufacturing test pattern generation (and transformation to their obfuscated forms) using dummy seed and functional obfuscation key. Manufacturing test

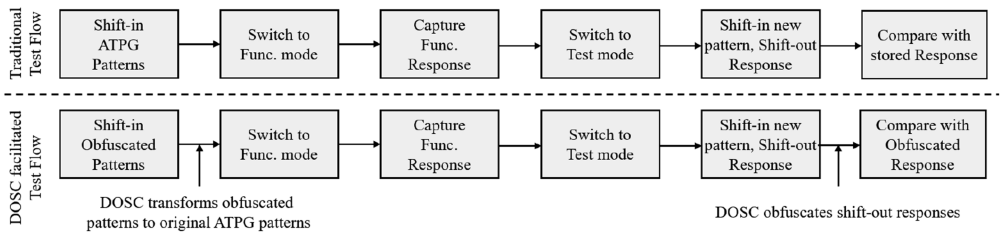

Fig. 5. Test flow in a DOSC vs. traditional design.

patterns can be generated using a set of dummy seed (for DOSC) and key values (for functional logic locked IP) loaded to the on-chip tamper-proof memory [62]. Functional obfuscation keys can be loaded into key registers (that are part of the scan chain [63]) from the tamper-proof memory during power on. The scan enable pin is utilized to select between functional obfuscation key and scan input [63]. Furthermore, scan chain being locked by DOSC will resist any scan-based attack on functional obfuscation keys as discussed in Section 6. Dummy seed and key values are different than the original unlocking seed and functional obfuscation keys. The purpose of dummy seed and key values are to perform manufacturing test in an untrusted facility without exposing the original unlocking seed and functional obfuscation keys. The original seed can be made unique from chip to chip, however, the functional obfuscation key has to be same for all the chips. A step-by-step test flow in a DOSC facilitated design is shown in Figure 5 along with a traditional test flow. As shown in Figure 5, test engineers must apply obfuscated test patterns. Knowing scan obfuscation keys for each test cycle and the XOR gates' location throughout the scan chain, test engineers can generate obfuscated test patterns. While the test engineer shifts in these obfuscated patterns, DOSC transforms them into original ATPG patterns before switching to functional mode. Correspondingly, when the test engineer shifts-out captured functional response, DOSC obfuscates shift-out responses, as shown in Figure 5. The test engineer receives obfuscated ATPG responses from the designer and compares them with in-field responses to identify defective parts. In the case of designs with scan compression, compressed test patterns are first decompressed using the decompressor transfer function and then transformed into obfuscated patterns. Similarly, captured responses are compressed first by utilizing the compressor transfer function before obfuscating them using scan obfuscation keys generated by DOSC. The ATPG pattern and response conversion can be done offline at any trusted facility. The test pattern and response conversion can only be done correctly if the LFSR seed is known, along with the exact architecture and XOR placement of DOSC. Therefore, an adversary who has no access to the seed stored in a tamper-proof memory and is trying to obtain the seed and functional obfuscation keys will have no means to perform such conversions and perform the SAT attack. A secure test flow considering test compression, their associated pattern, and response transformation is a part of our on-going research.

3.2 Dynamic Scan Obfuscation: Resistance to SAT Attack

In the case of dynamic obfuscation (i.e., DOSC), the scan chain obfuscation key changes periodically making it more difficult for the attacker to retrieve the design's static obfuscation key. Despite only trying to make logic obfuscation resilient to SAT attack, an effort that has been proven unsuccessful thus far, restricting access to the oracle through scan chain will resist oracle-guided SAT attacks as well as other scan-based attacks. Our initial scheme [27] was primarily developed to protect scan infrastructure from scan-based attacks, which aim to extract crypto keys exploiting the scan chain. Note that there exist fundamental differences in the threat model for scan attacks and the oracle attacks on obfuscated logic. In a scan attack, it is normally assumed that the attacker

does not have detailed knowledge of the architecture of the scan chain. However, in the logic locking threat model, the adversary has the detailed knowledge of the scan architecture, since he/she has access to the entire locked netlist. Therefore, any protection mechanism incorporated in the scan architecture would also be visible to the attacker. Hence, none of the existing scan protection techniques can directly address logic locking attacks using scan. Thus, we propose to employ the new DOSC architecture to prevent oracle-guided SAT attacks on a locked netlist.

We consider two types of obfuscation here: (i) Functional obfuscation: Logic locking of the functional circuitry of the IP; (ii) Scan obfuscation: Obfuscating the scan chain of the logic locked IP. The critical advantages of dynamically obfuscating scan chain are as follows.

(1) SAT attack requires the obfuscation key to be static in the duration of applying DIPs and retrieving the output responses. However, in DOSC, the scan obfuscation key changes in every or every few cycles. Therefore, when the DIPs are shifted in or the responses are shifted out, the scan obfuscation key has already been changed multiple times and altered scan inputs and/or responses.
(2) In case of dynamic obfuscation, the SAT tool needs to solve the circuitry (DOSC circuitry in our case) that generates the dynamic (pseudo-random in our case) obfuscation keys. The complexity of solving the key generator circuit to retrieve the seed by observing the scan out responses is extremely high.

If solving scan obfuscation circuitry can be designed in a way that the scan obfuscation key changes before the SAT solver reaches satisfiability, then the SAT attack complexity increases drastically, and the attack can be resisted. While dynamic obfuscation of scan chain is feasible, since changing scan obfuscation key scrambles just the shift-in/shift-out scan patterns only during test mode and they can be translated back to the original values off-line, it is not practical for functional obfuscation as changing functional obfuscation key changes the functionality of the entire design.

4 DETAILED SECURITY ANALYSIS OF DOSC

In this section, we perform mathematical modeling and security analysis of DOSC-inserted logic-obfuscated designs. At first, we present a functional model showing the scrambling operation performed by DOSC in the scan chain. Next, we estimate the SAT attack complexity of a DOSC-inserted logic-obfuscated design for different sizes/parameters of DOSC.

4.1 Functioning Model of DOSC-inserted Design

DOSC scrambles scan chain contents using an LFSR. An LFSR of length λ is a finite-state automaton that generates a semi-infinite sequence of elements of $B = \{0, 1\}$ with a linear recursive relation of degree λ. The output of the LFSR, that left shifts the internal states of the LFSR in the next cycle, depends on the current internal states of the LFSR and the function of the feedback path, as shown in Figure 6. Let us assume an LFSR of length λ with internal states $l = \{0, 1\}^{\lambda}$. The output of the LFSR at any time t can be represented by the following equation:

$$l^{t+1}[0] = \sum_{i=0}^{\lambda-1} c_i l^t[i] \ mod\ 2, \tag{3}$$

where $c_0, c_1, \ldots, c_{\lambda-1}$ are the feedback coefficients. The output at any time t is a pseudo-random number comprised of all the elements of internal states.

Now, let us consider a combinational logic-obfuscated circuit that implements a Boolean function $F_{locked} : \{I, K\} \rightarrow O_{locked}$, where input space $I = \{0, 1\}^m$ with m inputs, output space $O = \{0, 1\}^n$ with n outputs, and key space $K = \{0, 1\}^p$ with p key inputs. The locked circuit gets acti-

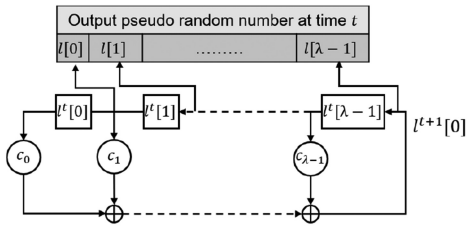

Fig. 6. LFSR. At each cycle internal states are shifted left while right-most bit is realized by feedback function.

Table 1. List of Notations for Mathematical Models

Notation	Description
t	Clock cycle number
N	Length of scan chain
λ	Length of DOSC
α	Permutation rate of DOSC
i	Scan cell position in the N-bit scan chain
k	Number of XORing for i^{th} scan cell
b	Spacing of XOR gates throughout the chain
$SI^t[i]$	Scan-in bit to i^{th} scan cell at t cycle
$(SI^t[i])'$	Scrambled pattern of $SI^t[i]$
$SO^t[i]$	Scan-out bit from i^{th} scan cell at t cycle
$(SO^t[i])'$	Scrambled response of $SO^t[i]$
$l^t[i]$	i^{th} bit of LFSR pattern at t cycle
γ	Number of control signals into LFSR
d	Ratio of number of variables to number of clauses

vated only when the correct unlocking key K_f is applied, $\exists\, K_f\ \forall I\ F_{activated} : \{I, K_f\} \rightarrow O_{locked} = O_{activated}$. However, in case of a sequential logic-obfuscated circuit, the output depends on the applied obfuscation key and the internal states along with the primary inputs, $F_{locked} : \{I, SI, K\} \rightarrow O_{locked}$, where internal state space, $SI = \{0,1\}^N$ for N number of sequential elements. Internal states are considered as pseudo-primary inputs, which can be accessed via scan chain [17]. Table 1 defines different notations used in the models.

