# Secure and Robust Key-Trapped Design-for-Security Architecture for Protecting Obfuscated Logic

Hadi Mardani Kamali

Department of Electrical and Computer Engineering, University of Florida, email: {h.mardanikamali@ufl.edu}

***Abstract*—Having access to the scan chain of Integrated Circuits (ICs) is an integral requirement of the debug/testability process within the supply chain. However, the access to the scan chain raises big concerns regarding the security of the chip, particularly when the secret information, such as the key of logic obfuscation, is embedded/stored inside the chip. Hence, to relieve such concerns, numerous secure scan chain architectures have been proposed in the literature to show not only how to prevent any unauthorized access to the scan chain but also how to keep the availability of the scan chain for debug/testability. In this paper, we first provide a holistic overview of all secure scan chain architectures. Then, we discuss the key leakage possibility and some substantial architectural drawbacks that moderately affect both test flow and design constraints in the state-of-the-art published design-for-security (DFS) architectures. Then, we propose a new key-trapped DFS (kt-DFS) architecture for building a secure scan chain architecture while addressing the potential of key leakage. The proposed kt-DFS architecture allows the designer to perform the structural test with no limitation, enabling an untrusted foundry to utilize the scan chain for manufacturing fault testing without needing to access the scan chain. Finally, we evaluate and compare the proposed architecture with state-of-the-art ones in terms of security, testability time and complexity, and area/power/delay overhead.**

***Index Terms*—Hardware Security, Logic Obfuscation, Secure Scan Chain Architecture.**

## I. Introduction

The several-billion cost of building a new semiconductor foundry, with huge recurring maintenance costs, has pushed many design houses to become fabless [1]. However, due to the lack of trustworthiness to offshore fabrication and testing entities, many security threats have emerged, such as IP piracy, reverse engineering, and IC overproduction [2]. To combat these threats, amongst many countermeasure solutions, logic obfuscation [3] introduces a form of post-manufacturing programming into the design, making the circuit's functionality to be dependent on the programming values, referred to as the *key*. After fabrication, when the design house receives the fabricated yet obfuscated ICs, the correct key will be programmed into a tamper-proof non-volatile memory (tpNVM), reducing the functionality of the IP to correct functionality. Without the correct key, the obfuscated design implements a different bogus function, generating a different output response (output corruption) to the same input.

### A. SAT Attack: The Game Changer

After the introduction of primitive logic obfuscation solutions [3, 4], in 2015, with assuming that the access to the scan chain of the circuit is *OPEN* for the test/debug purposes, a new and powerful attack based on Boolean satisfiability (SAT) was proposed that virtually threatened the security of the existing logic obfuscation schemes [5]. In the SAT attack, the adversary has access to (1) a successfully reverse-engineered yet obfuscated netlist, and (2) the activated/functional circuit with *OPEN* access to its scan chain. Similar to the miter circuit in the formal verification, in the SAT attack, a SAT solver is invoked iteratively to rule out the incorrect keys. In each iteration of the SAT attack, the SAT solver finds a specific input, called distinguishing input patterns (DIP), distinguishing between two sets of keys. By applying each DIP to the functional circuit, the adversary can rule out the incorrect set(s) before the next iteration. This flow continues until the SAT solver cannot find a new DIP. At this point, any key that generates the correct output for the set of found DIP is the correct key. The SAT attack on primitive logic obfuscation solutions could eliminate all incorrect keys within a few iterations (few minutes).

### B. Countermeasures against the SAT Attack Strength

The main strength of the SAT attack comes from two important factors: (1) The pruning power of each DIP is very high. The portion of incorrect keys that would be ruled out per each iteration is big. This will lead to successful SAT attack termination within a few iterations; (2) The access to the scan chain is *OPEN*, which helps the adversary to apply the SAT attack for each combinational logic part of the circuit, separately. Hence, all previously proposed countermeasures in the literature could be categorized into two main groups each trying to undermine one of these main strength factors of the SAT attack. The first group of countermeasures tries to either weaken the pruning power of DIPs or to introduce a solution that could not be formulated by the SAT attack. However, the main focus of the second group of countermeasures, on the other hand, is to restrict any unauthorized access to the scan chain to invalidate the possibility of engaging the SAT the attack (or one of its derivatives).

### C. Weakening/Disabling the SAT Attack

As mentioned previously, the first group of countermeasures tries to weaken/disable the SAT attack. Hence, there is no concern for the designer to leave the scan chain architecture *OPEN* [12, 20–24, 26, 28, 34–36]. These countermeasures could be divided into three sub-groups: (1) point-function structure, (2) cyclic and behavioral obfuscation, and (3) routing obfuscation.

*1) point-function (PF) structure:* In this group, the SAT attack can rule out few incorrect keys (the best case is *ONE*) per each iteration, which exponentially increases the number

TABLE I: Comparison of State-of-the-art Logic Obfuscation Techniques.

| | Defense | Corrupt ibility | Resilient against ⇓ | | | | | | | | | | | | Overhead | Test/Implementation Issues | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SAT [5] | Removal [6] | Approx. [7] | FALL [8] | cycSAT [9] | icySAT [10] | SMT [11] | CP&SAT [12] | NNgSAT [13] | seq-SAT [14, 15] | Shift&Leak [16, 17] | Scan Unlock [18, 19] | | Limitation | Test Complexity | Test Time |
| | SARLock [20] | | | ✗ | | ✓ | | | | | | | | | | | | |
| Point | Anti-SAT [21] | low | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | N/A | N/A | N/A | low | NO change | NO change | low |
| Function | SFLL [22] | | | ✓ | | ✗ | | | | | | | | | | | | |
| & | Cyclic [23] | | | | | | ✗ | ✗ | ✓ | | | | | | | NO change | NO change | low |
| Behavioral | SRCLock [24] | high | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | N/A | N/A | N/A | high | NO change | NO change | low |
| | Mem-Cyclic [25] | | | | | | ✓ | ✗ | ✓ | | | | | | | NO change | NO change | low |
| & | DLL [26] | | | | | | ✓ | ✓ | ✗ | | | | | | | NO change | NO change | low |
| Routing | Cross-lock [27] | very high | | | | | | | | ✗ | ✗ | | | | very high | | | |
| | Full-Lock [28] | very high | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | N/A | N/A | N/A | very high | NO change | NO change | low |
| | InterLock [12] | very high | | | | | | | | ✓ | ✓ | | | | high | | | |
| | EFF+RLL [29] | | | | | | | | | | | ✗ | ✓ | ✗ | low | NO change | NO change | low |
| | FORTIS+SLL [30] | | | | | | | | | | | ✗ | ✗ | ✓ | moderate | test coverage | NO change | low |
| | R-DFS+SLL [31] | | | | | | | | | | | ✗ | ✗ | ✓ | moderate | test coverage | NO change | low |
| Scan | mR-DFS+RLL [16] | high | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | moderate | test coverage | key init | high |
| Restrict | DynScan+SLL [32] | | | | | | | | | | | ✓ | ✓ | ✗ | high | trusted tester | NO change | low |
| | DisORC+TRLL [33] | | | | | | | | | | | ✓ | ✓ | ✓ | moderate | trusted tester | NO change | low |
| | **Proposed kt-DFS** | **high** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | **low** | **NO change** | **NO change** | **low** |

test coverage: In FORTIS, R-DFS, and MSSD, the scan chain would be blocked after key loading. Functional Test would be limited to observing POs.
key int: In mR-DFS (MSSD), since any form of shift operation has been blocked after key loading, a sys_rst is required for each test pattern, which results in extremely high test time.

of required SAT iterations [20–22]. However, these techniques suffer from some structural vulnerabilities that were eventually exploited to break them [6, 8, 38]. Besides, these techniques suffer from low output corruption. Hence, the adversary could also rely on approximate key with an extremely low error rate, which could be found by approximate SAT attacks [7].

*2) cyclic and behavioral obfuscation:* Since the SAT solver reads the circuits in conjunctive normal form (CNF), a SAT attack works fine if the logic obfuscation is of Boolean nature. So, in some obfuscation techniques, the behavioral properties of the circuit (e.g. timing) have been obfuscated that cannot (challenging to) be translated to CNF [26, 39–41]. Also, the SAT solver only works on directed acyclic graphs (cycle-free). Hence, in some other techniques, the cycle-based obfuscation is introduced, which traps the SAT solver in an infinite loop [23, 24, 34]. However, further investigation breaks these techniques using newer derivatives of SAT attack [9–11, 42].

*3) routing obfuscation:* The main aim of routing obfuscation is to increase the complexity of the SAT problem per each iteration. So, it will substantially increases *each SAT iteration execution time* [12, 27, 28]. In such techniques, symmetric routing structures have engaged. Routing structures will increase the complexity of the SAT circuit per each iteration, drastically. However, primitive routing obfuscation techniques [27, 28] are very recently broken using CP&SAT and NNgSAT [12, 13]. The most recent routing obfuscation technique, i.e. Interlock [12], twists routing and logic obfuscation to be resilient against newer threats.

## *D. Restricting Unauthorized Access to the Scan Chain*

As shown in Table I, although all three previously discussed sub-groups explore the obfuscation space without the necessity of restricting access to the scan chain, they suffer from one of the state-of-the-art attacks, or they incur large overhead. Hence, a few recent studies move towards the investigation of restricting scan access while the logic obfuscation is in place. Since access to the scan chain is a requirement of the SAT attack, the second group of countermeasures tries to block any unauthorized access to the scan chain [16, 17, 31, 33, 43–45].

After restricting the access to the scan chain, the adversary has to rely on primary inputs/outputs (PI/PO) for any attempt of de-obfuscation, and the SAT attack is no longer applicable in this case. This breed of the solutions could be divided into two main sub-groups, called scan chain blockage and scan chain obfuscation. As shown in Table I, the resiliency provided by this group of solutions is more reliable compared to that of the first group. However, the introduction of sequential-based SAT attack (using unrolling as well as bounded-model-checker (BMC)) reveals the vulnerability of the primitive state-of-the-art obfuscation techniques in this category. The most recent study in this category, DisORC + TRLL [33], is a combination of true random logic locking with restricted scan access. Although none of the existing attacks could break this new countermeasure, it still suffers from a few restrictions, particularly during test/debug. For instance, the tester must be in the design house (trusted) to apply the functional test. It might reduce the testability depending on the topological/functional characteristics of the design-under-test, such as cryptographic engines [46].

