TaintLock: Hardware IP Protection against Oracle-guided and Oracle-reconstruction Attacks*

Jonti Talukdar†, Arjun Chaudhuri‡**, Eduardo Ortega‡**, and Krishnendu Chakrabarty‡**
†Department of Electrical and Computer Engineering, Duke University, Durham, NC
‡School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ
**ASU Center for Semiconductor Microelectronics (ACME), Tempe, AZ

***Abstract*—Scan-obfuscation schemes used with logic locking lack the ability to perform scan authentication on a per-pattern basis. These methods are of limited effectiveness in obfuscating scan data and they remain vulnerable to SAT-based scan deobfuscation attacks. In addition, prior methods designed to perform scan-data authentication are not adequate under the strongest threat models used to assess logic locking. To alleviate these problems, we propose enhancements to TaintLock, a lightweight dynamic per-pattern authentication and encryption scheme that uses taint and signature bits embedded within each test pattern to provide authenticated scan access. To prevent IP theft through Oracle-free and Oracle-guided attacks, TaintLock is paired with truly random logic locking (TRLL). TaintLock cryptographically authenticates each test pattern using the embedded taint and signature bits and passing them through a substitution-permutation (SP) network. It further uses cryptographically generated keys to dynamically encrypt scan data for unauthenticated users. TaintLock, while offering a low overhead and non-intrusive secure scan solution may remain susceptible to a new class of Oracle-reconstruction attacks that use machine learning. Additionally, assuming test pattern security is compromised, it may be potentially vulnerable to a template-based SAT attack aimed at partial key recovery. We analyze the susceptibility of TaintLock against these threats and demonstrate its resilience. We also demonstrate that TaintLock can be easily integrated with popular test architectures such as embedded deterministic test (EDT). Finally, we also discuss the reconfigurable nature of TaintLock's architecture to support different levels of encryption and authentication.**

I. Introduction

The production of integrated circuits (IC) is increasingly reliant on a globalized supply chain leading to significant risk in the security of on-chip intellectual property (IP). Of particular concern is the vulnerability to IP theft through attacks such as IC counterfeiting, IP piracy through overbuilding, and netlist-reverse engineering [2], [3]. Security solutions that obfuscate IP through logic locking offer protection against these attacks by locking the IP using combinational key-gates and/or sequential obfuscation states [4]. However, powerful SAT-based Oracle-guided attacks are capable of rapidly pruning the key-search space to extract the correct key [4].

Recent work has shown that combining scan chain security with IP obfuscation methods such as truly random logic locking (TRLL) can not only thwart Oracle-guided attacks but also achieve resilience against Oracle-free attacks [5]. Existing scan protection schemes either obfuscate scan data or block scan access altogether [6]. The strongest scan-obfuscation schemes use keys generated from an LFSR to achieve dynamic obfuscation, i.e., dynamically obfuscating scan data for each pattern. However, scan authentication is not supported by such methods, thereby making it difficult to determine the trustworthiness of the end-user accessing the scan chains. Moreover, the use of linear obfuscation (using LFSRs) makes these methods vulnerable to attacks that remodel the dynamically obfuscated scan chains as a combinational logic-locked netlist [7], [8].

Combining dynamic scan obfuscation with per-pattern authentication offers the ability to independently validate the authenticity of each scanned pattern while preventing unauthorized users from using the scan chains through scan obfuscation. This requires embedding the information used for authentication within every test pattern. Prior work has demonstrated the vulnerabilities associated with scan authentication methods under the latest (strongest) threat models used in context of logic locking [1], [9], [10]. In this paper, we present **TaintLock**, a dynamic scan data authentication and encryption scheme that performs dynamic per-pattern authentication while achieving secure dynamic obfuscation (through encryption) for unauthorized users using *taint* bits embedded in the test pattern. A preliminary version of TaintLock was presented in [1].