To secure logic-obfuscated design against oracle-guided attacks that exploit the scan chain, we insert a λ-bit DOSC in the N-bit scan chain of the logic-obfuscated circuit, as shown in Figure 7(i). For any reasonable size design, the number of scan chain is much larger than the length of the LFSR ($N \gg \lambda$). Therefore, we *distribute* λ key gates from DOSC throughout the N-bit scan chain with a spacing of b. ATPG patterns in a DOSC-inserted chip goes through the following transformations.

Scan Shift-in: During scan testing, the internal states are first reset, and the chip is powered on in scan mode, which activates DOSC. Then the input patterns generated by an ATPG tool (e.g., Synopsys TetraMAX) are shifted into the scan chain through the scan-in port in a reverse order to clock cycle. For example, the scan-in bit for the last scan flip-flop is shifted into the scan-in port in the first cycle ($SI^0[N-1]$) and the rest of the scan-in bits are shifted serially in descending order. These input patterns are scrambled by the DOSC-generated dynamic scan obfuscation keys of the corresponding cycle before reaching their target scan flip-flops. The scrambling operation of DOSC is determined by (i) the location of the scan-in bit in the scan chain; (ii) the position of the

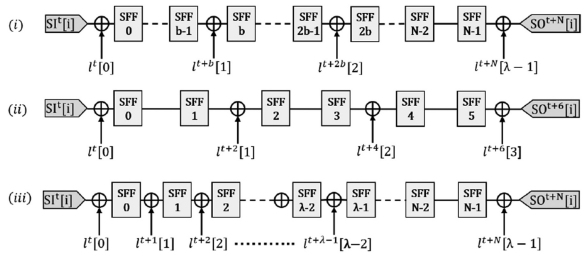

Fig. 7. Different spacing of DOSC XOR gates in the scan chain for, (i) spacing = b, (ii) example for $N = 6$, $\lambda = 4$, $b = 2$., and (iii) spacing = 1.

Table 2. Functional Model of DOSC Scrambling Operation with Example for $N = 6, \lambda = 4,$ and $b = 2$

	Scan Shift-in	**Functional Capture**	**Scan Shift-out**
Scan-in	Scrambled Scan-in	Capture	Scrambled Scan-out
$SI^{N-1-i}[i]$	$(SI^{N-1}[i])'$	$SO^{N}[i]$	$(SO^{2N-i}[i])'$
$SI^0[5]$	$(SI^5[5])' = SI^0[5] \oplus l^0[0] \oplus l^2[1] \oplus l^4[2]$	$SO^6[5]$	$(SO^7[5])' = SO^6[5] \oplus l^7[3]$
$SI^1[4]$	$(SI^5[4])' = SI^1[4] \oplus l^1[0] \oplus l^3[1] \oplus l^5[2]$	$SO^6[4]$	$(SO^8[4])' = SO^6[4] \oplus l^8[3]$
$SI^2[3]$	$(SI^5[3])' = SI^2[3] \oplus l^2[0] \oplus l^4[1]$	$SO^6[3]$	$(SO^9[3])' = SO^6[3] \oplus l^7[2] \oplus l^9[3]$
$SI^3[2]$	$(SI^5[2])' = SI^3[2] \oplus l^3[0] \oplus l^5[1]$	$SO^6[2]$	$(SO^{10}[2])' = SO^6[2] \oplus l^8[2] \oplus l^{10}[3]$
$SI^4[1]$	$(SI^5[1])' = SI^4[1] \oplus l^4[0]$	$SO^6[1]$	$(SO^{11}[1])' = SO^6[1] \oplus l^7[1] \oplus l^9[2] \oplus l^{11}[3]$
$SI^5[0]$	$(SI^5[0])' = SI^5[0] \oplus l^5[0]$	$SO^6[0]$	$(SO^{12}[0])' = SO^6[0] \oplus l^8[1] \oplus l^{10}[2] \oplus l^{12}[3]$

DOSC XOR gates in the scan chain, which is determined by spacing b; and (iii) the permutation rate α. The first two parameters control how many times scan-in bit for i^{th} scan-cell will be XOR'ed by DOSC-generated dynamic scan obfuscation key bits of which cycle. Figure 7(i) shows a scan chain with DOSC XOR gates placed for spacing b. Note that the scan chain values get XOR'ed more and more when shifted toward the later flip-flops in the chain. The permutation rate α along with DOSC seed decide scan obfuscation keys applied to the scan chain during each scan cycle. Therefore, the permutation rate $\alpha = 1$ produces the maximum obfuscation in scan-chain values. In this article, we consider permutation rate $\alpha = 1$.

The scrambled version of the scan-in patterns can be determined using the following equation:

$$(SI^{N-1}[i])' = SI^{N-1-i}[i] \oplus x[i],\ 0 \le i \le N-1, \tag{4}$$

$$x[i] = \begin{cases} \bigoplus_{k=0}^{\lambda-2} l^{N-1-i+k.b}[k], & \text{if } i \ge b(\lambda-2], \\ \bigoplus_{k=0}^{\lfloor i/b \rfloor} l^{N-1-i+k.b}[k], & \text{otherwise,} \end{cases} \tag{5}$$

where $k.b$ is the arithmetic product of spacing b and loop variable k. Note that all the scan-in bits appear at the input of their target scan flip-flop in the same cycle. Table 2 shows how scan patterns are transformed in the scan chain for an example of $N = 6, \lambda = 4$, and $b = 2$ (shown in Figure 7(ii)).

Functional Capture: Once scan-in patterns are shifted into the scan chain, the chip is switched to the functional mode and run for one cycle to capture the functional response. DOSC is disabled during the functional mode. At the end of the functional cycle, the scan chain is filled with responses generated by the circuit running in the functional mode, which can be represented by

$$SO^N = F_{locked}\left(I, \left(SI^N\right)', K_f\right). \tag{6}$$

Note that, due to the presence of DOSC, functional responses are realized by the scrambled version of the scan-in patterns. The third column (Functional Capture) of the Table 2 shows the functional response for the scrambled scan-in patterns. Nevertheless, the purpose of expression 6 is to portray how DOSC obfuscates the captured responses. Therefore, we kept corruption due to logic obfuscation inactive by applying the correct unlocking key, K_f. In a practical scenario, the unlocking key K_f being secret to the attacker will be replaced by K in expression 6 and make the captured responses even more obscure.

Scan Shift-out: At this step, the chip is switched back to the scan mode (where DOSC is re-activated) to shift-out captured responses serially through the scan-out port. Scan chain contents get scrambled by DOSC-generated scan obfuscation key bits. This scrambling process is similarly dependent on the location of the response in the scan chain, b, as well as α. In contrast to scan-in, the scan-out bits that are in later flip flops with be XOR'ed less times than those in earlier flip flops. This scrambling can be represented mathematically as follows:

$$(SO^{2N-i}[i])' = SO^{N}[i] \oplus y[i],\ 0 \leq i \leq N-1, \tag{7}$$

$$y[i] = \begin{cases} \bigoplus_{k=\lfloor i/b \rfloor+1}^{\lambda-2} l^{N+k.b-i}[k], \text{ if } i < b(\lambda-2), \\ l^{2N-i}[\lambda-1], \text{ otherwise.} \end{cases} \tag{8}$$

Note that the last scan flip-flop content comes out first through the scan-out port and rest of the contents keep coming serially. An example of scrambling process of captured functional response during shifting out is shown in Table 2. This scrambling process by DOSC's scan obfuscation keys generated removes direct correlation between scan-in and scan-out patterns, which is clearly visible in Table 2.

