This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

Efficient Key-gate Placement And Dynamic Scan Obfuscation Towards Robust Logic Encryption

Rajit Karmakar, *Student Member, IEEE,* Harshit Kumar, and Santanu Chattopadhyay, *Senior Member, IEEE*

Abstract—Logic encryption has emerged to be a potential solution to the problem of Intellectual Property (IP)-Piracy and counterfeiting. However, in the recent past, several attacks have been mounted on existing logic encryption strategies to extract the secret key. SAT attack, the most predominant one among them, exploits the unprotected Design-for-Testability (DfT) infrastructure as a backdoor to launch attacks on sequential circuits. Protecting the DfT infrastructure is of paramount importance to ensure the security of an Integrated Chip (IC). In this paper, we propose a new logic encryption scheme which dynamically obfuscates the scan operation for an unauthorized attempt of scan access. A detailed security analysis on the proposed secure DfT infrastructure demonstrates its ability to thwart SAT attack without compromising the testability of the design. A methodical key-gate placement strategy enables the proposed scheme to eliminate the leakage of key information through weak key-gate locations, offering protection against path sensitization and logic cone based attacks. Unlike other state-of-the-art SAT preventive schemes, our proposed method does not suffer from poor output corruption, which is a fundamental requirement of a logic encryption scheme.

Index Terms—Hardware Security, IP-piracy and counterfeiting, Logic encryption, Scan Obfuscation, Attacks and Countermeasures

1 Introduction

Modern ICs consist of different functional modules, called IP cores, provided by different IP vendors located worldwide. A system integrator integrates different IPs into a single chip according to the required specification. Such a global scale of the semiconductor supply chain eases the development of complex ICs. However, it introduces several security vulnerabilities like IP-piracy and IC-counterfeiting [1], [2]. Due to the high cost of setting up and maintaining foundries, fabless companies fabricate their designs using the offshore foundries. A rogue agent in an untrustworthy foundry can reverse engineer the GDS-II file to extract the functionality of the design. The reverse engineered design can be overproduced, sold to a rival organization, or may be used to insert Trojans [3]. The semiconductor industry suffers losses in several billions of dollars every year as a result of IP-piracy [4]. Consequently, thwarting IP-piracy has become one of the major objectives of a design engineer.

Logic encryption (also called as logic locking) [2], [5] is a viable and the most widely adopted Design-for-Security (DfS) solution which attempts to prevent IP piracy and IC-overbuilding (other DfS solutions are IC camouflaging [6], split manufacturing [7], watermarking [8], etc.). While split manufacturing or IC camouflaging offer protection against only either untrustworthy foundry or end-users, logic encryption can thwart piracy anywhere in the design flow, except the design house, which is assumed to be trusted entity [9]. In a logic-locked IC, the designer inserts additional logic elements (key-gates) into the circuit-design to conceal its functionality from the attacker. The resulting encrypted circuit exhibits incorrect functionality in the absence of the correct "key-inputs". The encrypted netlist is sent to the foundry for fabrication. After manufacturing, the locked chip returns to the design house, where the designer activates it by applying the secret key. These key-inputs are stored in a tamper-proof memory inside the chip.

Recently, extensive efforts have been devoted towards extracting the secret key of a logic-locked design in an attempt to clone its IP [5], [10], [11]. Despite extensive research, state-of-the-art logic encryption strategies suffer from several limitations, as discussed in Section 3. In this paper, we attempt to address these limitations by proposing a new logic encryption strategy which obfuscates a design in two phases. The first phase finds optimal key-gate locations while the second phase protects the scan data from unauthorized users. A detailed security evaluation of the proposed scheme shows its ability to thwart all the reported attacks on logic encryption.

2 Attacks On Logic Encryption

As shown in Figure 1, the untrusted regime in logic encryption comprises of the foundry and the end-user. We characterize the attacker's resources using the following attack model.

2.1 Attack Model

The attacker can be any rogue entity in the supply chain, post design house, either from the third-party foundry or an end user with access to the advanced reverse engineering tools. For extracting the values of the key-inputs, the attacker requires access to the following:

Rajit Karmakar, Harshit Kumar and Santanu Chattopadhyay are with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, India, 721032 e-mail: rajit@ece.iitkgp.ernet.in, harshitk@iitkgp.ac.in, santanu@ece.iitkgp.ernet.in

An initial version of the paper has been accepted in ISCAS 2018.

This work is partially supported by Dept. of Higher Education, Science & Technology and Biotechnology, Govt. of West Bengal, India and "Synopsys CAD Laboratory Projects (CADL)", sponsored by Synopsys Inc., USA

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

- A **functional IC** $\bar{S}$ which has been activated by the designer using the correct key. It can be obtained from the market. The attacker uses this IC as a black box to get the oracle (input-output pairs).
- A **locked gate-level netlist** S_{lock}, wherein the attacker has the knowledge of the logic encryption scheme and the location of the key gates; however, the key input is unknown. Such a netlist can be obtained from the GDS II file in case of untrusted foundry or by reverse engineering the target IC obtained from the market.

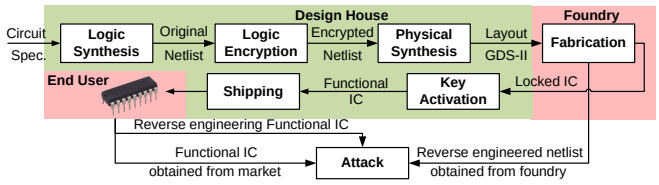


Figure 1: A block diagram of the attack model on Logic Encryption

2.2 Different Types of Attacks

The state-of-the-art attacks on logic encryption can broadly be classified into the following categories.

2.2.1 *Weak Key-gate Locations: A Source of Key Leakage*

Weak key-gate locations expose the encrypted IC to several attacks like path sensitization [5] and logic cone based attacks [10]. A path sensitization attack sensitizes a key-bit to the primary output by placing non-controlling values in the remaining gates of the sensitization path of the key-gate. A logic cone based attack uses divide-and-conquer strategy to reduce the attacker's complexity as brute force attacks can be attempted on the logic cone of the individual outputs.

2.2.2 *Scan-Based DfT : A Threat To Logic Encryption*

Scan-based DfT infrastructure aides in the extraction of key-inputs of a logic-locked design. One necessary requirement of such attacks is access to the scan-chain of the functional IC and the ability to switch between normal and scan mode without any loss of data. SAT [11] and Scan-based circuit partitioning attack [12], [13] fall under this category.

- **SAT Attack [11]:** This attack uses a SAT solver to iteratively search for a set of *distinguished input patterns* ($DIPs$) which prune the key search space. On termination, the attack yields an equivalent set of correct keys. However, SAT solvers require a Directed Acyclic Graph (DAG) *i.e.* a combinational circuit. Performing this attack on sequential circuit requires access to the internal states (flip-flops) of the circuit via scan-chains, essentially converting the circuit to a combinational one.
- **Scan-based Circuit Partitioning Attack [12], [13]:** Scan access drastically reduces the complexity of logic cone based attack by converting inputs and outputs of all the flip-flops to pseudo-primary outputs and inputs, respectively. This allows partitioning the circuit into smaller sub-circuits based on the *Input Cone of Dependency* ($ICOD$) of each flip-flop.

3 Related Works on Logic Encryption

3.1 Key-gate location Based Defense

Initial work on logic encryption, which was found to be insecure, inserts key-gates in the circuit at random locations [2]. Fault analysis (FA) based defense [14] uses VLSI testing principles of fault excitation, propagation, and masking to find the key-gate locations. This ensures high output corruption but fails to avert decryption attacks. Strong logic locking (SLL) [5] increases the dependency among the key-gates through clique-based selection (CBS) and prevents path sensitization attack. However, interdependent key-gate locations do not ensure high output corruption for incorrect keys. Furthermore, the algorithm for clique formation has high time complexity $O(G^2)$, where G is the size of the circuit in terms of number of logic elements. Logic cone based attack [10] can be partially prevented by inserting extra MUXes into the circuit and increasing the number of key-gates in the $ICOD$ of each output [10].

One fundamental limitation of all these defense strategies is that they are tailored to handle a specific type of attack and often select conflicting key-gate locations. This inhibits the designer from clubbing them together to provide defense against all the attacks. Thus, finding out the best locations for the key-gates has remained an open challenge. Furthermore, these strategies are vulnerable to the SAT and scan-based circuit partitioning attacks.

Existing countermeasures against SAT attack can be categorized into three major categories – 1. point function based, 2. cyclic logic encryption based, and 3. scan obfuscation/protection based SAT resilience. The techniques are detailed next.

3.2 Point Function Based SAT Resilience:

These techniques prevent SAT attack by incorporating a point function [1] which exponentially increases the number of DIPs to extract the correct key. SARLock [15], AntiSAT [16], TTLock [17], SFLL-HD [18], SFLL-Flex [18], and SFLL-Fault [19] are different point function based SAT resilience schemes. However, extremely low output corruption is a fundamental limitation of all these techniques. For example, SARLock and TTLock corrupt only one output bit for a single input pattern in the absence of the correct key. SFLL-HD [18], a generalized version of TTLock, ensures output corruption for relatively more number of input patterns (called protected input patterns). However, a wrong key corrupts only one output bit for the protected input patterns. Although SFLL-Flex [18] corrupts multiple output bits for multiple protected input patterns, the higher output corruption comes at the cost of compromised resilience against SAT attack (a linear increase in the number of protected patterns results in a logarithmic decrease in SAT resilience). SFLL-flex strips functionality of a design based on either randomly selected or designer-specified patterns using a time-consuming simulated annealing approach. SFLL-fault bypasses this time-consuming strategy by utilizing fault-injection based VLSI testing principles to identify the most cost-effective parts to strip and the list of protected input

1. A point function is a boolean function which yields the output 1 for exactly one input pattern.

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

cubes associated with it [19]. However, similar to SFLL-flex, SFLL-fault also ends up protecting only a few input patterns/cubes. In addition to the limited output corruptibility, SARLock and AntiSAT are also vulnerable to removal attack [17], bypass attack [20], and signal probability skew attack [21]. Both SARLock and TTLock are also vulnerable to approximate attacks like AppSAT [22] and Double-DIP [23]. SFLL-HD is also vulnerable to functional reverse engineering [24] and FALL [25] attacks.

3.3 Cyclic logic encryption for SAT prevention

SAT solvers cannot handle cycles in the circuit. Cyclic logic locking [26] exploits this phenomenon to introduce dummy cycles in acyclic combinational circuits. The modified circuit becomes SAT-unsolvable, thus cannot be deobfuscated by conventional SAT attack. However, some later attacks [27], [28] could break the dummy cycles and successfully decrypt the secret key. A recent work [29] has strengthened the security of cyclic logic locking by introducing hard to break cycles into the circuit.