## *E. Motivation and Contributions*

Table I provides a general overview of all state-of-the-art logic obfuscation solutions and existing attacks on them. As can be seen, the resiliency of techniques with restricted unauthorized scan access structure is more reliable, which results in getting more attention in recent years in the literature. However, each of these techniques also has its own drawbacks [47, 48]. To overcome these drawbacks and to guarantee the resiliency provided by a restricted scan chain architecture, in this paper, we introduce a new secure and robust key-trapped design-for-security (kt-DFS) architecture for protecting obfuscated logic. The proposed kt-DFS architecture will be resilient against all state-of-the-art de-obfuscation techniques at lower overhead, and it also has none of the limitations of the existing countermeasures, particularly during test/debug, such as high test complexity/time. The contributions of this work could be enumerated as follows:
(1) We first provide a holistic overview of architectures in which the unauthorized access to the scan chain is restricted

to guarantee the security of data (the secret) either in the cryptographic engine or in the obfuscated chips.
(2) We demonstrate how the shift operation could be perilous in a flawed scan-restricted architecture. Then, we introduce a new form of shift-and-leak attack on the state-of-the-art scan blockage architecture, which relies on glitches in the blockage circuitry, enforcing us to introduce a more reliable secure scan chain architecture.
(3) We propose a new scan blockage architecture, called key-trapped design-for-security (kt-DFS) architecture. In kt-DFS, we re-design and implement a new secure cell, as well as a new blockage circuitry for the scan chain, which helps to securely keep the content of the scan chain within the chip (with no possibility of leakage) after the key registration (activation).
(4) We thoroughly compare and contrast the proposed kt-DFS with the state-of-the-art secure scan architecture solutions in terms of security, testability, and overhead.

## II. Secure Scan Chain Architectures: from Cryptographic Engines to Obfuscated Circuits

Having access to the scan chain architecture is inevitable for test/debug purposes. In cryptographic engines, which have very sensitive secrets (e.g. encryption key), due to the high fault coverage, the test/debug step requires access to the scan chain to control and observe the internal states of the design-under-test (DUT). However, the scan access for testing might pose security threats to circuits that contain very sensitive secrets.

In the last decade, there have been a number of scan-based attacks on various cryptosystems (such as attacks on DES [49], AES [46], [50], RSA [51] and etc.). Since scan chains directly reveal the internal state of the logic blocks, it raises a big threat to crypto engines. Hence, numerous techniques have been introduced in the literature to show how the crypto engines could be protected against such threats. However, in comparison with the state-of-the-art secure scan chain architectures that mainly protect the secret of the obfuscated circuit, the threat model and assumptions are almost outdated.

### *A. Primitive Threat Model in Cryptographic Engines*

A wide range of primitive secure scan chain architectures in crypto engines relies on the following assumptions: (1) The adversary is not able to de-package the chip and probe the internal signals. (2) The adversary is familiar with the crypto algorithm. (3) The adversary can run the circuit in normal (functional) mode and scan (test) mode and switch between the two modes at any clock cycles. (4) The adversary could access the scan chain inputs/outputs (SI/SO). Hence, (s)he can shift data into internal scan chains through SI and receive the updated internal data for observation through SO. (5) The scan chain structure is unknown to the adversary.

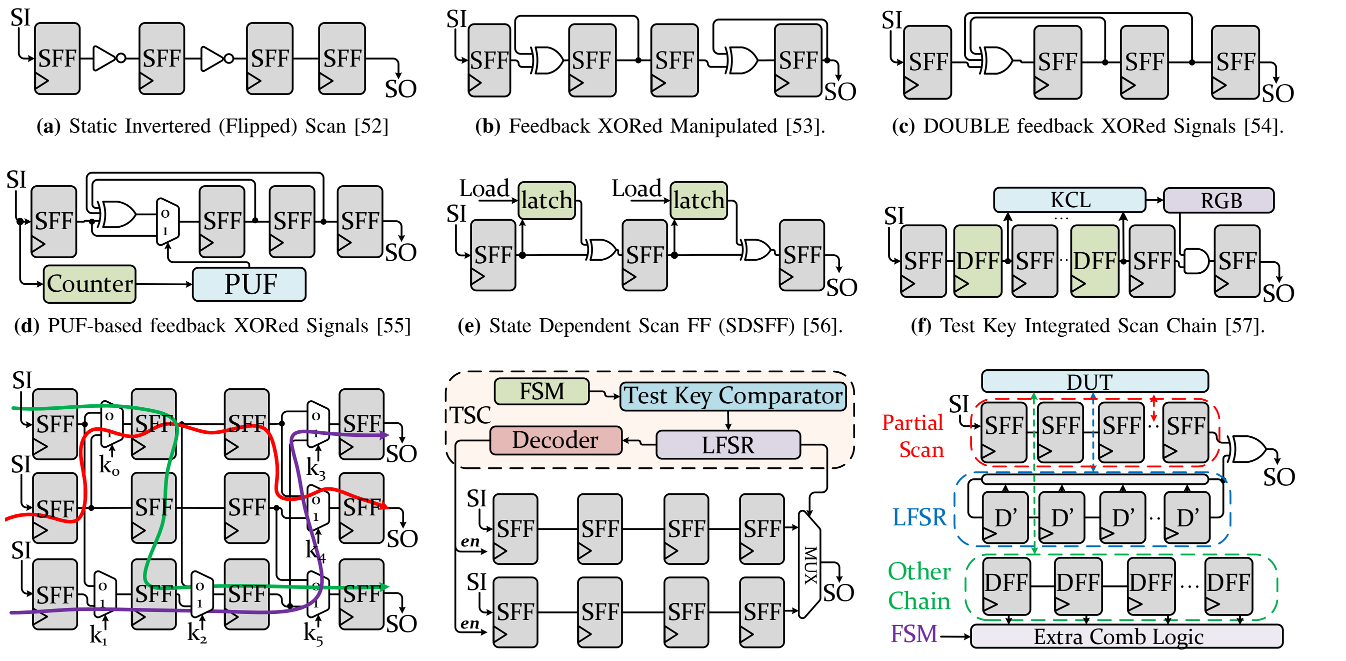
### *B. Secure Scan Architectures in Cryptographic Engines*

Based on the previously discussed threat model, various countermeasures have been proposed to combat scan-based attacks. A set of them proposes statically manipulation (re-ordering) of the scan chain. The very first one is flipped scan [52], demonstrated in Fig. 1(a). As shown, a certain number of inverters is inserted statically and arbitrarily between some of the scan flip-flops (SFF), which change the scan state during shifting in/out. But, the location of inverters in the flipped scan could be found by reset-and-shift attack [53]. Fig. 1(b) and Fig. 1(c) show two extended versions of the flipped scan, where feedbacks from other SFF are involved to avoid reset-and-shift attack [53, 54]. However, these static manipulation mechanisms are still vulnerable to cryptanalysis [55].

Fig. 1(d) shows how randomness could be added to overcome the vulnerability against cryptanalysis, called random-XOR (rXOR). In rXOR, a MUX enables the feedback based on the response of a physical unclonable function (PUF). Although PUF-based rXOR adds randomness to the scan chain, similar to the previous techniques, it still behaves statically, and the changes are only based on the PUF responses to its challenges. It also limits the test flow to be done by a trusted party. Fig. 1(e) shows how external triggers enable the scan architecture to overcome this limitation. It can change the structure of the scan chain dynamically using temporal dependency, called state-dependent SFF (SDSFF) [56]. In SDSFF, external *load* signal determines when the SFF value would be loaded into the latch, and by doing so, SDSFF can change the value of scan output using latch memorizing a past state of the SFF dynamically.

Using external triggers enables the secure scan architectures to dynamically change the state of the scan chain based on the value of external inputs. Hence, some of the newer techniques use external programming values, referred to as the ”key”, to add this capability into the scan chain architecture, which could be categorized as scan-based obfuscation techniques. Fig. 1(f) shows one of these key-based secure scan architectures [57], in which the state of the scan chain is dependent on a test key that is integrated into all test vectors. In this architecture, DFFs store the test key, thereby it must be checked continuously by the combinational key checking logic (KCL), and if the test key fails to be checked by KCL, the random bit generator (RBG) will make the scan chain output unpredictable using randomness. Fig. 1(g) shows another key-based manipulation of the scan chain structure, where a key-based scrambling mechanism determines the order of SFFs [58]. MUXes are added between selected SFFs, whose selectors are key inputs that control the ordering of scan path fragments. When the test key is not correct, the MUXes selectors are fed using random values.

In some solutions, the division of scan chains into several sub-chains has been evaluated. As shown in Fig. 1(h), in sub-chain based scan architecture [59], the scan chain is divided into smaller sub-chains of equal length. In this architecture, the test vectors are not sequentially shifted into each sub-chain but rather a linear feedback shift register (LFSR) performs a pseudo-random selection of a sub-chain to be filled. The LFSR is seeded (initiated) and controlled using a test key. Hence, when the test key is not correct, an incorrect sub-chain will be selected for shifting in/out. However, this technique is vulnerable to signature attack [51], which do not rely on the information of the scan chain order. In a similar approach shown in Fig. 1(i), partial scan chain architecture divides scan



**(a)** Static Invertered (Flipped) Scan [52]. **(b)** Feedback XORed Manipulated [53]. **(c)** DOUBLE feedback XORed Signals [54].

**(d)** PUF-based feedback XORed Signals [55]. **(e)** State Dependent Scan FF (SDSFF) [56]. **(f)** Test Key Integrated Scan Chain [57].

**(g)** Scrambled Scan Chain [58]. **(h)** Sub-chains based secure scan [59]. **(i)** Partial Secure Scan [60].

**Fig. 1:** Primitive Secure Scan Chain Architectures.

chain but removes (unchains) those SFFs that storing sensitive information from the full scan chain [60, 61]. Partial scan architecture consists of four major components: (1) The partial scan with no sensitive data, (2) the SFFs removed from the scan with sensitive data, which are connected to the DUT but not to the SIs/SOs, (3) an FSM that controls the value of the unchained SFFs, which provides the tester full controllability to the unchained SFFs, and (4) an LFSR that stores a backup copy of the removed SFFs to ensure the observability of them. Meanwhile, the output of the LFSR is XORed with the partial scan chain output. Without knowing the state of the FSM and the configuration of the LFSR, the adversary cannot control and observe the value in the unchained flip-flops.