Notes on Fault Testing and Diagnosis: Inclusion of DOSC changes actual ATPG patterns (SI^t) to scrambled patterns $(SI^t)'$ before applying to the functional circuit. To apply the actual ATPG pattern in the functional circuit in the presence of DOSC, the test engineer needs to shift-in $(SI^t)'$ through the scan-in port. Knowing the seed and location of XOR gates in the chain, the trusted test engineer can translate SI^t to $(SI^t)'$ utilizing the equations in Table 2 and apply them to the design. Similarly, utilizing the inversion property of XOR function, DOSC generated scan obfuscation keys can be used to de-obfuscate the initially shifted pattern $(SI^t)'$ to SI^t before switching to the functional mode and applying to the functional circuit. Similarly, the designer can do the same for responses during scan-out.

Without knowing the seed, the attacker can never identify the $(SI^t)'$ that needs to be shifted to the scan-in port that will eventually be translated to the original (SI^t) before switching to the functional mode, and same for scan responses. Therefore, the attacker cannot access any meaningful data through the scan ports to be able to perform SAT-based attacks and extract the functional obfuscation key.

4.2 Key Gate Insertion Technique

From Equations (4)–(8), it can be realized that DOSC scrambling process greatly depends on the XOR gate locations. To ensure maximum confusion in the scan chain contents, we consider the following constraints for key gate insertion with spacing b.

- One key gate needs to be placed at the beginning and another at the end of the scan chain; otherwise, some of the scan-in bits and output responses will remain unscrambled. Furthermore, if two key gates are placed at the farthest ends of the scan chain, during the SAT attack, the attacker needs to identify scan obfuscation key for $(2N-1)$ cycles after scan

obfuscation key is applied, which forces the attacker to unroll DOSC for $(2N - 1 + \lambda)$ times (λ cycles for shadow chain). The greater number of cycles the attacker has to unroll DOSC, the more complex the SAT attack will become.

- The rest of the $(\lambda - 2)$ key gates need to be distributed throughout the chain in such a way that maximizes scrambling. In Figure 7(iii), two XOR gates are placed at the farthest ends of the chain, and the rest of the $(\lambda - 2)$ XOR gates are placed at the initial side of the chain with a spacing, $b = 1$. Based on Equation (4), LFSR-generated patterns of consecutive cycles will be scrambling the scan chain contents for $b = 1$. Furthermore, in this case, scan out responses ($SO[\lambda - 1 : N - 1]$) will be XOR'ed only once for $i \geq (\lambda - 1)$, thereby reduces scrambling. Similarly, placing these $\lambda - 2$ gates with spacing of $b = 1$ at the end of the chain will reduce scrambling in the scan-in process. Increasing spacing results LFSR-generated patterns performing XOR operations on the scan chain contents to be b cycles apart from each other, which increases randomness among them. However, spacing among the XOR gates needs to be restricted to accommodate all the λ XOR gates inside the chain. By setting $i = N - b$ in the condition of Equations (4) and (8), we can ensure maximum scrambling for both scan-in patterns and scan-out responses. Solving for $i = N - b$, we get optimum spacing between DOSC XOR gates in terms of maximum scrambling:

$$b = \left\lfloor \frac{N}{\lambda - 1} \right\rfloor. \tag{9}$$

  We have performed experiments on how SAT attack complexity varies due to change in crucial DOSC gate placement. Results show that SAT attack complexity reaches the maximum when two key gates are placed in the two corners of the chain, and the rest of the key gates are distributed uniformly (see Section 5.5).

4.3 SAT Attack Complexity

We have developed a generic model of the mathematical complexity introduced by DOSC in terms of Boolean satisfiability.

SAT Problem: The Boolean satisfiability problem can be represented by propositional variables in the Galois Field of two ($GF(2)$), which are either true or false [36]. The properties of propositional logic are as follows

- A propositional variable or the negation of a propositional is known as a *literal*.
- A *propositional formula* is a combination of propositional variables related by only three logical operators: *AND, OR,* and *negation*.
- A *clause* is a disjunction (OR) of literals.
- In propositional logic, any algebraic equation can be represented as a conjunction (AND) of clauses, which is called *conjunctive normal form (CNF)*.

The SAT problem is the process of determining whether all the propositional variables in the propositional formula can be assigned values such that the propositional formula itself satisfies to be true. In the case of the SAT attack on the logic-obfuscated circuit, the SAT solver tool, such as [17], transforms the miter circuit to CNF and tries to achieve satisfiability by assigning values to the key inputs. If the CNF of the miter circuit can be assigned such a key value that the output of both locked netlist and unlocked chip matches for any input pattern, then the formula is satisfied (i.e., all the clauses of the CNF are satisfied). The difficulty of finding the key of a logic-obfuscated circuit directly depends on the complexity of solving CNF of the miter circuit developed. Hence, the difficulty of finding the seed of DOSC using a SAT solver relies on the mathematical complexity of the CNF representation of DOSC. *The complexity of a CNF-SAT problem depends on the number*

of clauses, the total length of all the clauses (sum of the literals of all the clauses), the complexity of each clause, and the number of variables [37].

Converting DOSC to CNF: To find the initial seed of DOSC, a SAT solver needs to transform the DOSC circuit to its CNF expression with internal states of LFSR being λ, propositional variables $l^t[0], l^t[1], \ldots, l^t[\lambda-1]$ (where t means time cycle) and solve. Once all the flip-flop outputs of the shadow chain become "1," the CNF of DOSC consists of the CNF expression of the λ-bit LFSR and the XOR chain. The *characteristic polynomial* of a λ-bit LFSR can be represented as following using the feedback coefficients $c_0, c_1, \ldots, c_{\lambda-1}$,

$$P(l) = 1 + c_0 l^t[0] + c_1 l^t[1] + \cdots + c_{\lambda-1} l^t[\lambda-1]. \tag{10}$$

Equation (10) is a monic polynomial that can be represented by a Frobenius companion matrix [38]. Therefore, the states of the LFSR can be represented using vectors of bits along with the companion matrix of the characteristic polynomial (Equation (10)), and can be computed in $GF(2)$, disregarding the polynomial equation:

$$\begin{bmatrix} l^{t+1}[0] \\ l^{t+1}[1] \\ l^{t+1}[2] \\ . \\ . \\ l^{t+1}[\lambda-1] \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 & . & . & c_{\lambda-1} \\ 1 & 0 & 0 & . & . & 0 \\ 0 & 1 & 0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} l^t[0] \\ l^t[1] \\ l^t[2] \\ . \\ . \\ l^t[\lambda-1] \end{bmatrix}. \tag{11}$$

The companion matrix of Equation (11) is a square matrix of size $\lambda \times \lambda$, where λ is the degree of the LFSR characteristic polynomial shown in Equation (10). We have the following conditions in the companion matrix shown in Equation (11).

- The sub-diagonal elements are all ones to perform left shift in the current internal state of the LFSR.
- The topmost row contains the coefficients of the characteristic polynomial to determine the feedback path.

The companion matrix transforms the current state of LFSR to the next state. This way, the next state of LFSR can be represented by a set of linear equations of the current state (which is the seed at the initial cycle) set of literals and added to the CNF formula of the functionally obfuscated design. By carefully analyzing the number of added clauses, literals, and the length of clauses, we can accurately estimate the resiliency added to our design against SAT attacks.

A linear equation comprising of m monomials, generates $2^{(m-1)}$ number of clauses and $m.2^{(m-1)}$ number of literals when converted to a CNF [37]. Not all the internal states contribute to the feedback polynomial, as one can see. The location of the feedback taps defines the period of the LFSR. To design a maximum length LFSR, we choose feedback taps such that the corresponding feedback polynomial is primitive. This means the following conditions are met: (i) the number of taps is even; (ii) the set of taps is set-wise co-prime, i.e., there is no common divisor other than 1 to all taps. With this choice, once the λ-bit LFSR is transformed to CNF, the contribution from the set of linear equations of states from Equation (11) in the overall CNF is of the following three types:

- If the number of feedback taps is β, then the feedback equation derived from the top row of the companion matrix realizes $(\beta-1)$ XOR gates, which generates $4(\beta-1)$ number of clauses and $12(\beta-1)$ number of literals in the CNF according to *Tseytin transformation* [40]. The significance of the multiplication constants 4 and 12 comes from the fact that a Boolean XOR operation generates 4 clauses of 3 literals each, once translated to a CNF sub-expression.