3.4 Scan protection for SAT prevention

A recent method [12] prevents SAT attack on scan-based sequential circuits by restricting scan read-out operations upon a switch from functional mode to test mode. This method introduces a special type of scan cells, called Secure Cell (SC), which securely hold the secret key. Three different working modes of the SCs control the entire functionality of the circuit as well as offer support for testing and debugging. This method prevents switching to a shift mode whenever the SCs hold the key values. This ensures that the functional responses, which capture essential oracles regarding the obfuscation key, do not get exposed via the scan path. This operation of blocking the scan read-out mitigates the SAT and other oracle guided attacks, which require scan access. Although this method restricts the observability of the functional response, it does not block the controllability of the scan chain. This controllability feature, offered by the secure cell strategy, is exploited by a recently proposed *shift & leak attack* [30], which can successfully break its defense.

Another SAT preventive logic obfuscation technique, named *Encrypt Flip-Flop*, has been proposed in [31], which inserts MUX key-gates between the scan cells to obfuscate the scan data. The select lines of the MUXes act as key inputs. Depending upon the secret key, the scan-in and scan-out vectors get modified during the shift operation, hiding the actual scan-data from the adversary. Unlike [12], *Encrypt Flip-Flop* restricts both controllability and observability of the scan chain to prevent SAT attack. Although this static scan obfuscation technique can thwart the basic SAT attack, it is vulnerable to an advanced SAT attack, called ScanSAT [32]. ScanSAT is capable of modelling the obfuscated scan chain as a logic locked combinational circuit. A subsequent SAT attack can extract the secret key.

A dynamic scan obfuscation approach has been proposed in [33]. Although the primary focus of this strategy is to protect the secret encryption key of an embedded crypto-module, its scan obfuscation property can be used to thwart SAT attack on logic locking. Similar to *Encrypt Flip-Flop*, this strategy also inserts XOR gates between the scan cells to obfuscate the scan data. However, dynamic scan obfuscation does not obfuscate the functional operation. Therefore, a standard logic locking technique has to be integrated with it for IP protection. Since the scan obfuscation key does not affect the functional operation, this technique gets the unique privilege of updating the key periodically. The dynamic change in scan obfuscation key makes it even harder to deobfuscate the scan-out data. However, as mentioned in [32], this method might also be vulnerable to ScanSAT attack if the key update cycle is longer than the number of DIPs required to break the circuit.

Since most of the scan obfuscation guided approaches prevent SAT attack by limiting scan access, another line of research has been initiated on launching the SAT attack on sequential circuits in the absence of scan access [34], [35]. These attacks mainly rely on unrolling a sequential circuit in time-frames to obtain the combinational counterpart of the circuit, on which the SAT attack is applicable. In [34], it has been shown that this type of attack is successful only on small sequential circuits. Although a recent strategy [35] attempt to accelerate the attack on sequential circuits, it suffers from scalability issues while applied on larger circuits.

The major takeaway from the above discussion can be summarized as follows:

- Although path sensitization and logic cone based attacks are relatively weak compared to the SAT attack, one cannot ignore these attacks since SAT preventive solutions may also leak the secret key through weak key-gate locations. Therefore the key-gate placement plays a pivotal role in logic encryption.
- State-of-the-art point function based SAT-resilience techniques do not provide high output corruption for incorrect key-inputs, thus fail to satisfy the fundamental requirement of logic encryption.
- None of the point function based defense strategies offers full protection against all the attacks.
- Mitigating SAT-attack is often insufficient to completely restrict the leakage of key information through the scan path, which depicts the requirement of scan protection.
- State-of-the-art scan obfuscation strategies also suffer from various attacks.

4 Contribution of the Paper

Shortcomings in the existing literature, as described in Section 3, have motivated us to propose a new logic encryption scheme which fulfills all the fundamental criteria of a secure logic encryption algorithm at realistic overheads. The salient contributions of the proposed technique are as follows.

- We insert key-gates in optimal locations maximizing the complexity of path sensitization and logic cone based attacks and simultaneously ensuring high output corruption for wrong keys. Unlike other key-gate placement techniques like SLL [5] and FA [14], our method does not suffer from conflicting key-gate locations to fulfill the fundamental criteria of logic encryption. The complexity of our key-gate placement strategy is $O(G \times O)$ compared to $O(G^2)$

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

of SLL, where G is the number of logic elements and O is the number of outputs of the circuit.
- We propose to dynamically obfuscate the scan operation for any attempt of unauthorized scan access. This prevents a rogue agent from having controllability and observability over the internal states of the circuit. The proposed method defeats not only the conventional SAT attacks but also the advanced ones like ScanSAT [32], which is capable of breaking static obfuscation [31] in the scan-chain.

We have performed a detailed security analysis, exhibiting the infeasibility of unrolling a realistic sequential circuit, eliminating the possibility of SAT attack on a logic-locked IC without having proper scan access. An in-depth security analysis on the proposed scheme shows its ability to thwart other attacks like ScanSAT, removal attacks, and so on. A detailed comparison of our technique with different state-of-the-art logic encryption and scan protection schemes has been performed to demonstrate various advantages of the proposed strategy over the existing ones. The test flow and key activation process show that the proposed dynamic scan obfuscation does not hamper the testability of the design.

A preliminary version of this work, capable of preventing path sensitization and logic cone based attacks, as well as, producing high output corruption, has been presented in [36]. However, inability to thwart the SAT attack, the most potent one on logic encryption, is the biggest limitation of [36]. As an improvement, the current work offers protection against SAT and other scan-based attacks on logic encryption strategy. The key-gate location selection algorithm of [36] also suffers from high time complexity, limiting its application to only small and medium-sized circuits. This drawback has also been addressed in this paper by significantly improving the key-gate selection strategy, ensuring its application on realistic circuits.

The rest of the paper is organized as follows. Section 5 presents our proposed key-gate location selection strategy, which is a two-step process. Section 6 presents the proposed scan obfuscation technique, which helps us to combat different variants of the SAT attack. The entire test flow and the key activation process are demonstrated in Section 7. Section 8 presents an in-depth security analysis against all the state-of-the-art attacks. Section 9 presents the results of our experimentation, including a case study on RISC-V CPU core. A detailed comparative analysis with all the state-of-the-art logic encryption techniques is presented in Section 10. Finally, Section 11 concludes the paper.

5 Phase I: Key-gate Location Selection

The first phase of the proposed encryption strategy is a two-step process of selecting optimal locations for the key-gates. The first step ensures protection against path sensitization and logic cone based attacks while the second step guarantees high output corruption for incorrect keys.

5.1 Step I: Measures to Prevent Path Sensitization and Logic Cone Based Attacks

Let us consider, a circuit having M inputs, O outputs, and G logic elements. Our strategy of selecting the best locations for encryption employs the following steps:

- We first find the logic elements present in the input cone of dependency ($ICOD$) of all the O outputs of the circuit. The process can be explained using Figure 2, which depicts the $ICOD$s of all the outputs using a graph-based representation. All the logic elements ($G1$ to $G10$) and the outputs ($O1$ to $O6$) are represented by nodes in the graph. An edge between an output O_j and a logic element G_i signifies $G_i \in ICOD(O_j)$.
- We select $O_{overlap}$ outputs ($O_{overlap} \leq O$) with the largest number of logic elements ($G_{overlap}$) in their overlapping input cone of dependency ($ICOD_{overlap}$). In this example, the outputs $O2$, $O3$, $O4$, and $O5$ form $O_{overlap}$, and the logic elements $G3$, $G4$, $G5$, $G6$, $G7$, and $G8$ form $G_{overlap}$. It may be noted that $\forall \bar{G} \in G_{overlap}$ and $\bar{O} \in O_{overlap}$, $\bar{G} \in ICOD(\bar{O})$.
- Inserting key-gates in $G_{overlap}$ region ensures that each output of $O_{overlap}$ includes all the key-gates in its $ICOD$. However, some logic elements of $G_{overlap}$ may also affect the outputs other than those in $O_{overlap}$. $G3$ is such a logic element in Figure 2, which affects the output $O1$ apart from the outputs of $O_{overlap}$. As $ICOD$ of $O1$ contains no other key-gates, key-gate inserted at $G3$ can easily be extracted through $O1$. Therefore, $G3$ is a weak location to insert a key-gate. We identify such weak locations in $G_{overlap}$ and remove them to form a new set of logic elements called G_{strong}, which affects only the outputs of $O_{overlap}$. In Figure 2, the logic elements $G4$, $G5$, $G6$, $G7$, and $G8$ form G_{strong}. We restrict the insertion of the key-gates in the G_{strong} region only.

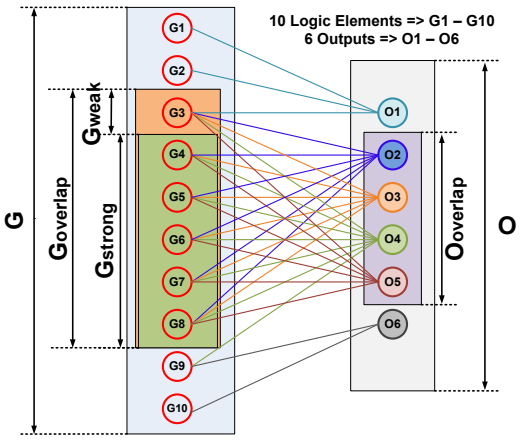


Figure 2: An example of finding overlapping input cone of dependency

5.2 Step II: Ensuring High Output Corruption for Incorrect Keys

Inserting key-gates anywhere in the G_{strong} region does not guarantee output corruption for incorrect keys, as the effect of the key may not necessarily propagate to the outputs. Therefore, to ensure high output corruption for wrong keys, we identify the suitable locations to insert key-gates in

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

the G_{strong} region. We use VLSI testing principles to find potential locations. Application of incorrect key is analogous to the excitation of either a stuck-at-0 (s-a-0) or stuck-at-1 (s-a-1) fault. Fault simulation tool HOPE [37] guided fault impact metric [14] identifies several locations in the circuit. It necessarily propagates the fault to the output and corrupts the requisite number of output bits for most of the applied input patterns. From a set of test patterns, this fault impact metric computes the following parameters:

1) N_0P_0: the number of patterns that detect a s-a-0 fault at the output of any gate.
2) N_0O_0: the total number of output bits that are corrupted by this s-a-0 fault.
3) N_0P_1: the number of patterns that detect a s-a-1 fault at the output of any gate.
4) N_0O_1: the total number of output bits that are corrupted by this s-a-1 fault.