### C. *Advanced Threat Model in Obfuscated Circuits*

Although many of the above-mentioned architectures, particularly key-based techniques, enhance the security of the system, they rely on an outdated and very limited threat model described in Section II-A. One big assumption of this threat model is that the scan chain structure is unknown to the adversary. However, many recent studies demonstrate and validate the possibility of successfully reverse-engineering an IC and a PCB via delayering, imaging, annotation, and netlist extraction [62]. Thus, since the adversary has access to the successfully reverse-engineered netlist, many of the previously discussed secure scan architectures would be invalid.

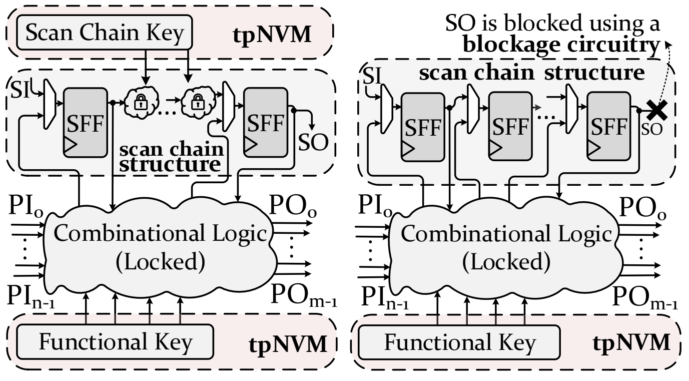
Similar to crypto engines, when logic obfuscation is in place, a protection technique for its secret, which is the obfuscation key, is required. Moreover, since the SAT attack or one of its derivatives on obfuscated circuits requires access to the scan chain, having a secure scan chain architecture in this case is much more demanding. In this case, the threat model on obfuscated circuits follows some stronger assumptions: (1) The designer is trusted, i.e., the personnel and the tools used in the design house are trustworthy. (2) The foundry and the end-user are untrusted. (3) The adversary has access to the successfully reversed-engineered but locked netlist. (4) The adversary can purchase an unlocked/licensed functional chip from the market, so (s)he could apply any desired inputs to an unlocked/licensed chip and monitor the correct outputs. (5) The adversary knows the logic obfuscation technique, as well as the location of the key gates. The obfuscation key is the only unknown value to the adversary.

### D. *Scan Chain Blockage vs. Scan Chain Obfuscation*

When the logic obfuscation is in place, and when the access to the scan chain is *NOT* secured, by getting the benefit of the access to the internal states through the scan chain, the adversary can retrieve the obfuscation key via different methodologies, such as SAT attack [5]. So, many recent studies on logic obfuscation started to explore the possibility of securing the scan access. All existing secure scan chain architectures over locked circuits could be categorized into two main groups: (1) Blocking the scan chain after activation process (scan blockage), and (2) independently obfuscating the scan chain (scan obfuscation). Fig. 2 demonstrates the architectural overview of the scan chain obfuscation and that of the scan chain blockage. It is worth mentioning that in both cases, an obfuscation technique has already been applied to the circuit independently. Hence, as shown in Fig. 2, in both cases, the combinational logic is already obfuscated, and the main aim of the secure scan architecture is to protect the key against existing attacks, such as SAT attack or its derivatives.

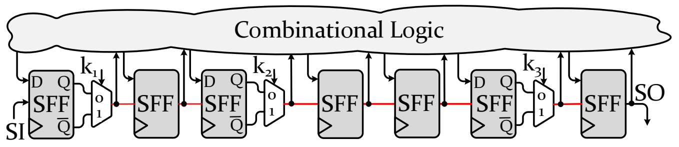
### E. *Logic Obfuscation with an Obfuscated Scan Chain*

To obfuscate the scan chain of an obfuscated circuit, the key gates must be inserted within the scan chain paths. Scan chain obfuscation should be done after design-for-testability (DFT)

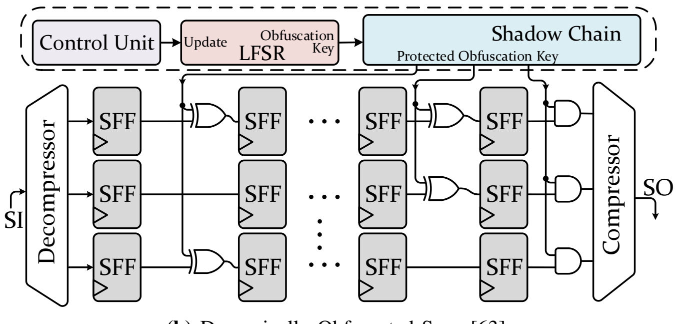


**(a)** Scan Chain Obfuscation **(b)** Scan Chain Blockage

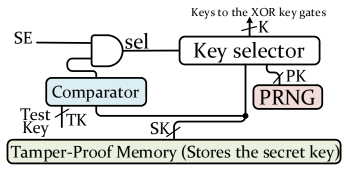
**Fig. 2:** Scan Chain Obfuscation vs. Scan Chain Blockage



**(a)** Encrypt Flip-Flop Strategy [29].



**(b)** Dynamically Obfuscated Scan [63].


Keys to the XOR key gates
K
SE
sel
Key selector
PK
PRNG
Comparator
Test Key
TK
SK
Tamper-Proof Memory (Stores the secret key)

| Input to the Key Gates at Different Conditions (K) | | |
|---|---|---|
| If $SE$ = 0 | If $SE$ = 1 | |
| $SK$ | if $TK$ = $SK$ | $SK$ |
| | if $TK$ ≠ $SK$ | $PK$ |

**(c)** Dynamic Encrypt Flip-Flop (EFF-Dyn) [32].

**Fig. 3:** Scan Chain Obfuscation Techniques

synthesis, where the scan chains are inserted. When the scan chain is obfuscated, the adversary has no longer the ability of loading/reading the initial/updated state into/from SFFs. So, the SAT attack and all other scan-based attacks will fail.

Fig. 3(a) shows the primitive scan obfuscation technique, called*encrypt flip-flop* (EFF) [29]. In EFF, the outputs of a list of selected SFFs are obfuscated (using key-controlled MUXes) based on a placement strategy. Unlike EFF that obfuscates the scan chain statically, a dynamically obfuscated scan (DOS) architecture is first proposed in [63]. As shown in 3(b), in DOS, the scan data is obfuscated using XOR gates, and one input of the XOR gates is controlled dynamically by an obfuscation key generated by a LFSR. So, all the test patterns and responses will be scrambled dynamically. Another dynamic-based scan obfuscation is dynamic encrypt flip-flop (EFF-Dyn) [64] that combines scan obfuscation approach from EFF [29] and a PRNG. In EFF-Dyn, as shown in Fig. 3(c), during either functional mode or the capture operation in test mode, the scan obfuscation key controls the key gates. During testing, an externally provided test key is expected. When this test key matches the scan obfuscation key, the key gates receive this correct key during the shift operations as well; and in case of a mismatch, however, the PRNG that updates the key in every clock cycle controls the key gates dynamically.

## *F. Attacks on Obfuscated Scan Chain Architectures*

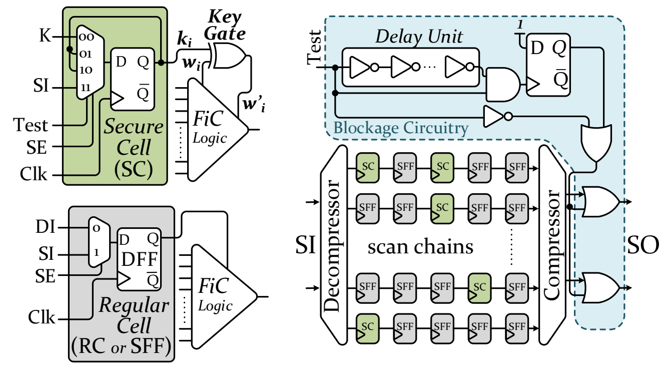
Shortly after the introduction of the primitive studies on scan chain obfuscation, new derivatives of the SAT attack demonstrate the feasibility of breaking these schemes, called *sequential SAT attacks* [14, 15, 18, 65, 66]. Sequential SAT attacks first try to convert each circuit to its combinational counterpart and then apply the SAT attack. This prepossessing step could be done using unrolling with a specific depth (e.g. $u$ times), or using a bounded-model-checker (BMC). So, for any given $u$, SAT solver will find $u$ input patterns, denoted as discriminating input *sequence* ($X_{DIS}$), that have at least one difference at one of $u$ output vectors with two different keys. This process continues until no further $X_{DIS}$ is found within the boundary of $b$. After reaching the boundary, the algorithm checks three criteria (Unique Completion (UC), Combinational Equivalence (CE), and Unbounded Model Check (UMC)) to determine if the attack can be terminated [14].

The sequential SAT was first introduced in [14] that is only applicable to static-based scan obfuscation techniques. The work in [15], improved and accelerated the primitive sequential SAT attack [14] via implementing several tweaks and dynamic optimization techniques in the attack procedure. However, it still only works on static obfuscation techniques. ScanSAT [18] is another sequential SAT attack, which is able to break dynamicity in DOS architecture. In scanSAT, by relying on the fact that the LFSR structure of DOS architecture (and its polynomial) are known to the adversary, it would be able to find the seed as well as update frequency parameter, which is the only secret in DOS architecture. So, it could derive all the keys that are dynamically generated on the chip. Another state-of-the-art sequential SAT attack is DynUnlock [67], which has a similar approach to ScanSAT, and shows how it is possible to break the dynamicity and find the seed of the PRNG in EFF-Dyn.