Table 3. Mathematical Complexity of DOSC

	Number of Clauses, C	Number of Literals, L	Number of Variables, V
LFSR	$\{4(\beta-1)+(\lambda-1)2^{(\gamma-1)}\}(2N-1)$	$\{12(\beta-1)+\gamma(\lambda-1)2^{(\gamma-1)}\}(2N-1)$	$\lambda+(2N-1)2+2\lambda(2N-1)$
XOR	$4N\lambda$	$12N\lambda$	$N\lambda$
Output	$N+1$	$N+1$	
Total	$\{4(\beta-1)+(\lambda-1)2^{\gamma}+4\lambda+1\}N$	$\{12(\beta-1)+\gamma(\lambda-1)2^{\gamma}+12\lambda+1\}N$	$\lambda+(2N-1)2+2\lambda(2N-1)+N\lambda$

- If the LFSR consists γ number of control signals, then each of the $(\lambda-1)$ equations derived from the lower $(\lambda-1)$ rows of the companion matrix (11) will contain γ number of monomials. Therefore, these $(\lambda-1)$ equations will generate in total $\{(\lambda-1).2^{(\gamma-1)}\}$ clauses and $\gamma(\lambda-1).2^{(\gamma-1)}$ literals in the CNF.
- The number of variables in the CNF expression depends on the number of input/output ports of the circuit, and internal variables due to unrolling. The I/O ports in a LFSR consists of seed, control signals, and outputs. Unrolling of DOSC introduces new internal variables in the CNF sub-expression, because unrolling a sequential circuit generates pseudo-primary input and pseudo-primary output. The pseudo-primary output of one-time frame drives the pseudo-primary input of the next time frame [41].

Therefore, the total number of clauses in CNF expression generated by a λ-bit LFSR with β feedback taps equals $4(\beta-1)+(\lambda-1).2^{(\gamma-1)}$. We again take the example where a λ-bit DOSC is inserted into an N-bit scan chain of a logic-obfuscated circuit through λ number of XOR gates. The actual functionality of the design is irrelevant here, since the attacker needs to break the DOSC architecture first. If an attacker attempts to perform the SAT attack on this DOSC-inserted design, then he/she needs to apply DIPs in N-bit scan chain and observe scan-out vectors in each SAT iteration. Both scan-in and scan-out vectors being scrambled by dynamic scan obfuscation keys must be de-scrambled to figure out a correlation between applied DIPs and collected responses to ultimately extract the key of functional obfuscation. From the scan-out response of the functional model of DOSC-inserted design presented in Table 2, it can be noticed that, to de-scramble scan-in and scan-out response of an N-bit scan chain, one needs to reveal the scan-obfuscation key for $(2N-1+\lambda)$ number of cycles. This requires unrolling of λ-bit LFSR for $(2N-1)$ number of stages. Therefore, the total number of clauses from LFSR will be multiplied by $(2N-1)$ for an obfuscated circuit with an N-bit scan chain. DOSC is inserted into the scan chain through a series of XOR gates, which also contribute clauses and literals in the CNF expression. In scan mode, each of the scan flip-flops in the scan chain is treated as pseudo-primary input and output, which generates one literal clause in the CNF.

The total number of clauses and literals contributed by DOSC in the CNF expression are summarized in the second and third column of Table 3 with an approximated total shown in the last row. Identifying the number of variables in the CNF expression is a bit tricky, since all the variables do not scale due to unrolling. The variables in the CNF expression that come from seed (λ) are shared by all the unrolled time frame model of DOSC. The control inputs, pseudo-primary inputs, and pseudo-primary outputs of unrolled LFSR of DOSC are scaled by $(2N-1)$ due to unrolling. Lastly, each of the DOSC XOR gates create one additional variable, according to Tseytin transformation [40]. The total number of variables generated by DOSC in the CNF expression is also shown in the last column in Table 3. Note that, these mathematical equations are based on permutation rate $\alpha=1$. For changing the permutation rate or XOR gate position, the number of times DOSC needs to be unrolled will change. Furthermore, it is noteworthy that the complexity introduced by the functional obfuscated circuit will be added to the complexity introduced by DOSC circuitry.

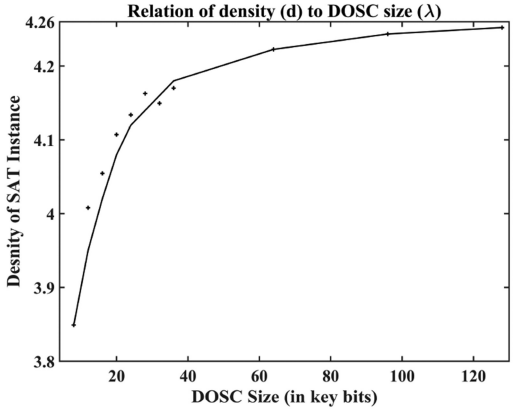

Fig. 8. Relation of the density of DOSC-inserted benchmark's SAT instance to DOSC key size.

Relation of SAT solvers/CNF Features to Time Complexity: Formulating the complexity of a SAT problem is a fundamental open question. However, researchers have found that the computational complexity of a SAT problem depends on the density of the combinatorial problem, the order of the problem (V), machine resources, the solver type, and solver heuristics [39]. The order represents the number of variables (V), and the density (d) represents the ratio of the number of clauses to the number of variables. Coarfa et al. [39] investigated the complexity of 3-SAT instance for varying density, order, and solver experimentally. Based on their investigation, the running time of 3-CNF SAT problem shows following phase transitions:

(1) *GRASP solver:* $t = O(2^{V\epsilon})$, where ϵ increases for $3.8 > d < 4.26$ and decreases for $d > 4.26$,
(2) *CPLEX solver:* $t = O(2^{V\epsilon})$, for $d > 4.0$,
(3) *CUDD solver:* $t = O(2^{V\epsilon})$, for $d > 0.5$ and ϵ is independent of density for $d > 2$.

From Table 3 it can be noticed that for a $\gamma = 3$, the ratio of the number of literals (L) to the number of clauses (C) is very close to 3, which makes SAT attack on DOSC-inserted logic-obfuscated design an NP-complete 3-CNF satisfiability problem [42]. *CUDD* solver is based on reduced ordered binary decision diagram (ROBDD) that are effective for hardware verification [39]. Experimental results in Section 5 of this article are based on modern SAT solvers like CUDD, MiniSAT [51], CryptoMinisat [50] and lingeling [49]. In Figure 8, we have shown how density (d) of DOSC-inserted design varies with changing DOSC size (λ). From the above explanation and Figure 8, it is clear that the complexity of the SAT attack on DOSC-inserted design always falls in the exponential region of running time, $O(2^{V\epsilon})$. The designer should choose DOSC architecture parameters, e.g., size of DOSC (λ), permutation rate (α), and XOR gate locations such that the attack complexity always grows exponentially. However, the actual time depends on machine resources and solver heuristics. In Sections 5.2 and 5.3, we show that the effect of DOSC architecture increases the SAT attack complexity and this effect is correlated with the CNF encoding introduced in this section (as shown in experimental results).

5 SAT ATTACK ANALYSIS AND RESULTS

In this section, we evaluate DOSC-inserted functional obfuscated design against different SAT attacks and present experimental results based on the analysis. Table 4 shows the statistics of the ISCAS'89 [34] and ITC'99 [35] benchmarks utilized for the experiments. All attacks were performed on a server with 32 core 2.6 GHz CPU and 64 GB memory; 3% of the resources were utilized

Table 4. Statistics of the Benchmarks Used for Experimental Validation

Benchmark	Inputs	Outputs	Gate count	Scan chain
s838	34	1	446	32-bit
b07	1	8	441	49-bit
s1423	17	5	657	74-bit
b14	32	54	10098	245-bit
s38417	28	106	22179	1636-bit

in the attack. For our experiments, we have used open-source SAT attack tool on logic locking [17] that is based on modern SAT solvers, e.g., MiniSAT [51], CryptoMinisat [50], and lingeling [49].

5.1 SAT Attack on LFSR Seed

In this attack technique, the attacker models DOSC for dynamic scan obfuscation key of each cycle and performs the SAT attack to trace back to LFSR seed only. From an attacker's perspective, this attack model should be the most promising one to compromise the security of DOSC-integrated functional obfuscated circuit, since the seed is the only static asset in DOSC. Furthermore, this attack exploits the minimum complexity bound in a DOSC-inserted obfuscated circuit as it can be performed utilizing test mode only; the functional circuitry is bypassed (represented by the dotted line shown in Figure 9) so it does not introduce any additional complexity. With the knowledge of the seed and the configuration of the LFSR, the attacker can attempt to identify scan obfuscation key at any cycle performing the scrambling translation on its own. This breaks the scan obfuscation security and clears the attacker's path to perform SAT attack to find the functional obfuscation key. Keeping this in mind, we have inserted different size DOSC in the scan chain of b07 ITC'99 [35] benchmark circuit and performed the SAT attack to reveal the seed. For all the attacks here, we have placed one DOSC key gate at the beginning and another at the end of the scan chain. The rest of the DOSC key gates are distributed throughout the chain uniformly.