The final fault impact of each gate is calculated as

$$Fault\ Impact = N_0P_0 \times N_0O_0 + N_0P_1 \times N_0O_1 \quad (1)$$

We rank each gate in the G_{strong} region based on the fault impact metric and select K locations with the highest values of fault impact metric. Algorithm 1 summarizes the proposed key-gate placement strategy.

Input : Original Netlist; Key size (K);
Output : Encrypted Netlist;
begin
  for *Each of the O outputs* **do**
    Find the logic elements present in its input cone of dependency ($ICOD$);
  Form a set of $O_{overlap}$ outputs with the largest number of logic elements ($G_{overlap}$) in the overlapping input cone of dependency ($ICOD_{overlap}$);
  for *each element of* $G_{overlap}$ **do**
    if *The logic element affects any output other than the* $O_{overlap}$ *outputs* **then**
      Include the logic element in the set G_{weak};
  Form a new set G_{strong} = $G_{overlap}$ - G_{weak};
  Rank all the logic elements of G_{strong} based on the fault impact metric;
  Select K locations from G_{strong} with the higher values of fault impact metric and insert key-gates at each of these locations;

Algorithm 1: *Key-gate Placement Strategy*

6 Phase II: Prevention of the SAT Attack

The first phase of the proposed logic encryption technique offers protection against all the vulnerabilities of weak key-gate locations. The second phase of our scheme helps us thwarting the powerful SAT attack. Before proposing our strategy, we first formalize the SAT attack.

The problem *DEC-IC*, of decrypting a logic-locked IC, can be formalized as follows.

DEC-IC: Given an encrypted IC, let *M* be the number of primary inputs, *O* be the number of primary outputs, and *K* be the number of key-inputs (key gates) in the circuit. An encrypted sequential circuit C_{lock} is represented by the mapping $Y = C_{lock}(I, K_X)$, where $I \in \{0,1\}^M$ represents the input, $Y \in \{0,1\}^O$ represents the output, and $K_X \in \{0,1\}^K$ represents an assignment of inputs to the key-gates. Additionally, F is the number of flip-flops in C_{lock}. Let the input-output relation of an unencrypted circuit be modeled by the black box $\bar{C}$, given by $Y = \bar{C}(I)$. The attacker's objective is to find the correct assignment of key-inputs K_X^* to the key gates, satisfying the following relation,

$$C_{lock}(I, K_X^*) = \bar{C}(I), \qquad \forall I \in \mathcal{I}, \quad (2)$$

where $\mathcal{I}$ is the universal set of input sequences. It should be noted that K_X^* may not be a singleton, that is, the correct key for an encrypted circuit may not be unique.

Complexity of DEC-IC: For a given set of input sequence **I**, the certificate for the problem *DEC-IC* is an assignment K_X^* such that $C_{lock}(I, K_X^*)$ concurs with the black box $\bar{C}$ on all input sequences in **I**. The certificate itself can be generated in $O(1)$ time while verifying its correctness can be performed in $O(|C_{lock}||I|)$. Since digital sequential circuits can be modeled as finite systems, given an input sequence $i \in$ **I**, the corresponding output is obtained in time which is polynomial in the size of the circuit. Hence, the stated problem is in NP and can be reduced to CNF-SAT which is complete for NP. The attacker's objective is to find the *sequence of discriminating inputs* **I** which is sufficient to determine the value of key-inputs.

Consequently, the attacker can use an off-the-shelf SAT solver for creating a SAT instance. The sequentiality of the circuit has to be circumvented for formulating a SAT problem, which requires modeling the circuit as a Directed Acyclic Graph (DAG). This can be achieved through one of the following ways:

- **Gaining access to the scan chain:** This enables the attacker to access the outputs of the flip-flops which contain information about their respective cone of dependencies. The circuit essentially behaves as a combinational circuit wherein the flip-flops behave as pseudo inputs/outputs.
- **Unrolling the circuit using time-frame expansion:** A sequential circuit is converted to its combinational counterpart by unrolling the circuit $d+1$ times where d is the minimal number of steps required to reach all states in the FSM modeling of the circuit.

The easiest possible way of converting a sequential circuit to its combinational counterpart is exploiting the unprotected scan-based DfT infrastructure. Our proposed defense strategy against SAT attack relies on protecting the scan operation, thus ensuring that no secret key information gets leaked via the scan path. Subsequently, we show that it is infeasible to unroll a realistically large sequential circuit and thereby extracting the key using SAT attack.

6.1 Scan Protection for SAT Prevention: The Proposed Defense Strategy

Since scan-chain is a primary source of key leakage, it must be protected. Destroying the scan-access after structural and functional tests is certainly one option to prevent key leakage via the scan path. However, it might not be wise, since it compromises the in-field testing and debugging capabilities of the functional IC. The other alternative is to have secure scan access, which can restrict unauthorized use

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

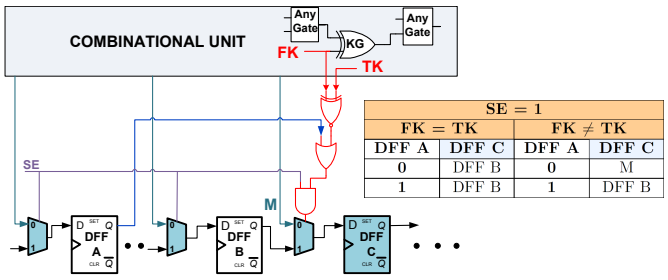
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

of scan chains without hampering the testability features. To prevent the misuse of the scan data, we integrate a test authentication step with our proposed logic encryption scheme, which grants scan access only to the verified test engineers. To get access to the scan chain of a logic-locked design, encrypted with a K-bit $Functional\ Key$ (**FK**), a user has to apply a K-bit $Test\ Key$ (**TK**). The scan chain operates correctly only if **TK** satisfies the relation $TK_i = FK_i,\ \forall\ i \in K$. Otherwise, scan data gets corrupted.

Working Strategy: The proposed test authentication based secure scan access strategy works as follows.

In traditional scan-based DfT infrastructure, internal flip-flops of the circuit are replaced with scan cells (SFFs). Each scan cell comprises of a MUX and a flip-flop. The scan cells are stitched together to form scan chains. Depending upon the value of a control input, named Scan Enable (SE), these scan cells either capture the functional response from the combinational part of the circuit or receive data from the previous scan cells. The proposed scheme encrypts the SE inputs of some scan cells to lock the scan operation. Only a correct *Test Key* (**TK**) can unlock the scan chain and restore the proper scan operation.

We explain the functionality of the proposed scan encryption architecture using the example of Figure 3. The scan control line of *DFF C* is encrypted using the proposed method. We call *DFF C* as **Encrypted** SFF. Instead of scan enable input SE, the *Test Key* (**TK**) controls the scan operation of *DFF C*. The other two flip-flops *DFF A* and *DFF B* are standard SFFs of the design, whose scan operation are controlled by SE.



SE = 1			
FK = TK		FK ≠ TK	
DFF A	DFF C	DFF A	DFF C
0	DFF B	0	M
1	DFF B	1	DFF B

Figure 3: Encrypting the DfT infrastructure

In functional mode of operation, *DFF C* captures the response from the combinational part of the design (line M), while in scan mode, it receives input from either line M or *DFF B* depending upon the correctness of **TK**. If **TK** = **FK**, *DFF C* behaves like a normal SFF and receives input from *DFF B*. If **TK** $\neq$ **FK**, in each clock cycle, the value of a random scan cell (*DFF A* in this case) dynamically decides whether *DFF C* receives input from *DFF B* or line M.

In any clock cycle i, for SE = 1, and **TK** $\neq$ **FK**:

- If *DFF A* = 0, *DFF C* gets the input from M in the $i + 1^{th}$ clock cycle.
- If *DFF A* = 1, *DFF C* gets the input from *DFF B* in the $i + 1^{th}$ clock cycle.

As the content of *DFF A* changes dynamically during scan shifting, the input source of *DFF C* also varies dynamically between the line M and *DFF B*, which hampers the scan operation and corrupts the scan-out response.

Figure 4 shows an example scan chain, consisting of 12 scan flip-flops, encrypted using the proposed technique. The combinational part of the circuit is encrypted with five key-gates. Since the length of the *Test Key* must be equal to the length of the *Functional Key*, we encrypt the scan control lines of five flip-flops of the scan chain, say *DFF 3*, *DFF 4*, *DFF 7*, *DFF 9*, and *DFF 10*. These scan cells are the **Encrypted** SFF. The outputs of five random flip-flops, say *DFF 11*, *DFF 1*, *DFF 5*, *DFF 2*, and *DFF 6* affect the scan controls of these **Encrypted** SFFs, whenever a wrong *Test Key* is applied in scan mode. To execute the proper scan operation, each bit of the *Test Key* must be the same as the corresponding bit of the *Functional Key*. Therefore, only an authenticated test engineer, having access to the correct *Test Key* gets the privilege to correctly operate the circuit in scan mode. An adversary, without the knowledge of the correct key, gets only corrupted scan-out response.

6.2 Disruption of Scan Operation for Wrong *Test Key*

Since a sequential circuit must be converted to its equivalent combinational counterpart before launching SAT attack, the attacker must first unlock the scan chain to restore the scan operation. Without the knowledge of the correct *Test Key*, the attacker is left with the choice of random guessing. In this section, we demonstrate how a wrong *Test Key* corrupts the scan operation.

6.2.1 Case I

Let us assume that the attacker performs scan operation on the proposed scan architecture of Figure 4. The attacker's objective is to control the scan chain to upload a known input vector into it and observe the functional responses captured in the scan cells. Let the scan chain captures a functional response **"101100010110"**, which the attacker would like to extract as it is. At the same time, he/she wants to upload the input vector say **"100011010010"**. Let us assume that the random *Test Key*, applied by the attacker, has three wrong key-bits: TK1, TK3, and TK4. Figure 5 shows the disrupted scan operation by this incorrect *Test Key*. The first column of the figure marks the different clock cycles. The subsequent 12 columns demonstrate how the value of each scan cell changes over the clock cycles, during the scan shifting under the applied *Test Key*. For better readability, each pair of the *Encrypted* and *Affecting* flip-flops have been highlighted using the same colour. For example, the values of the *Encrypted* $SFF\ DFF3$ have been marked in red, while the corresponding *Affecting* $SFF\ DFF11$'s contents are put in a red box. As can be observed from the figure, at 0^{th} clock cycle, the scan chain holds the captured functional response **"101100010110"**, that comes from the combinational unit. The attacker would expect to observe this response at the scan-out port. Similarly, the attacker uploads the desired input vector **"100011010010"** from the scan-in port and expects that after 12^{th} clock cycle, this input vector gets uploaded into the scan chain unchanged.