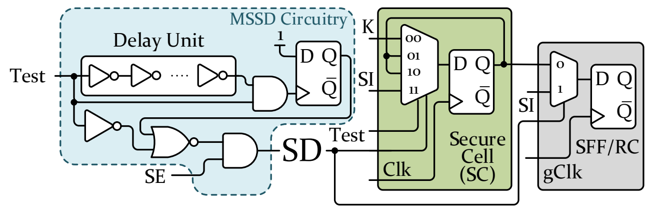
## *G. Logic Obfuscation with a Blocked Scan Chain*

Due to the failure of scan obfuscation architectures against sequential SAT attacks, more recent studies evaluate and reveal the effectiveness of the scan chain blockage after activation of the obfuscated circuit. Scan chain blockage not only limits the access to the scan chain at lower overhead, but it also has no impact on the structural test and negligible impact on the functional test while the overhead is considerably lower than scan chain obfuscation architectures.

The first scan blockage architecture was first introduced in [31], called **r**obust **d**esign-**f**or-**s**ecurity (R-DFS). In R-DFS [31], as shown in Fig. 4(a), the obfuscation key is stored in a custom-designed scan (storage) cell, denoted as secure cell (SC). Based on the value of the $SE$ pin and the new pin



**(a)** Secure Cell (SC) and Blockage Circuitry in R-DFS [31]



**(b)** Updated SC and Shift-Disable (MSSD) Circuitry in mR-DFS [16].
**Fig. 4:** Scan Chain Blockage Techniques

called $Test$, the key values could be loaded into SCs either directly from tpNVM (actual key values for the functional test) or through SI (dummy key for the structural test), and the SO will be blocked after activation (after actual key load into SCs) to avoid any secret leakage. $SE$ and $Test$ pins provide four different modes of operation in SCs of R-DFS, as demonstrated in Table II. However, another study on scan chain blockage shows how R-DFS is still vulnerable against a new shift-and-leak attack [16] when mode $M_{1a}$ still provides shift operation. The shift-and-leak attack breaks R-DFS by exploiting the availability of the shift-in process through the SI, and the capability of reading out the PO through chip pin-outs in the functional mode. Then, as a countermeasure to the shift-and-leak attack, the work in [16] proposes a modified version of robust design-for-security architecture (denoted as mR-DFS in this paper) with a slight modification to the R-DFS. As shown in Fig. 4(b), the leaking mode (mode $M_{1a}$) in R-DFS is fully blocked after the activation in mR-DFS using the new MSSD (blockage) circuitry. In mR-DFS, it is not possible to re-enable any shift mode in the scan chain after the actual key load into the SCs (after mode $M_0$). For $SD$ signal, when $Test$ = 1, $SD$ follows $SE$. But, after the first capture of the actual key (activation), i.e. when the $Test$ is low or when there is a positive transition on the $Test$, $SD$ becomes *ALWAYS ZERO*, thereby blocking the shift operation. Hence, there is no longer a mode where $SC$s can be bypassed, retaining their values, while SFFs can be loaded/shifted.

**TABLE II:** Modes of Operation in Secure Cell (SC) of R-DFS [31].

| Test | SE | Mode | Description |
|---|---|---|---|
| 0 | 0 | $M_0$ | The circuit is in functional mode. Actual keys from tpNVM applies to the Logic (Correct Functionality). |
| 0<br>1 | 1<br>0 | $M_{1a}$<br>$M_{1b}$ | The SCs hold their previous value. Based on the value of SE, RCs are in capture/shift mode. |
| 1 | 1 | $M_2$ | The SCs become part of the scan chain. Actual/Dummy keys from SI for structural testing. |

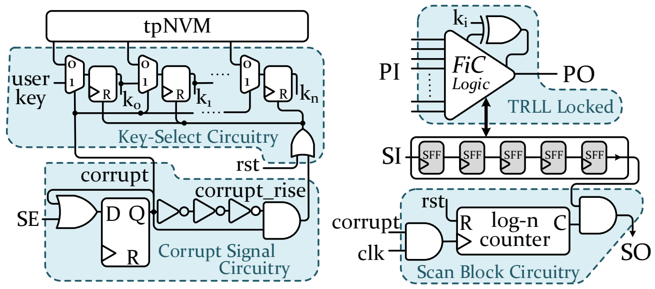


**Fig. 5:** The Dishonest Oracle with Trully Random Logic Locking [33]

In the most recent study, the scan chain blockage is integrated with a new concept called dishonesty of the Oracle [33]. In this new scheme, as shown in Fig. 5, with no extra *controlling* pin, with accessing to the scan chain (through $SE$), a $corrupt$ signal will be enabled, resulting in the disconnection of key storage from tpNVM, resetting the key storage to wiping out the correct key values, and blocking the scan out through a new scan blockage circuitry. So, it can corrupt the functionality (the oracle), when $SE$ is changed.

However, in comparison with the previously elaborated scan blockage techniques, it does not support functional test at an untrusted entity, and a trusted party must have the correct key to perform the functional test, which is a hard assumption to maintain in many practical cases.

## III. Why Blockage is Still Needed to be Improved?

Although scan blockage techniques could be more promising, each of the existing approaches still has its shortcomings. As discussed, R-DFS [31] is already broken using the shift-and-leak attack. In DisORC [33] only a trusted party can have the correct key to perform the functional test, which is a hard assumption to maintain in many practical cases. mR-DFS, on the other hand, is resilient against shift-and-leak attack, and it supports test at low overhead. However, our investigation shows that it also poses some testability/overhead challenges. These challenges are as follows:

### A. High Functional Test Time in mR-DFS

Since there is no longer mode $M_{1a}$ (Table II) in mR-DFS architecture, the tester has to rely on mode $M_2$ to shift in and load the SFFs. Also, since the shift is disabled when $Test$=0 or after the first positive transition on the $Test$[1], $Test$ must be high during power ON. Hence, the tester should use $M_2$ as the initial mode to shift in and load the initial state into the SFFs. After loading the initial state, the tester must switch the mode to $M_0$ to load the actual key (for the functional test). Since it is not possible to re-enable the shift process after switching to mode $M_0$, the tester has to rely on the responses on PO. For the next test pattern, the tester needs to switch back to the mode $M_2$ to shift in and re-load the next initial state corresponded to the next pattern. However, since any shift operation is blocked after switching to mode $M_0$ ($Test$ = 0), the tester cannot use shift-in anymore for shifting in

[1]Based on the MSSD circuitry shown in Fig. 4(b), when $Test$=0 or when there exists a positive transition on the $Test$, $SD$ becomes always zero, and $\{Test, SD\}$ = {01} or {11} could not be selected in SC's MUX any longer.

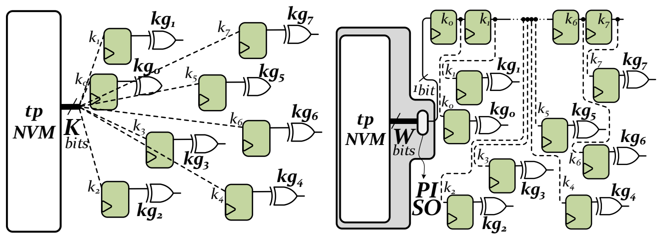
the next initial state. So, the tester has to reset the FF of the MSSD circuitry to re-enable shift-in. This reset re-enables the $SD$ to follow $SE$, thereby, the tester can shift in the next initial state. However this reset (*sys_rst*) will clear all storage elements, including SCs. So, it forces the tester to re-load the keys for the next test pattern. Hence, the actual key must be loaded again from tpNVM to accomplish the functional test, and this key reloading process (with each test pattern) significantly increases the functional test time. It should be noted that the initial state could be chosen to be used for a group of test patterns; however, choosing a specific initial state to be used for a group of patterns would increase the complexity of the functional test significantly. Besides, the designer cannot decouple the reset pins for MSSD. Assuming that this reset pin is separated, the adversary can engage it to re-enable shift operation while the actual key is in place and apply the shift-and-leak attack to retrieve the keys.

## B. Necessity of Duplicating the SCs

In mR-DFS, after shifting in the initial state to the SFFs using mode $M_2$, the tester switches to mode $M_0$ for *ONLY* one cycle to load the actual key. However, during this one cycle, the SFFs (loaded by initial state) would be updated. To avoid such problem, a clock gating circuitry has been introduced in mR-DFS to disable (gate) the clock for one cycle after switching from $M_2$ to $M_0$. However, the question is how the actual key could be loaded into all SCs in just one clock cycle? With no consideration in mR-DFS architecture, there are two possible solutions to load the actual key from tpNVM in one clock cycle: (1) engaging an ultra-wide memory that provides all bits (key bits) at once (one clock cycle) using only a single-clock read operation, (2) engaging temporary registers (FFs) to load the key into them (from tpNVM) at power ON, then connecting each SC to its corresponding temporary register (FFs) to be loaded in one clock cycle.

Regarding the former solution, as demonstrated in Fig. 6(a), it is required to have direct wiring from tpNVM to each SC (per each key bit). Hence, the ultra-wide memory must have an extremely high fanout to provide this direct connection (up to 1K-2K key wires). This ultra-high fanout wiring increases the complexity of placement and routing (PnR) process, and it would significantly decrease the performance of the design, and due to optimization constraints in each design, using this scheme is almost impractical.

The overhead of the latter method, as shown in Fig. 6(b), is more reasonable. However, the required reset (*sys_rst*) for loading the next initial state (described in Section III-A), will clear whole registers in the chip. So, a key re-loading from tpNVM to the temporary register is required for each (group of) test pattern. It raises two big problems in mR-DFS: (1) It significantly increases the required time for functional test, and (2) Since key re-loading from tpNVM takes more than one clock cycle, it violates the assumption of mR-DFS, where clock-gating disables the clock signal only for one clock cycle to preserve the value of the SFFs. So, after only one clock cycle, during the key re-loading, the SFFs would be updated, and the functional test will fail. It implies that even in mR-DFS, the functional test cannot be done in an untrusted entity.