In Figure 10, the left Y-axis shows the number of clauses in the CNF for different key-size DOSC in terms of experimental results and mathematical estimation from Table 3. The right Y-axis shows the exponential growth of SAT attack execution time. It is observed that the number of clauses in the CNF increases linearly with increasing DOSC size while the SAT attack execution time grows exponentially. We have considered a timeout margin of 10 days here. SAT attack successfully revealed seed in less than 2 h for 4- to 7-bit DOSC, in less than a day for 8- to 10-bit DOSC, in roughly 2 days for 12-bit DOSC, and in nearly 10 days for 16-bit DOSC. A small spike in the number of clauses of CNF for both experimental and model data at 8-bit DOSC occurred due to an increase in the number of XOR operations in the feedback equation. We also performed attacks for larger size DOSC (24–128 bit DOSC), however, all those attacks timed out. From the plot, one can easily extrapolate that it would take 18.5K years to reveal seed of a 50-bit DOSC-inserted into a 50-bit scan chain under this attack model; however, this estimation may differ from actual attack time due to the heuristic nature of SAT.

5.2 SAT Attack on Scan Obfuscation Keys

In this model, the attacker targets LFSR-generated dynamic scan obfuscation keys as a potential asset. With the scan obfuscation key of every cycle revealed, the attacker can reveal the scrambling operation performed by DOSC, which clears the attacker's path to perform SAT attack on logic-obfuscated functional IP. It is the theoretical requirement of the SAT attack that the SAT solver observes input sequence and output response of several cycles to perform a successful attack

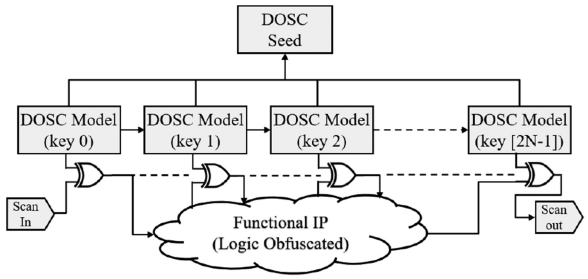

Fig. 9. SAT attack on DOSC-inserted design. As scan obfuscation key generated by DOSC changes each cycle, DOSC is modeled for each cycle before test patterns are applied to logically obfuscated functional IP.

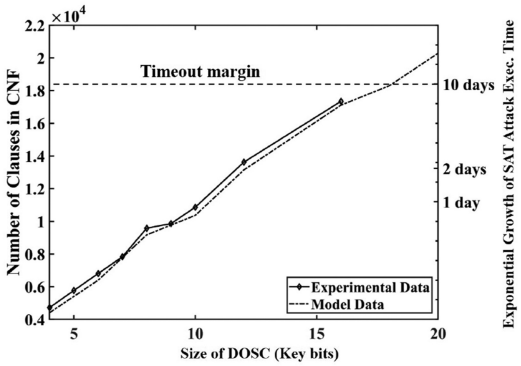

Fig. 10. Scan-only SAT attack complexity analysis and exponential growth of time with DOSC.

provided that the obfuscation key being static during this entire time. However, the scan obfuscation key generated by DOSC changes every cycle making this attack model futile.

5.3 SAT Attack on Obfuscation Key and DOSC Seed

In this attack model, the attacker scans in DIPs through the obfuscated scan chain and investigates the scanned-out responses to reveal functional obfuscation keys. While shifting DIPs into the scan chain and shifting out captured responses from the scan chain, each shifting bit gets scrambled by the scan obfuscation key of the corresponding cycle generated by DOSC (as LFSR keeps generating pseudo-random scan obfuscation keys at each cycle). SAT attack requires the obfuscation key to be static in the duration of applying DIPs and retrieving the output response. As shown in Figure 9, we modeled DOSC for dynamic scan obfuscation key at each cycle for a test event where N scan cycles are needed to shift-in the test patterns and N scan cycles are needed to shift-out the captured responses. We integrated the unrolled model of DOSC with the combinational equivalent of logic-obfuscated functional IP (solid line of XOR gates output to functional IP in Figure 9). As all the dynamic scan obfuscation keys are generated from a static seed, we performed the SAT attack on this model to trace back to the original seed of the DOSC and the secret unlocking key of the functional IP. We inserted different size DOSC in the scan chain of three different size benchmarks—small (s838), medium (b07), and large (s38417). For functional obfuscation, we have followed RLL [4] approach and inserted 32-bit functional obfuscation key gates. For each of the benchmarks, we have performed the SAT attack to reveal both DOSC seed and obfuscation key of functional IP. We have also estimated the complexity of the DOSC-inserted functional obfuscated benchmarks using our mathematical model discussed in Section 4-B. Observations from this ex-

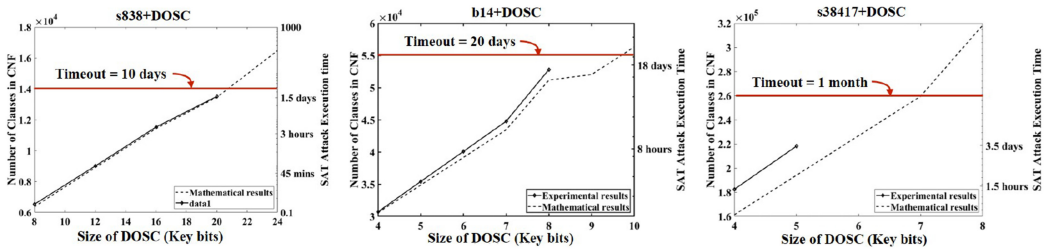

Fig. 11. SAT attack on s838, b14, and s38417 benchmarks integrated with different size DOSC. For each plot, the left Y-axis shows the number of clauses in CNF from the experimental and mathematical model for different size DOSC. The right Y-axis shows the exponential growth of SAT attack execution time for the corresponding benchmark in log scale. A timeout window of 10 days, 20 days, and 1 month has been considered for s838, b14, and s38417 benchmarks, respectively.

periment are shown in Figure 11. It can be seen that SAT attack complexity increases exponentially with increasing DOSC key size for all three benchmarks and out mathematical estimation of complexity is always lower than the actual one. We have considered a timeout margin of 10 days, 20 days, and 30 days, respectively, for small, medium, and large benchmark. In the case of the small benchmark (s838), our results demonstrate that SAT attack timed-out beyond 20-bit DOSC. For medium benchmark circuit (b14), the SAT solver took roughly 18 days to reveal both functional obfuscation key and DOSC seed. We have also performed the SAT attack on s38417, the largest available benchmark circuit from ISCAS'89 [34], with DOSC-inserted in the 1,636-bit scan chain. In this case, the SAT solver took less than 4 days to reveal 5-bit seed and unlocking key of s38417. Once we increased the DOSC key size to 6-bit, the attack went on for months without converging to any solution.

5.4 SAT Attack through Primary IOs

Without scan access, an attacker can no longer control or observe flip-flops for performing the SAT attack. However, the attacker may attempt to perform the SAT attack in functional mode only by unrolling the sequential design and using primary IOs. Sequential circuit unrolling increases the SAT attack complexity. Let us consider that a sequential design has N number of state elements. The total number of possible states in the design is 2^N, though not all states are valid states, i.e., not all states can be traversed through normal state transitions. Marchok et al. [43] showed that the number of valid states only increases linearly with the number of state elements (N), whereas the number of invalid states increases exponentially ($2^N - N$). As a result, it becomes exponentially difficult for the SAT solver to search through the state space and find the valid states.

5.5 Effect of DOSC XOR Gate Placement on SAT Attack

In Section 4-B, we explained DOSC XOR gate placement in the scan chain. We performed the SAT attack on s1423 benchmark, functionally obfuscated with a 20-bit random key (RLL [4]) and the scan chain obfuscated with a 16-bit DOSC, changing DOSC key gate placement. Placement techniques and their attack results are shown in Table 5. In each of the cases, one key gate is placed at the start and end of the chain. It can be observed that SAT attack time increases with increasing spacing and timeout (timeout margin = 10 days) when key gates are distributed uniformly throughout the chain with two gates at farthest corners.