Now, let us observe the scan shifting. Since FK2 = TK2 and FK5 = TK5, *DFF 4* and *DFF 10* remain unaffected and always receive inputs from their previous SFFs, that is, *DFF3* and *DFF9*, respectively. However, due to wrong *Test Key* inputs, the inputs to SFFs: *DFF3*, *DFF7*, and *DFF9* become unpredictable and dependent on their corresponding

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

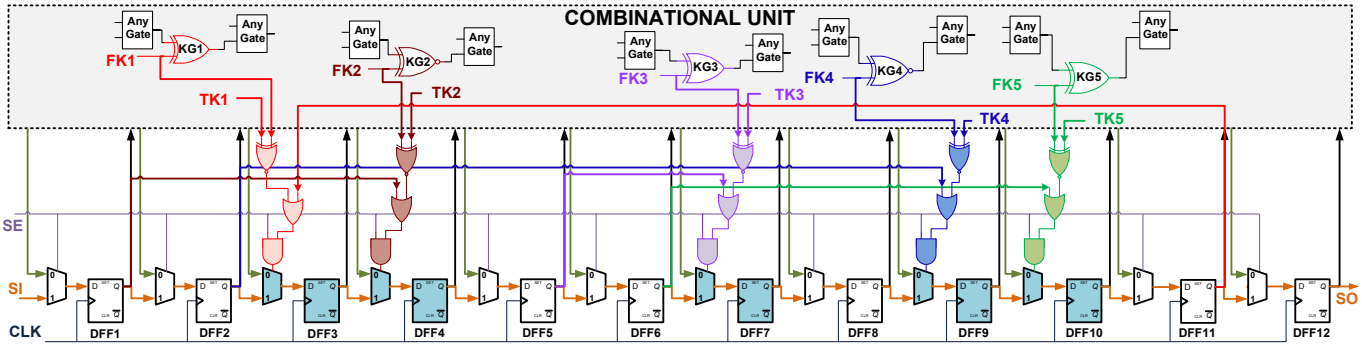


Figure 4: An example scan chain encrypted using the proposed DfT infrastructure

Affecting SFFs, that is, *DFF 11*, *DFF 5*, and *DFF2*, respectively. For example, in the initial two clock cycles, *DFF11* holds a value of '1'. Therefore, despite $FK1 \neq TK1$, data propagates from *DFF2* to *DFF 3*, which has been indicated using diagonal arrows from the values of $DFF2$ to $DFF3$. However, for the next 9 clock cycles, the value of *DFF11* remains '0', which forces *DFF3* to receive input from its combinational part instead of *DFF2*. Since the captured value of the functional response bit, connected to $DFF3$, is '1', for all these 9 clock cycles, $DFF3$ will hold the value '1' (indicated by the vertical down arrows from the 0^{th} clock cycle to the respective clock cycles). During these nine clock cycles, data from the previous flip-flops cannot get shifted further in the scan chain, resulting in a permanent loss of these data-bits. Again, in the 11^{th} clock cycle, $DFF11$ switches to '1', which allows $DFF3$ to receive input from $DFF2$. Similarly, the input source of $DFF7$ and $DFF9$ change dynamically over the clock cycles, depending upon the values of $DFF5$ and $DFF2$, respectively. Since the values of the *Affecting* flip-flops are also unknown to the attacker, the entire scan shifting operation becomes completely unpredictable. As a result, when the functional response is finally observed at the scan-out port, it gets completely corrupted and becomes **"000000000110"**. Similarly, the test vector **"100011010010"** to be scanned-in, also gets corrupted and modified to **"100111110011"**, when finally loaded into the scan chain.

Clk Cycle	#DFF 1	2	3 FK1 ≠ TK1	4 FK2 = TK2	5	6	7 FK3 ≠ TK3	8	9 FK4 ≠ TK4	10 FK5 = TK5	11	12
0	1	0	1	1	0	0	0	1	0	1	1	0
1	0	1	0	1	1	0	0	0	0	0	1	1
2	1	0	1	0	1	1	0	0	0	0	0	1
3	0	1	1	1	0	1	1	0	0	0	0	0
4	0	0	1	1	1	0	0	1	0	0	0	0
5	1	0	1	1	1	1	0	0	0	0	0	0
6	0	1	1	1	1	1	1	0	0	0	0	0
7	1	0	1	1	1	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1	1	0	0	0	0
9	0	1	1	1	1	1	1	1	1	0	0	0
10	0	0	1	1	1	1	1	1	1	1	0	0
11	0	0	1	1	1	1	1	1	0	1	1	0
12	1	0	0	1	1	1	1	1	0	0	1	1

Figure 5: An example of scan operation in the DfT infrastructure of Figure 4, under a random wrong key (Case I)

6.2.2 *Other Possible Cases*

We have also considered three other cases, where we have applied different *Test Keys*, different input test vectors and different initial scan chain values. Figure 6 shows the scan-out responses and uploaded test vectors by varying the incorrect *Test Key*, initial scan contents, and input test vectors.

Case	Wrong Key	Initial Scan Content	Input Test Vector	Scan Out Response	Uploaded Test Vector
Case I	FK1 ≠ TK1 FK3 ≠ TK3 FK4 ≠ TK4	101100010110	010010110001	000000000110	100111110011
Case II	FK2 ≠ TK2 FK4 ≠ TK4	101100010110	010010110001	010000000110	100111010011
Case III	FK2 ≠ TK2 FK4 ≠ TK4	101100010110	101010101010	000100000110	010111110101
Case IV	FK2 ≠ TK2 FK4 ≠ TK4	011001101011	010010110001	101101101011	100000011100

Figure 6: Variation of scan-out response and uploaded test vectors with the variation of wrong *Test Key*, initial scan contents, and input test vector

6.2.3 *Observations*

It can be inferred from Figure 6 that all the three parameters, *Test Key*, initial scan content, and the input test vector affect the scan operation in the absence of correct *Test Key*. A change in any one of these three parameters alters the observed scan-out response and the final uploaded test vector. The dependency on these three parameters (the initial values of the scan cells are also not in control of the attacker) leaves no clue for an attacker to guess the correctness of a random *Test Key*.

7 Test Flow and Key Activation

Post fabrication, ICs go through extensive testing before being released in the market. These tests include:

- **Manufacturing test:** It is a structural test which comprehensively checks a circuit for stuck-at and transition faults [38]. Functional operation of the chip is not required, thus it can be conducted on a locked IC considering any random key. In general, manufacturing test is performed in the foundry which is assumed to be insecure.
- **Functional test:** It checks for the correct functionality of the design. Thus, the chip must be activated before conducting this test, therefore, the test is performed in the design house in a secure environment.

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

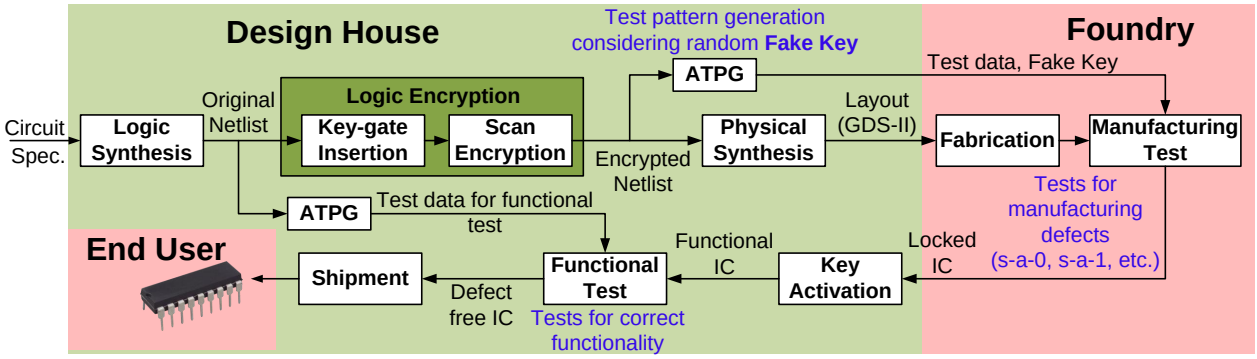


Figure 7: The entire design and test flow of our proposed scheme

We explain the steps involved in IC-testing using the flow chart in Figure 7. The designer generates test patterns for the manufacturing test considering the key-inputs as primary inputs. Since manufacturing tests do not require an activated functional chip, the designer uses an incorrect key (*Fake Key*) while generating the test patterns for this test. The *Fake Key*, test patterns, and corresponding outputs are shared with the foundry, which they utilize for conducting manufacturing tests. The proposed DfT architecture supports full scan-based manufacturing test without sharing the correct *Functional Key* with the foundry. To achieve this, the *Fake Key* is used as the *Test Key* which grants the testing agent full access to the scan-chain. Since, *Fake Key* $\neq$ *Correct Functional Key*, functional performance of the IC remains obfuscated.

After the manufacturing test, the chip is sent back to the design house where the IC is activated with the correct *Functional Key* and functional test is performed on the activated IC to check its functionality. The entire process of key activation and functional test is performed in a secure environment in the design house.

In-field test: Support for in-field testing is crucial for mission-critical devices. Our proposed DfT infrastructure offers full support for in-field testing and debugging only for the authenticated test engineers. An authorized test engineer, having access to the correct *Test Key*, can unlock the scan obfuscation and perform any required maintenance, debug, or diagnosis of the chip in the field. Performing in-field test on the proposed DfT infrastructure requires no modification in the test generation process, a test authentication alone is sufficient.

8 Security Analysis

In this section, we evaluate the resilience of the proposed scheme against different state-of-the-art attacks. First, we show how the proposed key-gate placement strategy offers protection against path sensitization and logic cone based attacks. Subsequently, we examine the security of our strategy against SAT, removal, and other advanced attacks like ScanSAT and Reset-and-SAT attack.

8.1 Security Evaluation Against Logic Cone Based Attack

Logic cone based attack targets outputs with the minimum number of key-gates in their $ICOD$. Inserting key-gates in the G_{strong} region guarantees that the $ICOD$ of any output contains either all or none of the key-gates, eliminating the brute force search by logic cone based attack. In Figure 2, the $ICOD$s of $O1$ and $O6$ contain no key-gate, while each of the other outputs contains all the key-gates in its $ICOD$. Complexity of the logic cone based attack for the proposed scheme is $O(2^{K+M1})$, where K is the number of key-bits and $M1$ is the number of primary inputs affecting the G_{strong} region. This is practically in-feasible for a reasonable value of K.