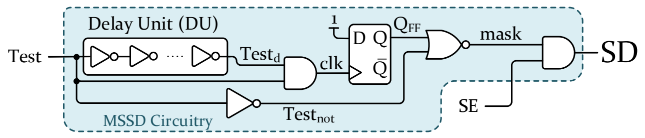


**(a)** Ultra-Wide One-Cycle Readout **(b)** Parallel-in Serial-out (PISO) with Duplicated Registers

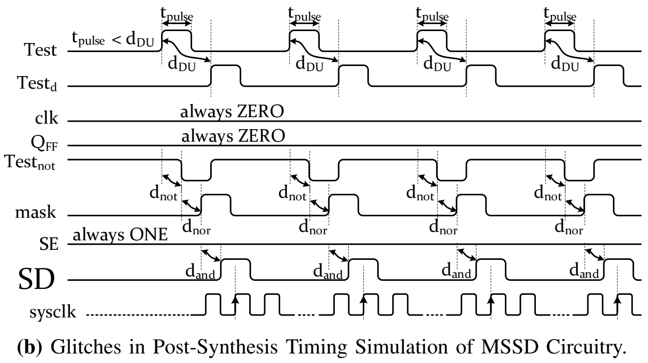
**Fig. 6:** Two Possible One-Cycle Key Readout from tpNVM.

## C. Re-enabling Shift using Leaky Glitches

In mR-DFS, as shown in Fig. 4(b), the selector of $MUX21$ in SFFs is controlled by $SD$, which becomes *ALWAYS ZERO* immediately after the first attempt of switching back from mode $M_0$ to $M_2$ (re-enabling shift process). Switching from mode $M_0$ to $M_2$ means that there is a positive transition on the $Test$ pin, and this positive transition allows the FF in MSSD circuitry to capture its input (*CONSTANT ONE*). However, there is still a possibility to switch back from mode $M_0$ to $M_2$ (positive transition on the $Test$ pin) while the FF does not capture its input (*CONSTANT ONE*) to make $SD$ to be *ALWAYS ZERO*. As shown in Fig. 7(a), a delay unit ($DU$) has been used as a part of the fan-in-cone of the FF in MSSD circuitry, which is built using 10 inverters [16]. Assuming that the adversary is aware of timing information of the circuit, as shown in Fig. 7(b), he or she generates a stimuli for $Test$ pin in which high pulses' width is less than the delay of $DU$ ($t_{pulse} < d_{DU}$). So, the inputs of the first $AND$ gate, $\{Test, Test_d\}$, have no overlap when both are high. So, DFF's $clk$ would be *ALWAYS ZERO*. Since it is assumed that the DFF sets to 0 on reset, $Q_{FF}$ would also be *ALWAYS ZERO*. So, the function of $NOR$ gate will be reduced to $NOT$ gate, whose input is $Test_{not}$. Then, $mask$ follows $Test$ with a delay of $d_{not} + d_{nor}$, and assuming that $SE$ is *ALWAYS ONE*, $SD$ follows $Test$ with a delay of $d_{not} + d_{nor} + d_{and}$. Since $SD$ controls the shift in mR-DFS, using these potential glitches,



**(a)** MSSD (Blockage) Circuitry in mR-DFS.



**(b)** Glitches in Post-Synthesis Timing Simulation of MSSD Circuitry.

**Fig. 7:** Re-enabling Shift after Actual Key Load in mR-DFS using Glitch.

the $SD$ can re-enable the shift operation after mode $M_0$.

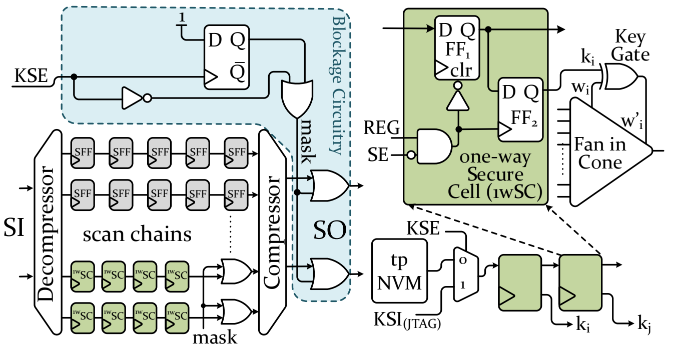
## IV. Proposed Solution: key-trapped Design for Security (kt-DFS)

To introduce a secure and robust scan chain architecture, three requirements must be met: (1) There must be no possibility of key leakage during the test (shift) or functional (capture) mode. (2) Both structural test and the functional test must be carried out in a reasonable time (low test time overhead compared to the test time of the original design) without significant loss of coverage. (3) The complexity of test flow (structural and functional) and the overhead of secure scan chain architecture must be minimal.

In our proposed key-trapped design for security (kt-DFS) scan architecture, to fulfill these requirements, first, the scan chain(s) of the key storage (SCs) are fully decoupled from the scan chain(s) of the regular cells (SFFs). **From the testability, fault coverage, and overhead perspective, there is no reason for stitching the SFF and SC cells in one chain, which has been the source of vulnerability in both R-DFS and mR-DFS**. In the experimental results, we show that the merge of SFFs and SCs would provide minimal (negligible) improvement in terms of overhead. However, similar to both R-DFS and mR-DFS, it significantly compromises the testability and security of the design. As illustrated in Fig. 8(a), there is no common path between SFFs and SCs in the proposed kt-DFS architecture.

To guarantee the security of SCs against any form of leakage, we introduce a new secure cell, called 1-way secure cell (1wSC), depicted in Fig. 8(b). Each 1wSC has two FFs: a chained storage ($FF_1$), and a trap storage ($FF_2$). The chained storage would be used to shift values in and out of the 1wSC or into the trap storage. The trap storage, on the other hand, is connected to its key gate. The behavior of 1wSC is controlled using two pins, called $REG$ and $SE$, and as captured in Table III, it can operate in three main modes:

1) **Functional Mode** ($M_0$): $\{REG, SE\}$={0,0}, in which since $REG = 0$, no more update will be seen in trap storage (FF$_2$s), and since $SE = 0$, the circuits and SFFs are in capture mode (functional mode). Also, another key could be loaded into chained storage (FF$_1$s), simultaneously (if needed).



**(a)** Decoupling SCs from SFFs  **(b)** One-way Secure Cell (1wSC)

**Fig. 8:** (a) Proposed kt-DFS Architecture with New Blockage Circuitry, (b) Using 1wSC for the Key. $KSE$ determines the source of the key.

**TABLE III:** Modes of Operation in kt-DFS.

| REG | SE | Mode | Description |
|---|---|---|---|
| 0 | 0 | $M_0$ (Functional Mode) | $FF_2$ must have the key*. $FF_1$ could capture the key*. |
| 0 | 1 | $M_1$ (Shift Mode) | $FF_1$ could capture the key*. |
| 1 | 0 | $M_2$ (Register Mode) | $FF_2$ are fed from $FF_1$. $FF_1$ will be reset to *ZERO*. Chain is erased. |
| 1 | 1 | $M_3$ (Shift Mode) | $FF_1$ could capture the key*. |

* Based on $KSE$, actual/dummy key could be loaded from tpNVM/$KSI$

2) **Shift Mode** ($M_{1,3}$): $\{REG, SE\}$=$\{X$,1$\}$, in which since $SE = 1$, SFFs are in shift mode. chained storage (FF$_1$s) is able to capture the key, simultaneously, and there is no action on trap storage (FF$_2$s).

3) **Register Mode** ($M_2$): $\{REG, SE\}$={1,0}, in which the pre-loaded key in chained storage (FF$_1$s) would be written into trap storage (FF$_2$s), and chained storage (FF$_1$s) will be reset.

The transfer of key value from $FF_1$ to $FF_2$ takes place after setting $REG = 1$ and $SE = 0$, which is called **Register Mode** ($M_2$). Registration of the key into trap storage takes place as a function of $REG$ and $SE$. This function (condition) is also used as the $RESET$ condition of all $FF_1$s to clear their values. Hence, $AND(Test, \overline{SE})$ is used as the clock source of $FF_2$s, and its toggled is used as the $RESET$ for $FF_1$s.

Also, the trap storage does not have a reset, and upon power-up randomly initialized to 0 or 1. So, upon transition of the key from chained storage to the trap storage, the adversary cannot determine the previous value of the trap storage (randomly initiated). This prevents the back-side imaging attack based on the captured heat map as described in [68][2].

In our proposed kt-DFS, the keys could be loaded into 1wSC from either tpNVM (for the functional test) or scan-in (JTAG for the structural test). Hence, the tester would be able to carry out the structural test by loading the desired key using KSI (JTAG). Another dedicated scan-enable pin in our proposed kt-DFS is used for the scan chain(s) of the SCs, called $KSE$. When $KSE$ is low, the $FF_1$ is fed using tpNVM (actual key for functional mode), and when $KSE$ is high, the $FF_2$ is fed using KSI (JTAG for the arbitrary key used during the structural test). Since the SCs are only added to store the key, none of the internal operations/computations will overwrite the content of the SCs. So, the scan chain of the SCs may have no shift-out pin to be observed. However, for the structural test, all wires of these chains are also required to be observed. So, with no leakage possibility, the output of chains of SCs are available *ONLY* when the $mask$ signal is $ZERO$.

The signal $mask$ is generated in the blockage circuitry and determines whether the actual key (from tpNVM) is loaded, and there exists a possibility of key leakage. So, when the $mask$ is $ONE$, the $SO$, and output of SCs chains would be masked. The signal $mask$ is the main signal of the proposed blockage circuitry, and it will block the $SO$ after the first attempt of (actual) key loading from the tpNVM into $FF_2$s (**Register Mode** ($M_2$)). When $KSE$ is low, the $FF_1$ is fed

[2]E.g. when the activity is observed on heat map for a specific storage element, the adversary cannot determine if the transition is $\{0 \rightarrow 1\}$ or $\{1 \rightarrow 0\}$, and if *NO* activity is observed, the adversary cannot determine if the transition is $\{0 \rightarrow 0\}$ or $\{1 \rightarrow 1\}$.

using tpNVM. So, $\overline{KSE}$ is used to mask the SO. Note that the actual key would be loaded into $FF_2$ when $REG$ = 1 and $SE$ = 0. However, before this condition, the tester has to load the actual key into $FF_1$s while the $KSE$ is low. Hence, by only considering $KSE$ = 0 as the blocking condition, the register mode is also covered. Accordingly, the SO would be no longer available when $KSE$ becomes low. Also, changing $KSE$ from 0 to 1 (rising edge in $KSE$) might cause the key leakage possibility. Hence, it is used as the trigger of the FF used in blockage circuitry to permanently block the SO after this change and avoiding any form of leakage.