5.6 Overhead Estimation

Table 6 shows the DOSC area and total power overhead as well as test coverage when integrated into different benchmarks. We have developed an automatic tool flow to integrate DOSC

Table 5. Effect of DOSC XOR Gate Placement

Placement Configuration	SAT Attack Time
Spacing = 1	6.25 h
Spacing = 2	13 h
Spacing = 3	3.88 days
Uniformly distributed	time-out

Table 6. Overhead of DOSC on Different Benchmark

Benchmarks	AES CORE	LEON2	LEON3s	LEON3mp	RISC
Size of Scan Chain	530	149434	185109	108872	7606
DOSC Area Overhead	4.67%	0.0545%	0.0533%	0.0919%	1.1392%
DOSC+Functional Obfuscation Area Overhead	5.7%	0.055%	0.0538%	0.0928%	1.151%
Power Overhead	6.15%	0.2845%	0.1973%	0.3512%	0.8189%
Test coverage	98.5%	97.83%	97.3%	97.89%	98.3%

architecture in a logic circuit. This way, we have integrated 128-bit DOSC architecture in the AES core, Leon2, Leon3s, Leon3mp, and RISC processor. Area overhead is up to 4.67% in the case of AES core. For other designs, the area overhead is well below 1%. Furthermore, functional obfuscation of IP may add up to 1% additional overhead [4]. Please note that the trusted platform module (TPM) is used for cryptographic key generation, storage, and restricted use [64] in SoC, which can be utilized for functional obfuscation key management. Although DOSC protects functional obfuscation from oracle-guided attacks, functional obfuscation keys are not a part of DOSC. Therefore, we did not include TPM in the area overhead calculation. Total power overhead is up to 6.15% in AES core. For other designs, total power overhead is below 1%. The power overhead mentioned in Table 6 is based on incremental area overhead due to DOSC insertion and power consumption of those additional gates. In a practical design, power overhead would be even more negligible as DOSC does not operate in functional mode, and test mode is run occasionally. As discussed in Section 3, the fact that DOSC does not alter test coverage is further verified in Table 6. For all the benchmarks, test coverage is above 97.3%. All these analyses are done on post-synthesis design. It is noteworthy that DOSC itself has a definite structure and minimal area overhead, which makes it 100% testable. Furthermore, dynamic scan obfuscation performed by DOSC only scrambles scan chain contents rather than corrupting functional outputs. Therefore, DOSC does not test coverage.

6 RESULT AND DISCUSSION ON OTHER ORACLE-GUIDED ATTACKS

In this section, we discuss how DOSC thwarts oracle-guided scan-based attacks [33, 45–47, 53] and other emerging attacks on logic locking [21–26]. A comparison of different countermeasures against oracle-guided and oracle-less attacks is presented in Table 7.

Scan-based Attacks: DFT techniques are often vulnerable to non-invasive scan-based attacks such as: differential attack [45, 46], resetting attack [33], flushing attack [53], and combinational function recovering attack [47]. All these attacks require access to an active oracle. For DOSC-inserted design, to successfully reveal the secret keys by performing known scan-based attacks, attackers must know the position of XOR gates and how the scan content changes throughout the chain. Consider that the number of flip-flops and XOR gates in a scan chain are N and λ, respectively. The possible combination of XOR gate positions is $\binom{N+1}{\lambda}$. So, the probability of successfully

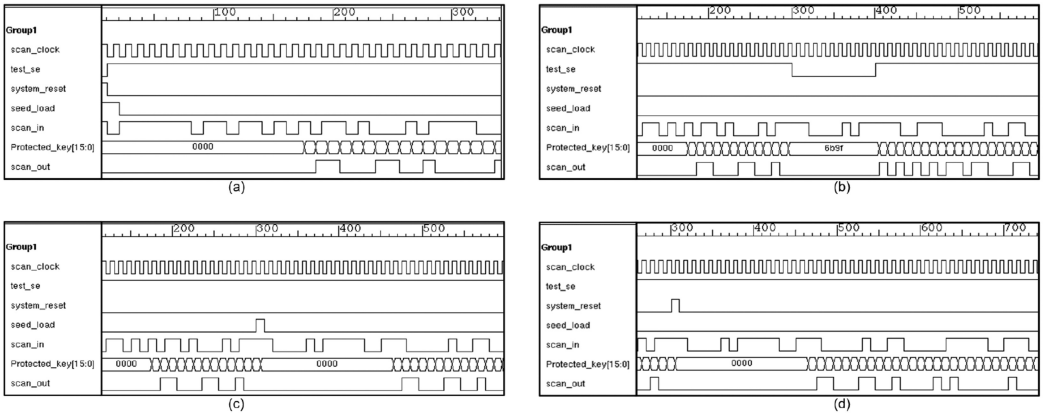

Fig. 12. Waveform showing DOSC resisting oracle-guided scan-based attack, (a) upon system reset, (b) test disable, (c) attempt to load seed in test mode, (d) system reset during test mode.

identifying the XOR gate position is $\binom{N+1}{\lambda}^{-1}$. For each combination of the position, the attacker must identify how the scan content changes from time to time. Therefore, the probability that an attacker can successfully extract the right scan structure is, $= \frac{1}{\sum_{\lambda=0}^{N}\binom{N+1}{\lambda}2^{\lambda}}$. Upon simplifying using binomial theorem, scan-based attack complexity in DOSC-inserted design is $O(3^N)$. In the worst case, if the attacker knows the XOR gate locations in the scan chain, then attack complexity will be at least $\Omega(2^{\lambda})$. To experimentally illustrate how DOSC safeguards design secret from oracle-guided scan-based attacks, we have carried out several control events on a logic locked design scan locked dynamically with a 16-bit DOSC and presented the waveform in Figure 12. Figure 12(a) exhibits the scenario when the chip is boot up in test mode. Usually this is done by performing a system reset, followed by an active-high test enable event. Right after the chip is switched in to test mode, the seed is loaded into the LFSR of DOSC from a tamper-proof memory, which starts activating the shadow chain registers one-by-one. After λ scan cycle, when the last register of the shadow chain turns "1," the dynamic scan obfuscation key starts scrambling the scan chain contents, and obfuscated scan contents start showing up at the scan output. By blocking scan output right after a system reset or seed is loaded, DOSC protects the logic-locked design from oracle-guided resetting attack [33] and flushing attack [53]. Furthermore, to resist the algebraic attack [65] on LFSR, the maximum allowed cycle for flushing is limited by the control vector. When the chip is switched to the functional mode from test mode in Figure 12(b), DOSC blocks scan out and halts the LFSR to the current state. Once the chip is switched back to the test mode, LFSR starts generating patterns, and DOSC starts scrambling scan chain contents that protect against differential attack [45] and combinational function recovering attack [47]. During test mode, if new seed is loaded or a system reset is performed, then the shadow chain is instantly reset, and the scan output is blocked for the next λ scan cycle (shown in Figures 12(c) and 12(d)). This way, the shadow chain ensures resistance against any correlation-based attack.

ScanSAT Attack: Alrahis et al. [26] claimed they revealed LFSR seed of dynamic scan obfuscation by iterative execution of ScanSAT attack using independent SAT attack runs. However, a major issue in Reference [26] is that the authors perhaps misunderstood the role of the shadow chain, which is one of the major components of DOSC. The shadow chain protects dynamic keys from resetting attack, flushing attack, and resists dynamic scan obfuscation keys being applied to the

Table 7. Comparison of Different SAT-resistant Techniques

Countermeasures	Oracle-guided Attacks							Oracle-less Attacks		
	SAT	SPS	AppSAT	Bypass	ScanSAT/DynUnlock	Removal	Scan-based	FALL	Structural	Removal
SARLock	✓	×	×	×	N/A	×	×	✓	×	×
Anti-SAT	✓	×	×	×	N/A	×	×	✓	×	×
SFLL	✓	✓	✓	✓	N/A	✓	×	×	✓	✓
Improved EFF	✓	✓	✓	?	✓/×	✓	✓	✓	✓	×
Scan Controller	✓	✓	✓	?	✓/×	✓	✓	✓	✓	×
DOSC	✓	✓	✓	✓	✓	✓	✓	✓	✓	×

scan chain during initial cycles. If the shadow chain is not performing its role in DOSC, then the attacker can observe the obfuscated scan patterns in the immediate next cycles after applying seed. In DOSC's correct implementation, the scrambled responses start to show up in the scan-out port only when the last flip-flop of the shadow chain becomes "1." Before that, no scan obfuscation key is applied in the scan chain. By that time, dynamic keys are shuffled away from the initial seed. If DOSC is implemented properly, then the shadow chain prohibits first λ cycles (for λ-bit DOSC) of LFSR generated dynamic scan obfuscation keys from applying into the scan chain. The SAT attack model that we discussed in Section 5.1 to reveal LFSR seed, is conceptually similar to ScanSAT attack [26] and DynUnlock [54]. We modeled DOSC for each cycle and performed the SAT attack to trace back to the LFSR seed. However, due to having shadow chain considered in the attack model, our attack complexity on DOSC-inserted design increased exponentially with LFSR seed size.