8.2 Security Evaluation Against Path Sensitization Attack

Protection against path sensitization attack demands high interdependency among all the key-gates. Before examining the dependency among the key-gates, we formalize the definition of dependency.

Dependency: A key K_1 is dependent on another key K_2 if the extraction of K_1 through an output, employing path sensitization attack, requires the knowledge of K_2. Consequently, K_1 cannot be extracted unless the correct key is applied to the key-gate K_2.

We consider the example shown in Figure 8(a). Sensitizing the key K_1 to output O requires fixing the line m to a known value by using the PIs. Next, we feed the same input pattern to the functionally activated chip which reveals the correct value of K_1.

We examine the position of another key-gate $KG_2 \in ICOD(O)$. Relative to the location of KG_1, there can be three possible positions of KG_2:

- **Fan-in cone of the line** m**:** In this case, line m cannot be fixed to a known value without the knowledge of the key K_2. Thus, K_1 cannot be sensitized to the output (Figure 8(b)).
- **Fan-out cone of** KG_1**:** In this case, the output of KG_1 cannot be propagated to the output O unless K_2 is known. Furthermore, KG_1, $KG_2 \in ICOD(O_{overlap})$. If KG_2 is present in the fan-out cone of KG_1, it will affect the sensitization of K_1 to all the $O_{overlap}$ outputs. Since KG_1 affects only the $O_{overlap}$ outputs, K_1 cannot be sensitized without the knowledge of K_2 (Figure 8(c)).
- **Any other location:** In this case, the effect of K_2 converges to the output O. Therefore, K_1 cannot

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

be sensitized to the output unless K_2 is correctly configured (Figure 8 (d)).

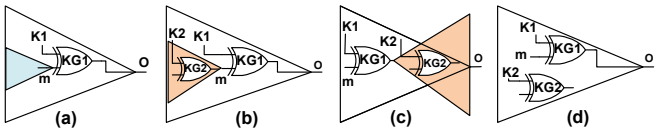


Figure 8: Different possible locations of key-gates in $O_{overlap}$

This analysis is valid for all $KG_i \in ICOD(O)$. Therefore, a key-input K_i is dependent on any other key-input K_j present in the G_{strong} region. Fixing all such K_js to their correct values requires brute force search with search space size of $O(2^{K-1})$, which is in-feasible for a reasonable key-size K. Inserting key-gates in the G_{strong} region is sufficient to mitigate path sensitization attack. Unlike SLL [5], G_{strong} can be found in time $O(G \times O)$. This drastically reduces the execution time of the algorithm.

8.3 Resilience to Satisfiability Attacks

The proposed defense encrypts the scan-chain, preventing a rogue agent from having *controllability* and *observability* over the internal states of the circuit. Since a rogue entity does not have access to the correct *Test Key*, the scan operation gets disrupted. As explained in Case I (Figure 5), the scanned out test vector "000000000110" is inconsistent with the captured functional response "101100010110", hampering the *observability* of the internal states. The restricted observability inhibits the attacker from extracting the important oracle regarding the correct key, captured in the internal flip-flops during the functional operation. Most importantly, unlike scan obfuscation guided schemes [31], [33], the proposed method does not obfuscate the scan data. Instead, a scan shifting, under a wrong *Test Key*, completely corrupts the functional response. It may be noted that there is a fundamental difference between scan obfuscation and scan data corruption. While the original functional response can still be retrieved by deobfuscating the obfuscated response, it can never be retrieved from a corrupted response. For example, the value '1' of DFF 8 (Figure 5), at 0^{th} clock cycle, captures the information of the keys located in its logic cone. However, this information gets lost in the next clock cycle when it passes through the encrypted scan cell $DFF9$, which is configured with a wrong *Test Key* bit. Similarly, the key information, captured in $DFF3$ and $DFF4$, also get lost and never recovered when they pass through $DFF7$ and $DFF9$, in the 4^{th} and 5^{th} clock cycles, respectively. SAT attack cannot use the obtained corrupted functional responses for eliminating the wrong keys.

Furthermore, without the availability of the correct *Test Key*, the attacker cannot control the test vector, which is uploaded, precluding the *controllability* of the internal states as well. Therefore, the attacker cannot exploit the scan-based DfT infrastructure for procuring the oracle and launching SAT attack. This can be considered analogous to having no access to the scan data. Therefore, the attacker is left with the sole option of unrolling the circuit. We elaborate on this procedure in the following section.

8.3.1 *Bounded Model Checking (BMC) Based Attack*

BMC-based attack relies on unrolling the circuit in time frame expansion to launch the SAT attack [34]. In BMC, the system properties are specified using temporal logic. This is achieved by specifying the system using propositional linear temporal logic (LTL) which inherits boolean variables and operators from propositional logic. It enables one to check the temporal properties of a finite system (digital sequential circuit) which is analogous to unrolling the circuit. Contrary to the Binary Decision Diagram (BDD)-based model checking, BMC leverages the success of SAT in solving boolean formulas for tackling the problem of state space explosion. However, satisfiability testing for a boolean formula is an NP-complete problem and has a limited capacity due to exponential dependency in the worst case.

We use the attack framework developed in [34] to test the defense. To start with, we explain the fundamentals necessary to comprehend our analysis and refer the reader to [34] for further details. The following semantics, establishing the FSM, are used in defining the Kripke structure:

- $\mathcal{K}_\mathcal{X} = \{0,1\}^K$ is the set of all possible key-input values to C_{lock}.
- $\mathcal{S} = \{0,1\}^F$ represents all combinations of state in the FSM $i.e.$ all possible values of flip-flops.
- $\mathcal{I} = \{0,1\}^M$, and $\mathcal{O} = \{0,1\}^O$ represent all possible combinations of input and output vectors respectively.
- $\delta_k : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S}$ and $\sigma_k : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O}$ are the state transition function and the output function respectively, for an assignment of key $k \in \mathcal{K}_\mathcal{X}$.
- $s_0 \in S$ is the initial state.

We build an FSM modeling the state transition relation of C_{lock} using the Kripke structure represented by quadruple $M = (S, I, T, L)$ where:

- $S = \{s_1, K_1, s_2, K_2, i\}$ represents the finite set of states: (s_1,s_2) $\in \mathcal{S}$, (K_1,K_2) $\in \mathcal{K}_\mathcal{X}$, and $i \in \mathcal{I}$. It means that for every key-input K_1 and K_2, there are two independent states in M, resulting from the transition dictated by the respective key-inputs. Moreover, these key-inputs remain fixed until the checker traverses the diameter of the FSM.
- $I = \{s_0, K_1, K_2, i\}$ represents the initial set of states: $\forall i \in \mathbf{I}$, $C_{lock}(i,K_1) = C_{lock}(i,K_2) = \bar{C}(i)$. It can be intuitively perceived that the correct assignment of key-gate should agree with the black-box. This prunes down the initial search-space for key-inputs.
- $T \subseteq S \times S$ represents the transition relation: $\forall\{s_1, K_1, s_2, K_2, i\} \in S$, $\exists\{s_1^t, K_1, s_2^t, K_2, i^t\} \in S$ such that $s_1^t = \delta_{K_1}(s_1, i)$, and $s_2^t = \delta_{K_2}(s_2, i)$. Given an input $i \in \mathcal{I}$, the *next state* s_1^t and s_2^t of the respective FSM is determined by the key-inputs K_1 and K_2 respectively.
- The labeling function is $L : S \rightarrow \{equiv\}$, where the atomic proposition $\{equiv\}$ is defined as: $L(s_1, K_1, s_2, K_2, i) = \{equiv\}$ if $\sigma_{K_1}(s_1, i) = \sigma_{K_2}(s_2, i)$, and $L(s_1, K_1, s_2, K_2, i) = \phi$ otherwise.

We define $M| = \mathbf{G}\ equiv$ such that the Kripke structure M satisfies the property $equiv$ over all initialized paths as stated above. A counterexample to $M| = \mathbf{G}\ equiv$ returns

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

an input sequence $\mathbf{I} \in \mathcal{I}$, which eliminates at least one of the key candidates K_1 or K_2 and, therefore, acts as a discriminating input sequence by definition. The sequence $\mathbf{I}$ is updated with this counterexample and the process is repeated. The attack terminates when there is only one candidate key left or the remaining candidate keys are combinationally equivalent.

Experimental Setup and Results:

ISCAS'85 and ITC'99 benchmarks were encrypted using the proposed defense as stated in Algorithm 1 and elaborated in Table 2. Since PHASE II of the defense encrypts the scan-chain, the attack simulations were run assuming no scan-access. The transformation for modelling XOR/XNOR key-gates as camouflaged NAND/NOR gates was performed in ≈ 150 lines of Python code. This enables us to simulate the model checking based attack [34] which performs decamouflaging on sequential circuits without scan access. The BMC based attack framework is in C++ and uses NuSMV as the back-end model checker [39]. The attacks were launched on the aforementioned netlists on a Intel(R) Core i5-3470 processor. A time-out of 12 hours was assumed for the attack simulation. We present the results of the attack in Table 1.

Table 1: Bounded Model Checking decryption results on ISCAS'89 and ITC'99 benchmarks (CR: Model checker crashed, TO: Time Out)

Benchmark	Size	Time (s)	Benchmark	Size	Time (s)
s13207	8589	1444	b17	32192	CR
s15850	10306	6488	b18	114561	CR
s38584	20679	CR	b19	231266	CR
s38417	23815	CR	b20	20172	CR
b14	10012	21031	b21	20517	CR
b15	8816	TO	b22	29897	CR

The experimental results show that the largest circuit on which the attack is successful, comprises of merely 10,000 gates. The attack simulations crashed for the larger benchmarks, which is attributed to NuSMV's inability to handle large model-checking instances. These results prove that the application of SAT attack in the absence of proper scan access is limited to only small sequential circuits, which are not under consideration in practical scenarios. As the size of modern ICs is orders of magnitude greater than the ISCAS and ITC benchmarks under consideration, we can reiterate the claim of Guin *et al*'s paper [12] that disrupting the scan operation makes it extremely difficult, if not impossible, to mount SAT attack using the available state-of-the-art resources.

8.4 Security Analysis Against ScanSAT Attack

Recently, a modified SAT attack named ScanSAT has been proposed in [32], which can break static obfuscation in scan chains. Static obfuscations mainly insert either XOR or MUX key-gates between the scan cells of the scan chain [31]. The scan-in and scan-out vectors get modified by the key-inputs during the scan shift operation, hiding the actual scan-data from the adversary. However, in the scan shift process, information regarding the key-inputs gets embedded into the modified scan-in and scan-out vectors, which are exploited by the ScanSAT attack. ScanSAT models the static scan obfuscation as a logic-locked combinational circuit, in which the key logic is inserted at the pseudo-primary inputs/outputs of the circuit. A subsequent SAT attack on the modeled circuit reveals the secret key, eventually deobfuscating the scan obfuscation.