### A. Testability/Security/Overhead in kt-DFS

Considering that the leakage problem in R-DFS and mR-DFS is for unnecessary stitching of the SFFs and SCs in the common scan chains, the SCs and SFFs scan chains are fully decoupled in kt-DFS, and the output of the scan chain(s) of SCs is also blocked. The values stored in the chained storage ($FF_1$s) will be cleared with the transfer of the key to the trap storage ($FF_2$s). This guarantees that key values are trapped and neither regular nor glitch-based shift attack can leak the key values to the SO.

In kt-DFS architecture, the functional test and the structural test could be done without any significant limitation or any substantial overhead. For the structural test, since the SCs are equipped with new $KSE$ and $KSI$ (JTAG) pins, it could be accomplished using the following steps. Using these steps, with no limitation and test time overhead, the tester would be able to accomplish the structural test.

1) $KSE \rightarrow 1$, mode $\rightarrow M_0$. Shift in the test key via $KSI$.
2) Switch to mode $M_2$: to write the key into the trap storage ($FF_2$), and to clear scan-connected storage ($FF_1$).
3) Switch to mode $M_1$: to shift in the initial state into SFFs.
4) Switch to mode $M_0$ for one clock for capturing new state.
5) Switch again to mode $M_1$ to shift out the SFFs to SO.

Unlike the structural test, the functional test requires the actual key. Hence, loading the key from tpNVM followed by register mode will block the SO. Considering the blockage of the SO, the steps of the functional test is as follows:

1) $KSE \rightarrow 0$, mode $\rightarrow M_0$. Shift in the actual key from tpNVM. (When $KSE$ = 0, the SO is blocked.)
2) Switch to mode $M_2$: to write the key into the trap storage ($FF_2$), and to clear scan-connected storage ($FF_1$).
3) Switch to mode $M_1$: to shift in the initial state into SFFs.
4) Switch to mode $M_0$ for one clock cycle for capturing the new state and clocklessly observe the PO.

During the structural test, the chains' outputs of 1wSCs are still available ($mask = 0$). The debug of all 1wSCs thus would be performed with no limitation. It should be noted that similar to R-DFS and mR-DFS, the tester accomplishes the functional test by observing the PO with negligible loss of coverage.

Decoupling the scan chain(s) of SCs from that of SFFs helps to facilitate the test flow for the tester compared to the test flow in mR-DFS. Despite mR-DFS with a mandatory *sys_rst* for each (group of) test pattern, no additional operation is required in kt-DFS for any form of the test. No *sys_rst* is required, and none of the operations are blocked after the first attempt of the actual key loading from tpNVM, and similar to R-DFS, only the SO is blocked to break the SAT attack. However, unlike R-DFS, it is fully secure against any form of leakage-based attacks, such as shift-and-leak.

Regarding the overhead, 1wSC has two FFs and has a larger footprint compared to the SCs used in R-DFS and mR-DFS. However, as discussed in Section III-B, the R-DFS, mR-DFS, and even DisORC, also need to transfer the key values from tpNVM to SCs. Using a very wide memory to derive thousands of keys is quite impossible in terms of area, and it imposes higher complexity during PnR. Hence, the R-DFS, mR-DFS, and DisORC also need to resort to a chain of temporary registers to transfer the keys. So, there is also a duplicated register per each SC in these schemes. Furthermore, compared to $MUX41$ in both R-DFS and mR-DFS, only one $AND$ gate and one $NOT$ have been used in each 1wSC, which slightly improves the area overhead. In the experimental result section, we demonstrate that compared to these architectures, the proposed kt-DFS incurs lower overhead while it guarantees security against existing attacks.

## V. Experimental Result

To analyze the security of the kt-DFS, and to provide better comparative results, with engaging the largest ISCAS-89 and ITC-99 benchmark circuits, as shown in Table IV, we re-produce the results for R-DFS+SLL [69], mR-DFS+RLL [16], DisORC+TRLL [33]. Also, for the proposed kt-DFS, we engage strong logic locking (SLL) [4] in all experiments to determine the location of key gates, and the number of key bits is set to 128/256. All the experiments have been accomplished on a Dell PowerEdge R620 equipped with Intel Xeon E5-2670 2.50GHz and 64GB of RAM, using Synopsys Design Compiler 2017.09, Tetramax 2017.09, and VCS 2017.12 tools along with the Synopsys generic 32nm library.

Table V demonstrates the area/power overhead mitigation of the proposed kt-DFS architecture compared to the state-of-the-art techniques. We achieved this improvement because we could remove multiplexers (MUX21 or MUX41) from our new proposed secure cell structure. Since we fully decoupled the SCs from SFFs, it allows us to remove the MUXes that were required to control the inputs of SCs when SCs and SFFs are in the same scan chain. It is worth mentioning that to avoid facing drastic area/delay overhead in the existing approaches, we use temporary registers used for temporary loading the key from the tpNVM (shown in Fig. 6(b)). Without this mechanism, we cannot have the keys in existing techniques at once as claimed.

Table VI reflects the impact of a varying number of scan chains as well as the key size when the proposed kt-DFS is in place. Since the chains of SCs are fully decoupled, varying the number of scan chains has a minimal impact on the overhead of the proposed architecture. For chains of SCs, we assume

**TABLE IV:** Specifications of the Benchmark Circuits (ISCAS'89, ITC'99).

| Circuit | s15850 | s35932 | s38584 | b17 | b18 | b19 |
|---|---|---|---|---|---|---|
| # of Inputs | 77 | 35 | 38 | 37 | 37 | 24 |
| # of Outputs | 150 | 320 | 304 | 97 | 23 | 30 |
| # of Gates | ∼10K | ∼16K | ∼20K | ∼28K | ∼95K | ∼190K |
| # of FFs | ∼0.5K | ∼1.7K | ∼1.5K | ∼1.5K | ∼3.3K | 6.6K |
| Area (mm$^2$) | 0.025 | 0.031 | 0.041 | 0.055 | 0.238 | 0.539 |
| Power (mW) | 1.37 | 1.98 | 2.91 | 3.27 | 9.08 | 19.4 |

**TABLE V:** Post-Synthesis Area/Power Overhead Comparison between R-DFS, mR-DFS using MSSD, DisORC, and the Proposed kt-DFS for Identical Timing Constraints. (Key Size = 128, Number of Scan Chain = 1)

| Circuit | R-DFS+SLL [31] | | mR-DFS+RLL [16] | | DisORC+TRLL [33] | | kt-DFS+SLL | |
|---|---|---|---|---|---|---|---|---|
| | Area (%) | Power (%) | Area (%) | Power (%) | Area (%) | Power (%) | Area (%) | Power (%) |
| s15850 | 17.85% | 24.76% | 15.02% | 21.77% | 19.07% | 25.44% | 12.68% | 16.27% |
| s35932 | 15.99% | 19.39% | 11.54% | 16.54% | 15.28% | 17.02% | 8.52% | 10.28% |
| s38584 | 15.52% | 18.95% | 12.37% | 15.28% | 14.71% | 15.83% | 8.76% | 9.92% |
| b17 | 7.31% | 10.21% | 6.33% | 9.17% | 7.81% | 10.51% | 2.76% | 6.24% |
| b18 | 3.89% | 6.71% | 2.85% | 5.51% | 3.73% | 7.42% | 0.97% | 3.38% |
| b19 | 2.53% | 4.55% | 1.28% | 4.06% | 2.15% | 1.81% | 0.51% | 1.75% |

**TABLE VI:** Post-Synthesis Area/Power Overhead of the Proposed kt-DFS with Different {Key Sizes, Number of Scan Chains}*.

| Area Overhead | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Circuit | {128, 1}* | {128, 2} | {128, 4} | {128, 8} | {256, 1} | {256, 2} | {256, 4} | {256, 8} |
| s15850 | 12.68% | 12.71% | 12.75% | 12.95% | 18.82% | 18.89% | 18.98% | 18.08% |
| s35932 | 8.52% | 8.65% | 8.72% | 8.95% | 11.22% | 11.26% | 11.32% | 11.42% |
| s38584 | 8.76% | 8.81% | 8.86% | 8.97% | 10.79% | 10.86% | 10.91% | 11.03% |
| b17 | 2.76% | 2.78% | 2.81% | 2.83% | 5.92% | 5.94% | 5.97% | 5.99% |
| b18 | 0.97% | 0.98% | 0.98% | 0.99% | 1.43% | 1.46% | 1.46% | 1.46% |
| b19 | 0.51% | 0.51% | 0.51% | 0.52% | 1.01% | 1.02% | 1.02% | 1.02% |
| **Power Overhead** | | | | | | | | |
| Circuit | {128, 1}* | {128, 2} | {128, 4} | {128, 8} | {256, 1} | {256, 2} | {256, 4} | {256, 8} |
| s15850 | 16.27% | 17.36% | 18.24% | 19.52% | 23.54% | 23.68% | 25.15% | 26.77% |
| s35932 | 10.28% | 10.43% | 11.63% | 12.71% | 13.28% | 13.13% | 14.26% | 15.82% |
| s38584 | 9.92% | 9.86% | 10.71% | 12.33% | 12.88% | 12.91% | 14.09% | 15.67% |
| b17 | 6.24% | 6.26% | 7.15% | 7.75% | 9.81% | 10.23% | 12.08% | 13.25% |
| b18 | 3.38% | 3.36% | 3.74% | 3.91% | 5.16% | 5.83% | 6.17% | 6.44% |
| b19 | 1.71% | 1.74% | 1.95% | 1.97% | 2.09% | 2.01% | 2.34% | 2.69% |

**TABLE VII:** Post-PnR Area Overhead of the Proposed kt-DFS (Decoupling SCs and SFFs) vs. the Same Architecture when SCs are Stitched with SFFs.