FALL Attack: FALL attack [25] on logic locking techniques use structural and functional analyses of locked circuits to identify the locking key. The attack methodology uses cube stripping and programmable functionality restoration. In this approach, authors identified sub-circuits corresponding to the cube stripping module, and then extracted the key using functional analysis of these nodes. For the obvious reasons of the target adversary model, FALL attack [25] applies to the circuits that implement stripped functionality logic locking techniques [20] and "hard code" the unlocking key in the cube stripping unit (which leads to the vulnerability). No functional or scan obfuscation keys are hard coded in DOSC-inserted designs, making it resistant to FALL attack.

SAT Variants—AppSAT and TimingSAT: App-SAT [24] is a variant of the original SAT attack [17] attack that requires very low output corruptibility, $Cr \epsilon O(\frac{1}{2^n})$, where n is the number of outputs. In case of SARLock [18] and Anti-SAT [19], $n \epsilon 32 - 64$. In the case of DOSC, LFSR generates pseudo-random scan obfuscation keys. The probability of any LFSR output net being logic "1" or logic "0" is $\frac{1}{2}$. Therefore, DOSC output corruptibility, $Cr \epsilon O(\frac{1}{2})$, is far greater than App-SAT [24] requirement, making DOSC resistant to App-SAT attack. Other variant of SAT attack that has recently been proposed, namely, TimingSAT attack [60], which leverages the timing characteristics of the existing gates in the design and embeds this into SAT instance to recover the delay key of the Delay Locking [59] by satisfying the setup and hold timing constraint along with the proper functioning design. The dynamic scan obfuscation that is performed by DOSC is independent of the gate delays. Therefore, attempt of TimingSAT attack [60] on DOSC will fall into the category of SAT attack model discussed in Sections 4 and 5 and run into exponential complexity.

SPS Attack: SPS attack [22] assumes similar threat model as SAT attack [17] that requires access to an oracle. SPS attack computes the signal probability skew of all gates, and the gate with the highest SPS is the suspect gate to identify SAT-resistant circuitry. For example, in the case of Anti-SAT [19], the highest SPS (=1) occurs at the output of the Anti-SAT block [22]. In the case of DOSC, LFSR generated dynamic keys with roughly 0.5 signal probability in its output due to the

pseudo-randomness, and the scan obfuscation is performed by the XOR gate that produces zero skew signal [22]. Therefore, signal probability of the outputs of DOSC block will be computed be as shown in the following equation [22]:

$$SPS = Pr[x = 1] - 0.5 \approx 0.5 - 0.5 \approx 0. \tag{12}$$

DOSC does not implement any one-point function-based flipping circuitry and hence introduces no skewed signal. We have performed SPS attack on s1423 benchmark, functionally obfuscated with a 20-bit random key (RLL [4]) and the scan chain obfuscated with a 16-bit DOSC. While the overall design had a highest SPS of 0.938, DOSC circuitry had highest SPS of only 0.367.

Key Sensitization Attack: Key sensitization [13] is an oracle-guided attack that requires access to the obfuscated netlist and a functional IC (oracle). The attacker tries to sensitize the key gate to output without being masked or muted by other key gates and inputs. An attacker tries to identify any back-to-back key gates, isolated key gates, or mutable key gates to carry out the attack. Based on the analysis, the attacker tries to generate a golden test pattern that can sensitize the target key gate's effect. We carried out a key sensitization attack on the s1423 benchmark functionally obfuscated with a 20-bit random key (RLL [4]) and the scan chain obfuscated with a 16-bit DOSC. However, due to the scan chain structure, each key gates are at the fan-out cone of the other key gates. Therefore, DOSC key gates cannot be isolated or muted. Moreover, locking the scan chain makes the attacker's goal to generate a golden pattern and determine the key even more difficult.

Bypass Attack: Xiaolin et al. [21] presented bypass attack against SARLock [18] and Anti-SAT [19] that presumes scan access to attack large sequential circuits. DOSC keeps scan access available only to authorized users by dynamically obfuscating scan chain. Thereby, DOSC resists bypass attack.

Removal Attack: Removal attack [23] attempts to retrieve the original circuit by removing or bypassing the SAT attack resistant solution in the circuitry. Removal attack [23] launches SPS attack [22], AppSAT attack [24], and sensitization-guided SAT attack to identify and remove the protection circuitry by taking advantages of structural traces and bias in the design. In the oracle-guided attack threat model, a working unlocked chip (oracle) is required along with the locked netlist to verify the extracted secret key. In the case of DOSC-inserted design the structure of scan obfuscation circuitry is well known to the attacker and removing DOSC from the oracle will restrain the attacker from verifying the correctness of the identified secret key. However, removing DOSC from the locked netlist (e.g., utilizing structural analysis [44]) is possible, which falls under the oracle-less attack threat model and out-of-scope for this article.

7 CONCLUSION AND FUTURE WORK

In this article, we have evaluated, in details, the security of DOSC scheme that restricts effective scan access to authorized users to protect against oracle-guided attacks and demonstrated both mathematically and experimentally how this architecture can combat SAT attack for extracting logic obfuscation keys. We have performed SAT attack on different DOSC-inserted benchmarks and shown that the time increases exponentially with DOSC key length. This article focused on LFSR as a pseudo-random key generator in DOSC architecture. Other alternatives like *non-linear feedback shift register, feed forward LFSR*, and so on, and their security analysis could be a possible expansion. DOSC can secure designs with scan compression, scan compaction, and advanced DFT structures by distributing XOR gates throughout the compressed chains. The pattern generation and security analysis of DOSC in such complex design scenario is considered for future study. The experimental results of SAT attack on DOSC-inserted design that is presented in Section 5 is indeed based on modern SAT solvers, e.g., lingeling [49] and CryptoMinisat [50]. These SAT solvers are

based on the advanced CDCL algorithm inspired by the older DPLL algorithm. There has been further evolution of SAT solvers where a combination of machine learning-based heuristics and the CDCL algorithm is utilized. Analysis of the attack complexity of DOSC-inserted design based on state-of-the-art SAT solvers is part of our ongoing research.

REFERENCES

[1] B. Shakya et al. 2017. Introduction to hardware obfuscation: Motivation, methods and evaluation. In *Hardware Protection through Obfuscation*. Springer, 3–32.

[2] A. B. Kahng et al. 1998. Watermarking techniques for intellectual property protection. In *Proceedings of the 35th Annual Annual Design Automation Conference (DAC'98)*. ACM, 776–781.

[3] IEEE. 2014. IEEE recommended practice for encryption and management of electronic design intellectual property. Retrieved from https://standards.ieee.org/findstds/standard/1735-2014.html.

[4] J. A. Roy et al. 2008. Epic: Ending piracy of integrated circuits. In *Proceedings of the Conference on Design, Automation and Test in Europe*. ACM, 1069–1074.

[5] R. S. Chakraborty and S. Bhunia. 2008. Hardware protection and authentication through netlist level obfuscation. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'08)*. IEEE Press, 674–677.

[6] R. S. Chakraborty and S. Bhunia. 2009. Harpoon: An obfuscation-based SoC design methodology for hardware protection. *IEEE TCAD Circ. Syst.* 28, 10 (2009), 1493–1502.

[7] J. Rajendran et al. 2013. Security analysis of integrated circuit camouflaging. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 709–720.

[8] R. W. Jarvis and M. G. Mcintyre. 2007. Split manufacturing method for advanced semiconductor circuits. U.S. Patent 7,195,931.

[9] M. T. Rahman et al. 2014. CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance (DFT'14)*. IEEE, 46–51.

[10] A. Chhotaray et al. 2017. Standardizing bad cryptographic practice. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'17)*. ACM, 1533–1546.

[11] J. J. Rajendran et al. 2013. Is split manufacturing secure? In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 1259–1264.