However, the following differences of the proposed scan protection strategy from the static obfuscation make it robust against the ScanSAT attack.

- Unlike static obfuscations, the proposed scheme does not embed any key into the scan data in the process of scan shifting. Either the value of the combinational part or the previous scan cell gets shifted to an encrypted scan cell based on the correctness of the key value.
- Even for a wrong key, the source of the encrypted scan cells gets dynamically modified in every clock cycle based on the content of a random scan cell in a completely unpredictable manner.
- A wrong *Test Key* completely corrupts the functional response, leaving hardly any meaningful information for further processing by the ScanSAT attack.

ScanSAT claims that it may also break the dynamic scan obfuscation [33] if the key update cycle is larger than the number of DIPs required to extract the keys. ScanSAT attributes this extremely low number of DIPs for decrypting the locked IC to the unique impact of the incorrect key on the oracle. Note that the dynamic scan obfuscation proposed in [33] also embeds the key information into the scan data similar to the static scan obfuscations. The only difference is that the key gets updated after certain clock cycles. However, as described in the proposed defense, the scan-out and the scan-in responses are not directly affected by the incorrect-key. Instead, they have an additional dependence on the contents of the scan chain as well as the output of the combinational logic. Hence, the keys are not easily distinguishable, reducing the efficacy of SAT attack. Moreover, the dynamic modification of the source of the encrypted scan cells in every clock cycle makes the job of the attacker further difficult.

8.5 Probable Reset-and-SAT Attack and Countermeasure

Although the proposed scheme allows selecting most of the *Encrypted* flip-flops randomly, the last *Encrypted* SFF of the scan chain should be chosen carefully. Otherwise, the DfT infrastructure might leak some of the keys to the attacker. In this section, we discuss a possible vulnerability of the proposed DfT infrastructure and the necessary preventive measures, which must be taken while selecting the location of the last *Encrypted* flip-flop.

8.5.1 *Attack Scenario*

Let us consider the following scenario. Resetting the scan chain sets all the flip-flops to their initial state values. From the reverse engineered netlist, it is possible to extract the initial state value of each flip-flop by examining whether the circuit RESET line is connected to the CLEAR or the PRESET pin of the flip-flop. Although the attacker cannot upload a desired input vector into the scan chain, he/she can always

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

reset the scan chain to fix the internal flip-flops to their initial state values. Subsequently, the attacker can apply an input pattern to the primary inputs. The corresponding response gets captured in the scan cells. This action can be performed multiple times for different input patterns by resetting the scan chain each time. Any possible way of extracting the captured response of each input pattern would help the attacker launching the SAT attack on the circuit.

Let us consider the example scan chain of Figure 4. Irrespective of the correctness of the *Test Key*, the responses captured in all the scan cells, starting from the last *Encrypted* SFF (i.e. $DFF10$) to the last scan cell (i.e. $DFF12$), can be extracted correctly. Therefore, a repeated reset and subsequent SAT attack can retrieve all the key-bits present in the input logic cone of the flip-flops $DFF10$, $DFF11$, and $DFF12$. However, the key-bits of all other key-gates, located in the input logic cones of the flip-flops between $DFF1$ to $DFF9$, cannot be retrieved using SAT attack until the correct value of $FK5/TK5$ is known. $FK5$ can be extracted, if the key-gate $KG5$ is present in the input logic cone of any of the flip-flops $DFF10$, $DFF11$, or $DFF12$. Once $FK5/TK5$ is known to the attacker, he/she will be able to scan out the data captured in the flip-flops ranging from $DFF9$-$DFF12$. This will also expose the key-gates present in the input logic cone of $DFF9$ to the SAT attack. This way, the attacker can proceed backwards to unlock the scan chain further and retrieve the other keys.

8.5.2 *Countermeasure*

The attacker would get success in his/her approach only if the keys can be retrieved in a proper sequence, starting from the scan out port. For example, the attacker must first extract the correct value of $FK5/TK5$, then $FK4/TK4$, and so on. Until the value of $FK5/TK5$ is retrieved, the content of its previous scan cells cannot be scanned out. This will stall the propagation of the attack in the backward direction. Therefore, to prevent any leakage of the keys via this iterative Reset-and-SAT attack, it is imperative to protect the key of the last *Encrypted* flip-flop. Since the last *Encrypted* flip-flop acts as the final lock of the scan chain, the best way to ensure zero leakage of the keys is encrypting the last scan cell of the chain and restricting the insertion of key-gates in its input logic cone. To ensure that no key-gates are placed in the input logic cone of the last scan cell, the designer has to remove it from the G_{strong} region at the time of selecting potential key-gate locations (Section 5.1). The input logic cone of the last scan cell might also include other flip-flops. A reset operation sets those flip-flops to their fixed initial values, which can be treated as pseudo-primary inputs. Thus, the input logic cone of the last scan cell does not cover the input logic cones of those flip-flops.

8.6 Sustainability Against Removal Attack

As the proposed obfuscated scan architecture is the main impediment to SAT and scan-based circuit partitioning attacks, any rogue entity in the foundry who has access to the encrypted netlist, can bypass the scan obfuscation by directly connecting the SE input to the scan flip-flops. Performing SAT attack requires two copies of the chip: a functional chip and a locked netlist (called attacker's copy). SAT attack uses the scan chains of the functional chip to extract the vital information regarding the obfuscation key. *It should be noted that only the oracle from an activated IC contains information about the key.* To successfully launch the SAT attack on the proposed scheme, the attacker requires an activated chip whose scan obfuscation has been bypassed by means of design tampering.

However, any fabricated chip with tampered scan architecture, must first be sent to the design house for activation with the correct keys. Design tampering can be easily detected at the design house while performing in-house testing of the chip. This can be achieved by applying a wrong *Test Key* and checking scan-out vector. If the scan-out response remains the same as the scan-in vector upon application of the wrong *Test Key*, the designer can infer the circumvention of the scan obfuscation. Similarly, an attempt of removing the scan controller in the foundry can be detected by performing a scan operation immediately after a global reset on a fabricated chip.

Therefore, any tampered IC without being activated reveals no clue regarding the key. The attacker can only use his/her tampered copy to perform a normal scan operation considering any random key. For verifying the correctness of the key, the attacker requires the oracle from an activated chip. Unless the scan obfuscation of an activated chip can be bypassed, any design modification in the attacker's copy does not aid in extracting information about the correct key-inputs. Bypassing the scan defense in an activated IC requires pico-probing the fabricated circuit which is considered in-feasible for large scale circuits [40].

9 Experimental Results

In this section, we present the results of our experiments on several ISCAS'89 and ITC'99 [41] benchmarks. The key-gate placement algorithm (Algorithm 1) has been implemented in C. It identifies the best locations in the circuit to insert the key-gates. For each of the benchmarks, the algorithm finds the largest overlapping input cone of dependency (Step I). Next, we apply 10000 random patterns to the netlist and analyze the fault impact of each gate of G_{strong} using the fault simulator HOPE [37]. We select 128 locations with the higher values of fault impact metric and insert key-gates in those locations (Step II). All simulations have been carried out on a computer with 3.20 GHz Intel(R) Core(TM) i5-3470 processor and 4 GB RAM

Table 2 presents the results of the algorithm for each of the selected benchmarks. Columns 2, 3, and 4 report the number of inputs, outputs, and the logic elements present in the circuit. Columns 5 and 6 report the number of logic elements (G_{strong}) present in the largest overlapping input cone of dependency ($ICOD_{overlap}$) and the number of outputs ($O_{overlap}$) getting affected by this $ICOD_{overlap}$, respectively. For example, out of 20679 logic elements of the benchmark $s38584$, 18272 belong to the $ICOD_{overlap}$. 268 outputs (out of the total 304 outputs) are reachable from each of the 18272 logic elements. These 18272 logic elements do not propagate to any other output apart from these 268 outputs. The column Gate coverage presents the % of the total logic elements covered by the $ICOD_{overlap}$ and Output coverage shows the % of total outputs, getting

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

Table 2: Analysis of the proposed encryption strategy for $ISCAS'89$ and $ITC'99$ benchmarks. (K = 128)

Circuit Name	# Inputs	# Outputs	# Gates	G_{strong}	$O_{overlap}$	Gate Coverage (%)	Output Coverage (%)	Execution Time			
								Step I	Step II	Total	[36]
s13207	62	152	8589	5116	72	59.56	47.3	58s	1m4s	2m2s	2h43m
s15850	77	150	10306	8284	27	80.38	18	1m10s	1m19s	2m29s	3h50m
s38584	38	304	20679	18272	268	88.36	88.15	42m4s	13m21s	55m25s	125h12m
s38417	28	106	23815	22644	33	95.08	31.13	7m54s	3m8s	11m02s	21h21m
b14	32	54	10012	9003	52	89.92	96.29	2m8s	2m33s	4m41s	7h30m
b15	36	70	8816	8097	70	91.84	100	2m11s	8m6s	10m17s	6h56m
b17	37	97	32192	19451	30	60.42	30.92	44m7s	49m45s	93m52s	138h54m
b18	37	23	114561	109482	20	95.56	86.95	113m46s	194m26s	5h8m	-
b19	24	30	231266	227443	27	98.35	90	695m9s	669m25s	22h64m	-
b20	32	22	20172	18465	22	91.53	100	5m44s	15m25s	21m9s	17h21m
b21	32	22	20517	18769	22	91.48	100	5m53s	15m37s	21m30s	17h40m
b22	32	22	29897	27021	22	90.38	100	11m38s	16m13s	27m51s	36h29m

affected by any key-gate. In majority of the circuits, more than 80% logic elements belong to the $ICOD_{overlap}$. Only two circuits ($s13207$ and $b17$) have 60% of the total logic elements covered by the $ICOD_{overlap}$. These results show that the algorithm finds sufficient space to choose the best locations for the key-gates. The high output coverage (except a few circuits like $s15850$, $s38417$, and $b17$) shows that wrong keys can corrupt a high % of the output bits.

Finally, Columns 9 and 10 report the execution times of each of the two steps of the algorithm. From the results of the total execution time, one can observe the advantage offered by the linear time complexity of the proposed algorithm. It may be noted that our algorithm is capable of encrypting small circuits (≈10K gates) in 10-16 minutes, while medium sized circuits (≈ 20K/30K gates) can be encrypted within 2/3 hours. Encryption of the largest circuit $b19$ (231K gates) takes up to a maximum of 22 hours.