| Circuit | Decoupling SCs and SFFs | | | | Stitching SCs with SFFs | | | |
|---|---|---|---|---|---|---|---|---|
| | {128, 1}* | {128, 2} | {128, 4} | {128, 8} | {128, 1} | {128, 2} | {128, 4} | {128, 8} |
| s15850 | 13.53% | 13.72% | 13.81% | 13.97% | 12.86% | 13.22% | 13.28% | 13.72% |
| s35932 | 9.13% | 9.22% | 9.41% | 9.55% | 8.56% | 8.65% | 9.12% | 9.32% |
| s38584 | 9.07% | 9.54% | 9.38% | 9.66% | 8.67% | 9.01% | 9.18% | 9.23% |
| b17 | 3.01% | 2.96% | 3.22% | 3.18% | 2.76% | 2.64% | 2.89% | 3.09% |
| b18 | 1.05% | 1.06% | 1.06% | 1.06% | 0.94% | 0.97% | 0.98% | 1.01% |
| b19 | 0.68% | 0.67% | 0.69% | 0.69% | 0.65% | 0.66% | 0.66% | 0.66% |

**TABLE VIII:** Test Coverage and Key Leakage Comparison between R-DFS, mR-DFS using MSSD, DisORC, and the Proposed kt-DFS for Identical Timing Constraints. (Key Size = 128, Number of Scan Chain = 1)

| Circuit | Original | R-DFS [31] | | mR-DFS [16] | | DisORC [33] | | Proposed kt-DFS | |
|---|---|---|---|---|---|---|---|---|---|
| | Test (%) | Test (%) | Key Leak (#) | Test (%) | Key Leak (#) | Test (%) | Key Leak (#) | Test (%) | Key Leak (#) |
| s15850 | 100% | 100% | 127 | 100% | 127 | 100% | 0 | 100% | 0 |
| s35932 | 100% | 100% | 128 | 100% | 128 | 100% | 0 | 100% | 0 |
| s38584 | 100% | 100% | 128 | 100% | 128 | 100% | 0 | 100% | 0 |
| b17 | 99.91% | 99.72% | 127 | 99.69% | 127 | 99.91% | 0 | 99.67% | 0 |
| b18 | 99.77% | 99.78% | 126 | 99.73% | 126 | 99.97% | 0 | 99.73% | 0 |
| b19 | 99.81% | 99.78% | 127 | 99.78% | 127 | 99.82% | 0 | 99.78% | 0 |

that when key size is 128, it is one chain of SCs, and when the key size is 256, it is two chains of SCs. In terms of power overhead, since dividing the scan chain into multiple ones always incur more power overhead, the impact ratio is a bit higher. Also, as demonstrated in Table VI, when we increase the key size (128 $\rightarrow$ 256), since the main part of the overhead is the extra FFs added for key storage (This is the same for all existing techniques), for larger circuits, due to decreasing the ratio of SCs to SFFs, the overhead is less, and in general, it is small enough as expected.

Unlike state-of-the-art techniques, in the proposed kt-DFS, we fully decoupled SCs from SFFs. However, this separation will affect the placement and routing of chains to get the most benefit of optimization steps during DFT synthesis (based on the locality of storage cells). Table VII shows the area overhead after placement-and-routing (post-PnR) of our proposed kt-DFS when SCs are decoupled from SFFs versus when SCs are stitched with SFFs. As shown, when the number of scan chains for SFFs is increased, due to breaking SFFs into sub-domains (sub-locality), post-PnR area overhead is much closer to the stitched version. On the other hand, when the number of scan chains is also few (like 1), the impact of decoupling is also minimal. Hence, to summarize, decoupling SCs from SFFs has minimal impact on post-PnR overhead.

Table VIII represents the structural test coverage and the leakage of R-DFS, mR-DFS, DisORC, and our proposed kt-DFS when the number of scan chains is set to be 1. For almost all techniques, the manufacturing test works perfectly fine, and the test coverage is roughly the same for all cases. However, for R-DFS and mR-DFs with stitching architecture, shift-and-leak and glitch-based shift-and-leak attack can recover the key of the locked circuit. However, our proposed architecture, with decoupled structure, helps keeping the locked circuit secure against leakage possibilities.

Since access to the scan chains is restricted in these techniques, the SAT attack cannot be deployed. This does not prevent an attacker from deploying the unrolling or bounded-model-checking (BMC) [14] attack that only needs PI/PO. However, this group of attacks runs into scalability issues as they rely on two sub-routines which are in PSPACE and NP [16]. Even the accelerated version of this attack (described in [15]) fails to terminate for even small designs. Besides, new techniques such as DFSSD [45] shows how low overhead techniques, like deep faults and shallow state duality, could be used to break the state-of-the-art sequential SAT attacks. To show the lack of scalability of the BMC or unrolling-based SAT attacks, we apply KC2 [15] on kt-DFS+SLL locked circuits, and the results are reflected in Table IX. As shown, this attack could only work for the two smallest circuits with the key size equal to 100, and for all other cases, it fails to reach the result before the time-out ($10^5$ Seconds).

Table X represents a general and top-view comparison between the state-of-the-art secure scan architectures versus our proposed one. As shown, all techniques consist of a new blockage circuitry to restrict access to the scan chain helping us to combat the SAT attack. However, with no full disabling of the shift access, our proposed kt-DFS can provide the high resiliency with no leakage at lower overhead. We counted the number of logic gates that are required per each technique, and this confirms why the proposed kt-DFs incurs less area/power

**TABLE IX:** Execution Time of Fast Sequential SAT (KC2) [15] on the SSL Obfuscated Circuits [70] with Scan Secured with the Proposed kt-DFS.

| Circuit | Key Size = 100 | | Key Size = 200 | |
|---|---|---|---|---|
| | Iterations | Execution Time (s) | Iterations | Execution Time (s) |
| s15850 | 31 | 2666 | *timeout* | *timeout* |
| s35932 | 184 | 15328 | *timeout* | *timeout* |
| s38584 | 77 | 11709 | *timeout* | *timeout* |
| b17 | *timeout* | *timeout* | *timeout* | *timeout* |
| b18 | *timeout* | *timeout* | *timeout* | *timeout* |
| b19 | *timeout* | *timeout* | *timeout* | *timeout* |

- timeout: $10^5$ Seconds $\approx$ one day (Stop the attack when time reaches timeout)

**TABLE X:** Major Differences between Existing and the Proposed Scan Blockage Technique in terms of Overhead / Test / Security.

| Circuit | R-DFS [31] | mR-DFS [16] | DisORC [33] | Proposed kt-DFS |
|---|---|---|---|---|
| Mechanism | New Secure Cell<br>+ New Blockage Circuitry<br>+ Stitching Secure Cells with SFFs | New Secure Cell<br>+ Blockage Circuitry + Disabling Shift<br>+ Stitching Secure Cells with SFFs | New Blockage Circuitry<br>+ Oracle Dishonesty + Key Selection<br>+ Disabling Shift + JTAG Key | New Secure Cell<br>+ New Blockage Circuitry<br>+ Decoupling Secure Cells from SFFs + JTAG Key |
| Suggested Obfuscation | SLL [70] | RLL [3] | TRLL [33] | SLL [70] |
| Logic (Gate-Level) Overhead | $(2K+1)$×FFs<br>$(K)$×MUX41<br>(4)×INV1<br>$(M+1)$×OR2<br>(1)×AND2 | $(2K+4)$×FFs<br>$(K)$×MUX41<br>(11)×INV1<br>$(M+2)$×OR2 & (2)×NOR2<br>(4)×AND2 | $(2K+log(M)+1)$×FFs<br>$(K)$×MUX21<br>(4)×INV1<br>$(M+2)$×OR2<br>$(log(M)+4)$×AND2 & (1)×NAND2 | $(2K+1)$×FFs<br>(1)×MUX21<br>(3)×INV1<br>$(M+1)$×OR2<br>(1)×AND2 |
| Structural Test | Supported ✓ | Supported ✓ | Supported ✓ | Supported ✓ |
| Functional Test | Through POs ✓ | NOT Supported ✗* | NOT Supported ✗* | Through POs ✓ |
| Attacked by | Shift and Leak [16] | Glitch-based Shift and Leak | None | None |

$K$: Size of the Key Input $M$: Number of Scan Chains *: Functional test could be done only at design house (trusted entity)

overhead compared to its competitors. Also, although DisORC provides a secure mechanism with no leakage possibility, its area overhead is higher than other methods, and it has limitations during the test, such as lack of functional test support by untrusted entities. It is worth mentioning that the main aim of this work is focusing on the scan architecture, and we do not consider the advantages/disadvantages of the suggested obfuscation technique that should be integrated with these architectures. Based on our experimental results, we added a ranking in Table X for most three important factors: (1) security, (2) overhead, and (3) testability, and we see that in all cases our proposed scheme could be one of the best amongst all state-of-the-art approaches.

## VI. Conclusion

Although having access to the scan chain of ICs is mandatory for testability/debugging, it raises concerns about the security of the chips, particularly when there exists secret within the chip, such as the symmetric/asymmetric key of cryptographic algorithms and logic obfuscation key. In this paper, we first reviewed all solutions and countermeasures proposed to build a secure scan chain architecture, used either for crypto engines or when logic locking is in place. We showed that many of the preliminary approaches relied on a very limited threat model that is changed drastically over time. We then evaluated the most recent design-for-security (DFS) architectures with all their shortcomings. We proposed a new DFS solution, denoted as key-trapped DFS (kt-DFS) that addresses the prior art shortcomings. In kt-DFS, we introduced a new secure storage cell for the storage of key values. The proposed secure cell allows us to trap the key after being loaded, preventing different forms of shift and leak attacks (glitch based or logic-based), while safely removing the key upon transition from functional to test mode. At the same time, we illustrated that using the proposed kt-DFS architecture, the design can safely undergo manufacturing and functional testing without incurring any significant limitation in terms of an increase in the test time while maintaining desirably low overhead.

## References

[1] A. Yeh, "Trends in the Global IC Design Service Market," *DIGITIMES research*, 2012.

[2] M. Rostami *et al.*, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[3] J. A. Roy *et al.*, "EPIC: Ending Piracy of Integrated Circuits," in *Design, Automation and Test in Europe (DATE)*, 2008, pp. 1069–1074.

[4] J. Rajendran *et al.*, "Security analysis of logic obfuscation," in *Design Automation Conference (DAC)*, 2012, pp. 83–89.