[12] J. Rajendran et al. 2013. Fault analysis-based logic encryption. *IEEE Trans. Comput.* 64, 2 (2013), 410–424.

[13] J. Rajendran et al. 2012. Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 83–89.

[14] J. Robertson and M. Riley. 2018. The big hack: How china used a tiny chip to infiltrate U.S. companies. Bloomberg.

[15] DARPA. 2019. Automatic implementation of secure silicon. Retrieved from https://www.darpa.mil/news-events/2019-03-25.

[16] DARPA. 2017. Darpa electronics resurgence initiative. https://www.darpa.mil/work-with-us/electronics-resurgence-initiative.

[17] P. Subramanyan et al. 2015. Evaluating the security of logic encryption algorithms. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'15)*. IEEE, 137–143.

[18] M. Yasin et al. 2016. Sarlock: Sat attack resistant logic locking. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'16)*. IEEE, 236–241.

[19] Y. Xie and A. Srivastava. 2019. Anti-sat: Mitigating sat attack on logic locking. *IEEE Trans. Integr. Circ. Syst.* 38, 2 (2019), 199–207.

[20] M. Yasin et al. 2017. Provably secure logic locking: From theory to practice. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'17)*. ACM, 1601–1618.

[21] X. Xu et al. 2017. Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 189–210.

[22] M. Yasin et al. 2017. Security analysis of anti-sat. In *Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. IEEE, 342–347.

[23] M. Yasin et al. 2017. Removal attacks on logic locking and camouflaging techniques. *IEEE Trans. Emerg. Top. Comput.* 8, 2 (2017), 517–532.

[24] K. Shamsi et al. 2017. Appsat: Approximately deobfuscating integrated circuits. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'17)*. IEEE, 95–100.

[25] D. Sirone and P. Subramanyan. 2019. Functional analysis attacks on logic locking. In *Proceedings of the Design, Automation, and Test in Europe Conference (DATE'19)*. IEEE, 936–939.

[26] L. Alrahis et al. 2019. ScanSAT: Unlocking static and dynamic scan obfuscation. In *IEEE Trans. Emerg. Top. Comput.* (2019). https://doi.org/10.1109/TETC.2019.2940750

[27] X. Wang et al. 2017. Secure scan and test using obfuscation throughout supply chain. *IEEE Trans. Integr. Circ. Syst.* 37, 9 (2017), 1867–1880.
[28] A. Cui et al. 2016. Static and dynamic obfuscations of scan data against scan-based side-channel attacks. *IEEE TIFS* 12, 2 (2016), 363–376.
[29] S. M. Plaza and I. L. Markov. 2014. Protecting integrated circuits from piracy with test-aware logic locking. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14)*. IEEE Press, 262–269.
[30] M. El Massad et al. 2015. Integrated circuit (ic) decamouflaging: Reverse engineering camouflaged ICs within minutes. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'15)*. 1–14.
[31] R. Karmakar et al. 2018. Encrypt flip-flop: A novel logic encryption technique for sequential circuits. Retrieved from https://arXiv:1801.04961.
[32] K. Kursawe et al. 2009. Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'09)*. IEEE, 22–29.
[33] G. Sengar et al. 2007. Secured flipped scan-chain model for crypto-architecture. *IEEE Trans. Integr. Circ. Syst.* 26, 11 (2007), 2080–2084.
[34] F. Brglez et al. 1989. Combinational profiles of sequential benchmark circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. 1929–1934.
[35] F. Corno et al. 2000. Rt-level itc'99 benchmarks and first atpg results. *IEEE DTC* 17, 3 (2000), 44–53.
[36] C. McDonald et al. 2008. An algebraic analysis of trivium ciphers based on the boolean satisfiability problem. In *Proceedings of the 4th International Workshop on Boolean Functions: Cryptography and Applications*. 173–184.
[37] G. V. Bard et al. 2007. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf (2) via sat-solvers. https://eprint.iacr.org/2007/024.
[38] A. Klein. 2013. Linear feedback shift registers. In *Stream Ciphers*. Springer, 17–58.
[39] C. Coarfa et al. 2000. Random 3-sat: The plot thickens. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. Springer, 143–159.
[40] G. S. Tseitin. 1983. On the complexity of derivation in propositional calculus. In *Automation of Reasoning*. Springer, 466–483.
[41] J. C.-M. Li and M. S. Hsiao. 2009. Fault simulation and test generation. In *Electronic Design Automation*. Elsevier, 851–917.
[42] M. Cygan et al. 2016. On problems as hard as cnf-sat. *ACM Trans. Algor.* 12, 3 (2016), 41.
[43] T. E. Marchok et al. 1995. Complexity of sequential ATPG. In *Proceedings of the European Design and Test Conference (ED&TC'95)*. IEEE, 252–261.
[44] P. Chakraborty et al. 2018. Sail: Machine learning guided structural analysis attack on hardware obfuscation. In *Proceedings of the IEEE Asian Hardware-Oriented Security and Trust Conference (AsianHOST'18)*. IEEE, 56–61.
[45] E. Biham and A. Shamir. 1997. Differential fault analysis of secret key cryptosystems. In *Proceedings of the Annual International Cryptology Conference*. Springer, 513–525.
[46] J. D. Rolt et al. 2013. A novel differential scan attack on advanced dft structures. *ACM Trans. Design Autom. Electr. Syst.* 18, 4 (2013), 58.
[47] L. Azriel et al. 2016. Exploiting the scan side channel for reverse engineering of a vlsi device. Technical Report, Technion, Israel Institute of Technology. CCIT Report 897.
[48] El Massad et al. 2017. Reverse engineering camouflaged sequential circuits without scan access. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'17)*. 33–40.
[49] Armin Biere. 2013. Lingeling, plingeling and treengeling entering the SAT competition 2013. In *Proceedings of the SAT Competition*.
[50] Mate Soos. 2016. The CryptoMiniSat 5 set of solvers at SAT competition 2016. In *Proceedings of the SAT Competition*.
[51] Niklas Sorensson and Niklas Een. 2005. Minisat v1. 13-a sat solver with conflict-clause minimization. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*. 1–2.
[52] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. 1999. NuSMV: A new symbolic model verifier. In *Proceedings of the International Conference on Computer Aided Verification*. 495–499.
[53] Y. Atobe et al. 2012. Dynamically changeable secure scan architecture against scan-based side channel attack. In *Proceedings of the IEEE International SoC Design Conference (ISOCC'12)*. 155–158.
[54] N. Limaye et al. 2020. DynUnlock: Unlocking scan chains obfuscated using dynamic keys. Retrieved from https://arXiv:2001.06724.
[55] A. Sengupta et al. 2018. ATPG-based cost-effective, secure logic locking. In *Proceedings of the IEEE 36th Very Large-scale Integration Test Symposium (VTS'18)*. 1–6.
[56] A. Jain et al. 2020. Atpg-guided fault injection attacks on logic locking. Retrieved from https://arXiv:2007.10512.

[57] R. Karmakar et al. 2019. Efficient key-gate placement and dynamic scan obfuscation towards robust logic encryption. *IEEE Trans. Emerg. Topics Comput.* (2019). https://doi.org/10.1109/TETC.2019.2963094
[58] Q. Nguyen et al. 2020. A secure scan controller for protecting logic locking. In *Proceedings of the IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS'20)*. IEEE.
[59] Y. Xie et al. 2017. Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction. In *Proceedings of the 54th Annual Design Automation Conference*.
[60] A. Chakraborty et al. 2020. Evaluating the security of delay-locked circuits. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* (2020). https://doi.org/10.1109/TCAD.2020.3008843
[61] Muhammad Yasin et al. 2017. What to lock? Functional and parametric locking. In *Proceedings of the on Great Lakes Symposium on Very Large-scale Integration (VLSI'17)*.
[62] Muhammad Yasin et al. 2016. Activation of logic encrypted chips: Pre-test or post-test? In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'16)*. IEEE.
[63] Ujjwal Guin et al. 2016. FORTIS: A comprehensive solution for establishing forward trust for protecting IPs and ICs. *ACM Trans. Design Autom. Electr. Syst.* 21, 4 (2016), 1–20.
[64] Tolga Acar et al. 2015. Key management using trusted platform modules. U.S. Patent No. 9,026,805.
[65] Nicolas T. Courtois and Willi Meier. 2003. Algebraic attacks on stream ciphers with linear feedback. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin.

Received June 2020; revised October 2020; accepted December 2020