We also present a comparison between the execution times of our proposed algorithm and the algorithm presented in [36]. Column 12 of Table 2 reports the total execution time of the optimal key-gate location selection strategy presented in [36]. While both the algorithms offer the same degree of benefits, the algorithm of [36] takes enormous time, which can be as high as 136 times more, compared to the proposed method for a circuit consisting of 20K gates ($s38584$). The algorithm of [36] could not be run on larger circuits like $b18$ and $b19$ due to the scalability issue. In addition to the steps mentioned in this paper, the key-gate selection algorithm of [36] uses an extra step, which calculates the dependency of each candidate key-gate locations. This extremely time-consuming dependency calculation method is the main reason behind its high execution time, limiting its application to small circuits only. However, after further investigation, we found that this time-consuming step is redundant since all the key-gates present in the $ICOD$ of any output are by default dependent on each other (refer to Section 8.2). We do not need to calculate the dependency of each candidate gate separately. This drastically reduces the execution time of the algorithm without compromising on the quality of the solution.

Since the first two steps of the key-gate location selection algorithm of [36] are the same as the Algorithm 1 of this work, the G_{strong} region is the same in both of the works. Therefore, [36] also produces the same % of gate coverage and output coverage values, as reported in Table 2.

Output Corruptibility For Wrong Keys: To measure the output corruptibility of the proposed key-gate selection scheme, we have simulated each benchmark with 100000 random input patterns. Random incorrect keys were selected and Hamming Distance was measured between the correct and the obtained outputs. Figure 9 shows a comparison of average output corruption between different logic encryption strategies for K = 128. For this comparison, we have mainly considered strong logic locking (SLL) [5], fault analysis (FA) based (both XOR and MUX-based) [14], and optimal key-gate location selection [36] strategies. It may be noted that, although we have restricted the insertion of key-gates only in the G_{strong} region, the output corruption results of our technique is comparable with the original fault analysis based approach [14]. However, unlike our method, this strategy is unable to prevent path sensitization or logic cone based attacks. On the other hand, SLL [5], which prevents path sensitization attack, produces poor output corruption compared to our method. The proposed key-gate location selection strategy offers the same level of output corruption as reported in [36], but takes much less time as indicated in Table 2.

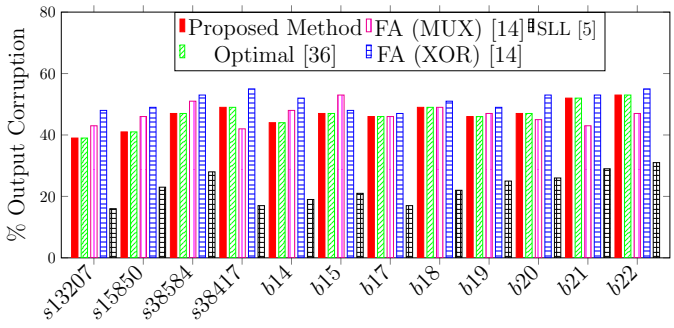


Figure 9: Comparison of average output corruption between different logic encryption strategies for K = 128

9.1 Area, Power, and Delay Overheads

To evaluate the area, power, and delay overheads of our proposed encryption technique, we have synthesized each of the designs (encrypted and original netlist) in Synopsys Design Vision tool [42] using Faraday 90nm library. Scan chain is inserted in the circuit and $Scan\ Enable$ (SE) line of 128 flip-flops is encrypted using the methodology described in Section 6.1.

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

Overheads of other encryption techniques like traditional XOR/XNOR based [2], [5], [14], MUX-based [14] and Obfuscation Cell (OC) based [43] encryption strategies are compared with our method. Area overhead of SARLock [15] and Anti-SAT [16] methods, which are capable of preventing the powerful SAT attack, is also considered. Due to the high implementation complexity, overheads of SARLock and Anti-SAT was estimated by adding the area of the extra hardware incurred by them. Calculation of power overhead requires implementation, therefore, power overhead of our method is not compared with these two methods. Figure 10 and 11 compare the area and power overheads of different logic encryption strategies. Our proposed encryption strategy doesn't insert any key-gate in the critical path resulting in zero delay overhead. Therefore, delay overhead comparison is not considered.

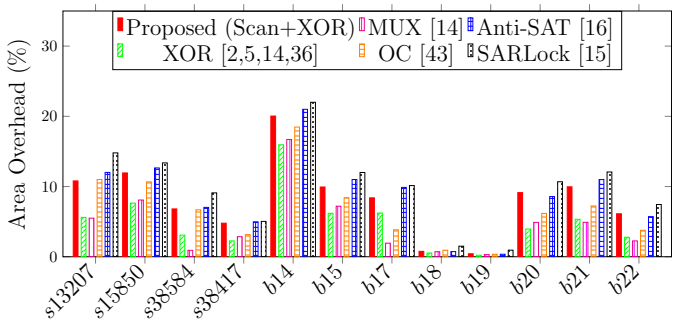


Figure 10: Comparison of area overheads between different logic encryption techniques (K = 128)

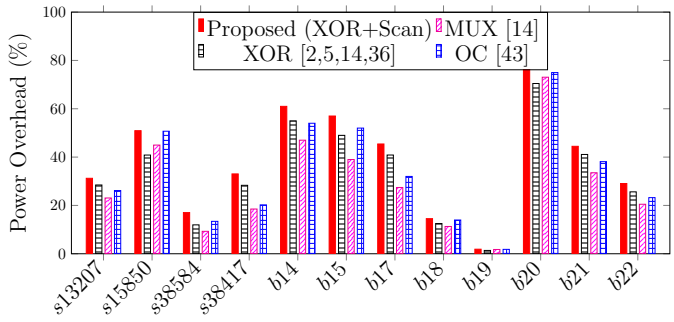


Figure 11: Comparison of power overheads between different logic encryption techniques (K = 128)

One can observe from Figures 10 and 11 that the area and power overheads of our proposed method is high compared to traditional XOR/XNOR-based ([2], [5], [14]) and MUX-based [14] encryption. This is due to the integration of scan encryption. Unlike the proposed defense, other strategies fail to prevent SAT or scan-based circuit partitioning attacks. Our area overhead is comparable to the SAT-preventive methods SARLock [15] and Anti-SAT [16].

9.2 A Case Study on RISC-V CPU Core

As a case study on a real circuit, we have applied the proposed logic encryption technique on a RISC-V CPU core. Table 3 shows the results of our experimentation on PicoRV32 CPU core, which implements the RISC-V 32 bit instruction set (IMC extensions) [44]. The first step of Algorithm 1 selects 25031 logic elements (out of 26209) and 265 outputs (out of 335) as the members of the largest overlapping input cone of dependency. It may be noted that in the real circuit as well, a high percentage of total logic elements and outputs belong to the largest overlapping input cone of dependency, depicting the feasibility of the proposed key-gate selection scheme. From these 25031 logic elements of the G_{strong} region, the second step finds 128 suitable locations based on the fault impact metric and inserts the key-gates in those locations. Finally, we have encrypted the scan operation using the strategy described in Section 6.1. To evaluate SAT-resistance, we have mounted the Bounded Model Checking attack on the circuit, assuming no scan access. The attack simulation has crashed for the circuit, proving the failure of SAT-attack on the proposed scheme.

Table 3: A case study on RISC-V CPU core PicoRV32

Circuit Name	# Inputs	# Outputs	# Gates	G_{strong}	$O_{overlap}$	BMC Attack
PicoRV32	97	335	26209	25031	265	CR

10 Comparison With The State-of-the-art Logic Encryption Schemes

In this section, we present a comparison of our technique with different state-of-the-art logic encryption schemes.

10.1 Point Function Based SAT-resilience vs. Scan Protection Based SAT-resilience

As mentioned in Section 3, point function based strategies [15]–[19] increase SAT attack complexity by exponentially increasing the number of DIPs. Consequently, they fail to meet the fundamental requirement of output corruptibility of a logic encryption scheme. Most of these strategies are susceptible to several other attacks, summarized in Table 4. Although SFLL-flex [18] and SFLL-fault [19] are yet to be broken by any of the existing attacks, both of them offer protection only to a limited number of input patterns/cubes.

High output corruption and point function based SAT-resilience are two contradictory attributes which cannot be satisfied simultaneously. Point function based SAT-resilient techniques cannot be integrated with either FA or SLL based strategy in an attempt to improve output corruptibility. A multi-layered defense of a point function and a SAT-vulnerable scheme can be reduced to a single layer defense of point function solely [18].

On the other hand, one can make the valid assumption that all practical circuits are reasonably large in size and sequential in nature. Restricting the scan access to the unauthorized users eliminates the possibility of SAT attack without compromising on output corruptibility.

Not only that, despite improving the runtime compared to SFLL-flex, SFLL-fault is much slower compared to our proposed scheme. For example, as per the reported data in [19], SFLL-fault takes in the range of 200+ to 300+ minutes to obfuscate the benchmark circuits $b20$, $b21$, and $b22$. On the other hand, our proposed scheme takes in the range of 20-30 minutes to obfuscate these circuits, which is ten times faster

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

Table 4: Comparison with different point function based SAT-resilient logic encryption schemes [×: vulnerable]

Different SAT resilient strategies	Different Attacks			Nature of Output Corruption
	Removal [17]	Bypass [20]	Other Attacks	
SARLock [15]	×	×	DoubleDIP [23], AppSAT [22]	One output bit for only one input pattern
Anti-SAT [16]	×	×	SPS [21]	poor
TT-Lock [17]	✓	×	DoubleDIP [23], AppSAT [22]	One output bit for only one input pattern
SFLL-HD [18]	✓	✓	Fall Attack [25], Functional reverse engineering [24]	One output bit for few input patterns
SFLL-Flex [18]	✓	✓	-	Multiple output bits for few input patterns
SFLL-fault [19]	✓	✓	-	Multiple output bits for few input patterns
Proposed Scheme	✓	✓	-	High output corruption for most of the input patterns (Avg 45%)

Table 5: Comparison of the proposed scheme with different scan obfuscation guided SAT resilient techniques

SAT-resilient strategies	Different Attacks			Testability Support	Output Corruption
	Shift & Leak [30]	ScanSAT [32]	Removal [17]		
Secure Cell + SLL [12]	×	✓	✓	✓	Low
Encrypt Flip-Flop [31]	✓	×	✓	✓	varies from circuit to circuit
Dynamic Scan Obfuscation + SLL [33]	✓	Probably vulnerable	✓	✓	Low
Proposed Scheme	✓	✓	✓	✓	High

than SFLL-fault. A similar kind of results can be observed for other circuits as well.