[5] P. Subramanyan *et al.*, "Evaluating the Security of Logic Encryption Algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[6] M.Yasin *et al.*, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[7] K. Shamsi *et al.*, "AppSAT: Approximately deobfuscating integrated circuits," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[8] D. Sirone *et al.*, "Functional analysis attacks on logic locking," *IEEE Transactions on Information Forensics and Security*, 2020.

[9] H. Zhou *et al.*, "CycSAT: SAT-based Attack on Cyclic Logic Encryptions," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 49–56.

[10] K. Shamsi *et al.*, "IcySAT: Improved SAT-based Attacks on Cyclic Locked Circuits," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–7.

[11] K. Z. Azar *et al.*, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, pp. 97–122, 2019.

[12] H. M. Kamali *et al.*, "InterLock: An Intercorrelated Logic and Routing Locking," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[13] K. Z. Azar *et al.*, "NNgSAT: Neural Network guided SAT Attack on Logic Locked Complex Structures," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[14] M. El Massad *et al.*, "Reverse engineering camouflaged sequential circuits without scan access," in *IEEE/ACM Int'l Conf. on Computer-Aided Design*, 2017, pp. 33–40.

[15] K. Shamsi *et al.*, "KC2: Key-Condition Crunching for Fast Sequential Circuit Deobfuscation," pp. 1–6, 2019.

[16] N. Limaye *et al.*, "Is robust design-for-security robust enough? attack on locked circuits with restricted scan chain access," *arXiv preprint arXiv:1906.07806*, 2019.

[17] H. M. Kamali *et al.*, "On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic," in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, 2020, pp. 1–6.

[18] L. Alrahis *et al.*, "ScanSAT: Unlocking Obfuscated Scan Chains," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 352–357.

[19] N. Limaye *et al.*, "DynUnlock: Unlocking Scan Chains Obfuscated using Dynamic Keys," in *Design, Automation and Test in Europe (DATE) Conference*, 2020.

[20] M. Yasin *et al.*, "SARLock: SAT Attack Resistant Logic Locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[21] Y. Xie *et al.*, "Mitigating SAT Attack on Logic Locking," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016, pp. 127–146.

[22] M. Yasin *et al.*, "Provably-Secure Logic Locking: From Theory to Practice," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1601–1618.

[23] K. Shamsi *et al.*, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 173–178.

[24] S. Roshanisefat *et al.*, "SRCLock: SAT-resistant cyclic logic locking for protecting the hardware," in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 153–158.

[25] A. Rezaei *et al.*, "Cyclic Locking and Memristor-based Obfuscation against CycSAT and inside Foundry Attacks," in *Proceedings of the*

*Conference on Design, Automation and Test in Europe (DATE)*, 2018, pp. 85–90.

[26] Y. Xie *et al.*, “Delay Locking: Security Enhancement of Logic Locking against IC Counterfeiting and Overproduction,” in *Proceedings of Design Automation Conference (DAC)*, 2017, p. 9.

[27] K. Shamsi *et al.*, “Cross-lock: Dense layout-level interconnect locking using cross-bar architectures,” in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 147–152.

[28] H. M. Kamali *et al.*, “Full-Lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks,” in *Proceedings of Design Automation Conference (DAC)*. ACM, 2019, p. 89.

[29] R. Karmakar *et al.*, “Encrypt flip-flop: A novel logic encryption technique for sequential circuits,” *arXiv:1801.04961*, 2018.

[30] U. Guin *et al.*, “Fortis: a comprehensive solution for establishing forward trust for protecting ips and ics,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, pp. 1–20, 2016.

[31] U. Guin *et al.* , “Robust design-for-security architecture for enabling trust in ic manufacturing and test,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 818–830, 2018.

[32] R. Karmakar *et al.*, “Efficient Key-gate Placement And Dynamic Scan Obfuscation Towards Robust Logic Encryption,” *IEEE Transactions on Emerging Topics in Computing*, 2019.

[33] N. Limaye *et al.*, “Thwarting All Logic Locking Attacks: Dishonest Oracle with Truly Random Logic Locking,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[34] S. Roshanisefat *et al.*, “SAT-Hard Cyclic Logic Obfuscation for Protecting the IP in the Manufacturing Supply Chain,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2020.

[35] H. M. Kamali *et al.*, “LUT-lock: A Novel LUT-based Logic Obfuscation for FPGA-bitstream and ASIC-hardware Protection,” in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 405–410.

[36] G. Kolhe *et al.*, “Security and complexity analysis of lut-based obfuscation: From blueprint to reality,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[37] A. Sasan *et al.*, “Sequential and Combinational Satisfiability Attacks,” *Encyclopedia of Cryptography, Security and Privacy*, pp. 1–6, 2022.

[38] X. Xu *et al.*, “Novel Bypass Attack and BDD-based Tradeoff Analysis against all Known Logic Locking Attacks,” in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2017, pp. 189–210.

[39] K. Z. Azar *et al.*, “Data Flow Obfuscation: A New Paradigm for Obfuscating Circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 643–656, 2021.

[40] H. M. Kamali *et al.*, “Chaolock: Yet another SAT-hard Logic Locking using Chaos Computing,” in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 387–394.

[41] M. S. Rahman *et al.*, “O’Clock: Lock the Clock via Clock-gating for SoC IP Protection,” in *Design Automation Conference (DAC)*, 2022, pp. 1–6.

[42] K. Z. Azar *et al.*, “Threats on logic locking: A decade later,” in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. ACM, 2019, pp. 471–476.

[43] H. M. Kamali *et al.*, “SCRAMBLE: The State, Connectivity and Routing Augmentation Model for Building Logic Encryption,” in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020, pp. 153–159.

[44] K. Z. Azar *et al.*, “COMA: Communication and Obfuscation Management Architecture,” in *Int’l Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2019, pp. 181–195.

[45] S. Roshanisefat *et al.*, “DFSSD: Deep Faults and Shallow State Duality, A Provably Strong Obfuscation Solution for Circuits with Restricted Access to Scan Chain,” in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[46] B.Yang *et al.*, “Secure scan: A design-for-test architecture for crypto chips,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2287–2293, 2006.

[47] K. Z. Azar *et al.*, “From cryptography to logic locking: A survey on the architecture evolution of secure scan chains,” *IEEE Access*, vol. 9, pp. 73 133–73 151, 2021.

[48] H. M. Kamali *et al.*, “Advances in Logic Locking: Past, Present, and Prospects,” *Cryptology ePrint Archive*, 2022.

[49] B. Yang *et al.*, “Scan based side channel attack on dedicated hardware implementations of data encryption standard,” in *2004 International Conference on Test*. IEEE, 2004, pp. 339–344.

[50] R. Nara, *et al.*, “A scan-based attack based on discriminators for aes cryptosystems,” *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 12, pp. 3229–3237, 2009.

[51] R. Nara *et al.*, “Scan-based side-channel attack against rsa cryptosystems using scan signatures,” *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 93, no. 12, pp. 2481–2489, 2010.

[52] G. Sengar *et al.*, “Secured flipped scan-chain model for crypto-architecture,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2080–2084, 2007.

[53] M. Agrawal *et al.*, “Scan based side channel attacks on stream ciphers and their counter-measures,” in *International Conference on Cryptology in India*. Springer, 2008, pp. 226–238.

[54] S. Banik *et al.*, “Improved scan-chain based attacks and related countermeasures,” in *International Conference on Cryptology in India*. Springer, 2013, pp. 78–97.

[55] S.Banik *et al.*, “Cryptanalysis of the double-feedback xor-chain scheme proposed in indocrypt 2013,” in *International Conference on Cryptology in India*. Springer, 2014, pp. 179–196.

[56] Y. Atobe *et al.*, “Dynamically changeable secure scan architecture against scan-based side channel attack,” in *2012 International SoC Design Conference (ISOCC)*. IEEE, 2012, pp. 155–158.

[57] J. Lee *et al.*, “A low-cost solution for protecting ips against scan-based side-channel attacks,” in *24th IEEE VLSI Test Symposium*. IEEE, 2006, pp. 6–pp.

[58] D. Hely *et al.*, “Scan design and secure chip,” in *IOLTS’04: 10th International On-Line Testing Symposium*. IEEE, 2004, pp. 219–224.

[59] J.Lee *et al.*, “Securing designs against scan-based side-channel attacks,” *IEEE transactions on dependable and secure computing*, vol. 4, no. 4, pp. 325–336, 2007.

[60] X. Chen *et al.*, “Partial scan design against scan-based side channel attacks,” in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1484–1489.

[61] M. Inoue *et al.*, “Partial scan approach for secret information protection,” in *2009 14th IEEE European Test Symposium*. IEEE, 2009, pp. 143–148.

[62] U. J. Botero *et al.*, “Hardware trust and assurance through reverse engineering: A survey and outlook from image analysis and machine learning perspectives,” *arXiv preprint arXiv:2002.04210*, 2020.

[63] D. Zhang *et al.*, “Dynamically obfuscated scan for protecting ips against scan-based attacks throughout supply chain,” in *2017 IEEE 35th VLSI Test Symposium (VTS)*. IEEE, 2017, pp. 1–6.

[64] R. Karmakar *et al.*, “A scan obfuscation guided design-for-security approach for sequential circuits,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.

[65] S. Roshanisefat *et al.*, “RANE: An Open-Source Formal De-obfuscation Attack for Reverse Engineering of Logic Encrypted Circuits,” in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2021, pp. 221–228.

[66] K. Z. Azar *et al.*, “Warm Up before Circuit De-obfuscation? An Exploration through Bounded-Model-Checkers,” in *International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 1–4.

[67] N.Limaye *et al.*, “DynUnlock: Unlocking Scan Chains Obfuscated using Dynamic Keys,” *arXiv preprint arXiv:2001.06724*, 2020.

[68] M. T. Rahman *et al.*, “The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes,” *Int’l Symp. on Hardware Oriented Security and Trust (HOST)*, pp. 1–10, 2018.

[69] X. Wang *et al.*, “Secure scan and test using obfuscation throughout supply chain,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, 2017.

[70] J. Rajendran *et al.*, “Security Analysis of Integrated Circuit Camouflaging,” in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security (CCS)*, 2013, pp. 709–720.