It has been shown that TaintLock is an effective low-cost/overhead architecture that supports cryptographically secure scan access through both test-pattern based authentication and encryption. A unique authentication and encryption key, that changes dynamically, is computed using the taint-bits embedded within each test pattern itself. The same pattern can have a different key embedding, based on the order in which it is scanned. TaintLock employs a challenge-response authentication mechanism using dynamically changing keys that are generated cryptographically and on the fly from taint bits embedded in each test pattern. The keys used for authentication and encryption change based on the pattern and on the order in which it is scanned in. Application of an incorrect key for a given test pattern triggers scan data encryption. As TaintLock is integrated with a low-overhead high output-corruptibility logic locking scheme (TRLL), it is resilient against not only Oracle-guided attacks but also against Oracle-free attacks, including removal/bypass attacks. In addition, its lightweight non-linear scan encryption scheme offers resilience against all existing attacks mounted on obfuscated scan chains, while incurring less than 0.2% area overhead for large circuits.

*This research was supported in part by the Semiconductor Research Corporation (Task ID 2994.001) and the National Science Foundation under grant no. CNS-2011561. A preliminary version of this paper was presented at European Test Symposium, 2022 [1].

However, new classes of attacks have come to light since TaintLock's conception. In this work, we demonstrate the applicability of a new class of Oracle-reconstruction attacks that utilize machine learning (ML) techniques to reconstruct locked IP protected by TaintLock [11]. We further demonstrate that the scan-based authentication and encryption network designed as part of the TaintLock architecture is able to limit the effectiveness of machine learning-based Oracle reconstruction attacks such as LORAX [11]. Furthermore, assuming that test pattern security is compromised, we also demonstrate the ineffectiveness of a template-based SAT attack aimed at partial key recovery. We show that both categories of attacks remain ineffective against the security guarantees provided by TaintLock. In addition, we also perform a design space exploration of the TaintLock architecture, analyzing the power, performance, area, and security overhead of the TaintLock architecture. While TaintLock offers security guarantees against a wide array of reverse engineering attacks by embedding signature and taint bits within scan patterns, we further show it provides support for test compression schemes such as embedded deterministic test (EDT).

The main contributions of this paper are as follows:

- We present a design space exploration of the TaintLock architecture, a dynamic per-pattern scan data authentication and encryption mechanism that utilizes substitution-permutation (SP) networks to authenticate every test pattern based on embedded taint and signature bits.
- We devise an integer linear programming (ILP)-based approach to optimally allocate taint and signature locations for any set of test patterns and integrate it with the parametrized TaintLock architecture to generate authenticated test patterns.
- We integrate support for pattern compaction along with EDT (embedded deterministic test) when considering test authentication through TaintLock. This allows us to enable signature- and taint-embedding while maintaining support for EDT-based test compaction.
- We evaluate the effectiveness of ML-guided Oracle reconstruction attacks that utilize machine learning techniques to reconstruct the Boolean functionality of the IP without scan access. We demonstrate that Oracle-reconstruction attacks are of limited effectiveness against TaintLock.
- We demonstrate the effectiveness of TaintLock against a template-based SAT attack that is aimed at recovering partial keys by reusing the taint and signature information present in a leaked test set.
- We show that TaintLock is adversarially indistinguishable, resilient against known- and chosen-plaintext attacks and any form of Oracle-guided attack, and secure against state-of-the-art scan de-obfuscation attacks.

The rest of the paper is organized as follows: Section II reviews the background and related prior work on secure scan methods and highlights their vulnerabilities. Section III describes the TaintLock architecture and its sub-blocks in detail. Section IV presents a security analysis of TaintLock. Section V presents testability analysis, Section VI presents its overhead analysis, and Section VII concludes the paper.

II. Background and Related Prior Work

In this section, we analyze the drawbacks of recent scan-obfuscation methods and demonstrate their vulnerabilities against the strongest threat models used in logic locking.

A. Secure Scan Chains and IP Protection

Secure scan methods have been paired with IP obfuscation to prevent leakage of critical on chip data including locking keys and functional state of the IP. Early scan-protection methods majorly focused on protecting on-chip cryptographic cores from data leakage and side-channel attacks by flipping scan bits and scrambling the data stored in scan chains. By obfuscating scan data through bit flips, such methods were able to attain data security but remained vulnerable to attacks focused on netlist reconstruction through structural reverse engineering [4].