The proposed scan obfuscation based strategy offers the following advantages over the point function guided SAT resilient schemes.

- High output corruption for an incorrect key.
- Flexibility in key-gate placement, maximizing the effort of path sensitization and logic cone based attacks.
- Faster encryption.

The support for both manufacturing and functional tests enables the proposed scheme to ensure production of reliable and defect-free chips.

10.2 Comparison with different scan obfuscation guided SAT-resilient techniques

Table 5 presents a comparative analysis between the proposed method and other scan obfuscation guided SAT-resilient strategies. All of these methods rely on blocking the scan observability, although they vary in their implementation techniques. Both *Encrypt Flip-Flop* [31], and dynamic scan obfuscation [33] obfuscate the scan data using a secret key at the time of scan shifting. In the process, the key information gets embedded into the obfuscated response, which is exploited by the ScanSAT attack [32] to expose the keys eventually. On the other hand, the proposed scheme relies on a test authentication step to grant proper scan access. An incorrect *test key* results in a corrupted scan-out response, which cannot be processed further to recover the original functional response. The corrupted scan-out data, with no key information embedded into it, does not allow ScanSAT attack to retrieve the key. All of these reported scan obfuscation schemes, except Secure Cell-based technique [12], also offer restricted controllability of the scan data. Restricting the scan controllability is also essential, since it can be observed that the *Shift & Leak* attack [30] exploits this feature to attack the Secure Cell strategy.

As can be seen from the table, all of these methods offer full testability support, which is extremely important. Although the Secure Cell-based scheme has not reported any result on output corruptibility, it relies on SLL-based strategy to insert the key-gates. Since SLL does not offer high output corruptibility, Secure Cell-based technique should also exhibit the same behavior. Similarly, dynamic scan obfuscation, integrated with SLL, produces low output corruption. On the other hand, explicit measure at the time of key-gate placement ensures that the proposed method offers high output corruption for an applied wrong key.

10.3 Comparison with different key-gate placement schemes

A comparison of our technique with other key-gate placement strategies has been presented in Table 6. Most of the existing schemes cannot prevent both path sensitization and logic cone based attacks and simultaneously offer high output corruptibility. On the contrary, our proposed scheme can satisfy all the criteria of an efficient key-gate placement strategy. Furthermore, unlike our proposed technique, none of these methods can prevent SAT and scan-based circuit partitioning attacks. We have also compared the % of output corruptibility between different key-gate placement strategies. For this purpose, we have experimented on all the twelve ISCAS 89 and ITC 99 benchmark circuits reported in Table 2. We have encrypted each of these circuits using different key-gate placement strategies reported in Table 6. We have simulated each encrypted circuit using 100000 random input patterns considering random wrong keys. For all the different techniques, we have taken the average of the % output corruption for all the experimented benchmark circuits and reported them in Table 6. It can be observed that random logic locking [2], SLL [5], and SLL + logic cone

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

prevention [10] produces poor output corruptibility, while the other methods, including the proposed scheme, produce high output corruption for wrong keys.

Table 6: Comparison of proposed scheme with different key-gate placement schemes [×: vulnerable]

Different key-gate placement schemes	Different Attacks			% Output Corruptibility
	Path [5] Sensitization	Logic Cone [10]	SAT [11]	
Random [2]	×	×	×	32.64
Fault Analysis (FA) [14]	×	×	×	51.08
Strong Logic Locking [5]	✓	×	×	22.83
SLL + Logic Cone Prevention [10]	✓	✓	×	25.34
External Key-Dependency [45]	✓	✓	×	47.3
Optimal Key-gate Location [36]	✓	✓	×	46.66
Proposed Scheme	✓	✓	✓	46.66

11 Conclusion

In this paper, we have proposed a new logic encryption strategy which encrypts a design in two phases; the first phase focuses on finding out the best locations of the key-gates ensuring defense against path sensitization and logic cone based attacks, while the second phase restricts any unauthorized scan access. Security analysis on the proposed secure DfT infrastructure demonstrates its ability to thwart SAT and scan-based circuit partitioning attacks on any large circuit used in practical scenarios. Unlike most of the SAT-preventive techniques, our method does not suffer from low output corruption for wrong keys. A single security framework of preventing all the state-of-the-art attacks, including SAT and offering high output corruption makes the proposed strategy different from the other cutting-edge logic encryption techniques. With the advantage of full testability support, the proposed scalable and straightforward scheme can easily be integrated with any design flow to enhance the security of the design.

References

[1] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[2] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *DATE*, 2008, pp. 1069–1074.

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.

[4] "Innovation is at risk: Losses of up to $4 billion annually due to ip infringement," *SEMI*. [Online]. Available: http://semi.org/en/innovation-risk-losses-4-billion-annually-due-ip-infringement

[5] M. Yasin, J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, no. 99, pp. 1–1, 2015.

[6] L.-W. Chow, J. P. Baukus, and W. M. Clark Jr, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," Nov. 13 2007, uS Patent 7,294,935.

[7] R. W. Jarvis and M. G. Mcintyre, "Split manufacturing method for advanced semiconductor circuits," Mar. 27 2007, uS Patent 7,195,931.

[8] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking techniques for intellectual property protection," in *DAC*. ACM, 1998, pp. 776–781.

[9] M. Yasin, J. J. Rajendran, and O. Sinanoglu, "Trustworthy hardware design: Combinational logic locking techniques."

[10] Y.-W. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *LATS*. IEEE, 2015, pp. 1–6.

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

[12] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in ic manufacturing and test," *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 2018.

[13] R. Karmakar, S. Chattopadhyay, and M. Chakraborty, "Improving security of logic encryption in presence of design-for-testability infrastructure," in *ISCAS*. IEEE, 2019, pp. 1–5.

[14] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Tran. on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[15] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *HOST*. IEEE, 2016, pp. 236–241.

[16] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," *IACR Cryptology ePrint Archive*, vol. 2016(590), 2016.

[17] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to lock?: Functional and parametric locking," in *GLSVLSI*. ACM, 2017, pp. 351–356.

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

[19] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "Atpg-based cost-effective, secure logic locking," in *VTS*. IEEE, 2018, pp. 1–6.

[20] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," *IACR Cryptology ePrint Archive*, 2017.

[21] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-sat," *IACR Cryptology ePrint Archive*, vol. 896, 2016.

[22] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately deobfuscating integrated circuits," in *HOST*. IEEE, 2017, pp. 95–100.

[23] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *GLSVLSI*. ACM, 2017, pp. 179–184.

[24] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, and M. Al-Qutayri, "Functional reverse engineering on sat-attack resilient logic locking," in *ISCAS*. IEEE, 2019, pp. 1–5.

[25] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," *arXiv preprint arXiv:1811.12088*, 2018.

[26] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *GLSVLSI*. ACM, 2017, pp. 173–178.

[27] H. Zhou, R. Jiang, and S. Kong, "Cycsat: Sat-based attack on cyclic logic encryptions," *IACR Cryptology ePrint Archive*, vol. 626, 2017.

[28] Y.-C. Chen, "Enhancements to sat attack: Speedup and breaking cyclic logic encryption," *ACM Trans. on Design Automation of Electronic Systems*, vol. 23, no. 4, p. 52, 2018.

[29] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *DATE*. IEEE, 2018, pp. 85–90.

[30] N. Limaye, A. Sengupta, M. Nabeel, and O. Sinanoglu, "Is robust design-for-security robust enough? attack on locked circuits with restricted scan chain access," *arXiv preprint arXiv:1906.07806*, 2019.

[31] R. Karmakar, S. Chatopadhyay, and R. Kapur, "Encrypt flip-flop: A novel logic encryption technique for sequential circuits," *arXiv preprint arXiv:1801.04961*, 2018.

[32] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "Scansat: unlocking obfuscated scan chains," in *ASPDAC*. ACM, 2019, pp. 352–357.

[33] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure scan and test using obfuscation throughout supply chain," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, 2018.

[34] M. El Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," in *ICCAD*. IEEE, 2017, pp. 33–40.

[35] Y. Kasarabada, S. Chen, and R. Vemuri, "On sat-based attacks on encrypted sequential logic circuits," in *ISQED*. IEEE, 2019, pp. 204–211.

[36] R. Karmakar, H. Kumar, and S. Chattopadhyay, "On finding suitable key-gate locations in logic encryption," in *IEEE ISCAS*, 2018, pp. 1–5.

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2963094, IEEE Transactions on Emerging Topics in Computing

[37] H. K. Lee and D. S. Ha, "Hope: An efficient parallel fault simulator for synchronous sequential circuits," *IEEE Tran. on CAD of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, 1996.
[38] U. Guin, Z. Zhou, and A. Singh, "A novel design-for-security (dfs) architecture to prevent unauthorized ic overproduction," in *VTS*. IEEE, 2017, pp. 1–6.
[39] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, Mar 2000. [Online]. Available: https://doi.org/10.1007/s100090050046
[40] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *CHES*. Springer, 2009, pp. 363–381.
[41] C. Albrecht, "Iwls 2005 benchmarks," in *IWLS*, 2005.
[42] *Design Vision "User Guide", Version 2002.05*, Synopsys Inc.
[43] J. Zhang, "A practical logic obfuscation technique for hardware security," *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 1193–1197, 2016.
[44] [Online]. Available: https://github.com/cliffordwolf/picorv32
[45] R. Karmakar, N. Prasad, S. Chattopadhyay, R. Kapur, and I. Sengupta, "A new logic encryption strategy ensuring key interdependency," in *VLSID*. IEEE, 2017, pp. 429–434.

Rajit Karmakar received his MS degree in Microelectronics and VLSI from Indian Institute of Technology, Kharagpur, India, in 2015. He is presently a PhD student in the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, India. His current research interests include hardware security and VLSI Testing. He is a student member of IEEE.

Harshit Kumar is a dual degree student in the Department of Electronics and Electrical Communication Engineering at IIT Kharagpur. He is currently pursuing a specialization in VLSI & Microelectronics. His research interest includes hardware security and brain-machine interfaces.

Santanu Chattopadhyay is a Professor with the Department of Electronics and Electrical Communication Engineering at IIT Kharagpur. He has published more than 150 technical papers in peer-reviewed journals and conferences. His current research interests include hardware security, digital circuit design, testing and diagnosis, network-on-chip design and test, and low-power circuit design and test. He is a senior member of IEEE.

2168-6750 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on December 18,2020 at 08:40:28 UTC from IEEE Xplore. Restrictions apply.