Methods that replace certain scan-cells in the design with custom secure scan cells achieve both logic- and scan-data obfuscation. Secure scan-cells can block scan access and introduce bit flips in scan data unless the IP is activated using the functionally correct scan unlocking key [12], [13]. One such method use embedded XOR gates in the scan chain to flip scan data values dynamically through keys generated from on-board LFSRs [12]. Other methods allow scan-cells to control the polarity of the output (Q or $\overline{Q}$) depending on the value of the locking key [13]. Dynamic obfuscation can then achieved by modifying the keys dynamically through the use of onboard LFSRs. However, due to the use of linear structures (embedded XOR gates and LFSRs), attacks such as DynUnlock [7] and ScanSAT [8] can apply variations of the SAT attack to extract the obfuscation key (LFSR seed).

Scan-obfuscation methods have also been proposed to cryptographically encrypt scan data. However, these methods utilize cryptographic macros and suffer from large area overhead and test times [14]. Additionally, these methods also lack support for in-field functional debug, dynamic per-pattern authentication, while also impacting the timing profile of functional paths [6].

Although methods that support scan pattern authentication exist, they suffer from test data and test time overhead due to the requirement of additional security flops. They also lack support for multi-pattern tests such as Launch-on-Capture (LOC) / Launch-on-Shift (LOS). When applied in context of stronger threat models today, these secure scan methods remain vulnerable to reconstruction attacks aimed at extracting the keys used for pattern authentication. One such method is low-cost secure scan (LCSS) [9]. LCSS supports static scan-data authentication by embedding the authentication key in dummy flip-flops added to the original scan chain. The dummy flip-flops used to store security bits also lead to increased test time and tester memory overhead. Another method [10] achieves dynamic scan data authentication by randomizing the test keys (SSTKR) that are generated using an on-board LFSR. Similar to LCSS, the test keys are either embedded in dummy flip-flops added to the original scan chain, or within don't-

care bits of each test pattern. However, this approach does not specify any selection criteria for the don't-care bits, thereby failing to address the issue of efficiently embedding the test key across the don't-care bits for patterns in the test data set. Additionally, the authentication keys in the above methods are not generated using specified bits embedded within the test pattern itself, making it easy to reverse-engineer the key from already available test data [1].

More recent methods that protect the scan interface through the use of parallel latches and skew-based flip flops have been proposed [15], [16]. In [15], a parallel latch-based architecture is used to authenticate a user supplied key by comparing it with an on-board key stored in a disposable programmable read only memory. In [16], skew-based key flip-flops are used to synchronize the delivery of the user supplied key to achieve authentication. All of these methods rely on the authenticity of both the value of the supplied key as well as the clock cycle at which the key is supplied. Although these methods work well to gate keep scan access from unauthorized users, the overall netlist still remains unprotected from netlist-level reverse engineering, thereby preventing these methods from protection against IP theft.

B. Threat Model and Assumptions

The most pervasive threat model used in IP obfuscation assumes that both the foundry and end-user are untrusted [4]. Thus, the attacker has access to the following: (1) *Locked Design* (C_{lock}): An untrusted end-user or foundry can obtain the reverse-engineered (RE'd) netlist of the locked design; (2) *Activated Chip* (C_{or}, also known as an Oracle or unlocked chip): The adversary has access to an activated chip purchased from the open market; (3) *In-field Test Patterns* (P_{auth}): The untrusted end-user has access to authenticated test patterns provided by the design house for in-field functional debug of the *activated chip*.

While the first two assumptions are common across all logic-locking techniques [4], the third assumption is relevant when a scan-authentication mechanism allows an end-user to apply authenticated test patterns to the activated device. These patterns may be supplied by the design house for in-field debug. Fig. 1 illustrates the typical flow of an obfuscated IP through the supply chain. Based on the above threat model, the attacker first uses the RE'd netlist to identify the type of security architecture used for logic and scan obfuscation. As the attacker has no control over the patterns included in P_{auth}, these patterns are not sufficient to mount an Oracle-guided attack. Thus, the attacker is forced to analyze P_{auth} to extract the scan authentication key to gain Oracle access. We do not consider invasive electrical probing attacks because these methods do not exploit the vulnerability of the obfuscation method itself, but rely on the shortcomings of the fabrication technology [2].

As shown in [1], the patterns in P_{auth} are sufficient to reconstruct the static scan authentication key, k_{static}, in LCSS, and the set of dynamic authentication keys, $K_{dyn} = \{k_1, k_2, ..., k_n\}$, in SSTKR. As the authentication keys are not

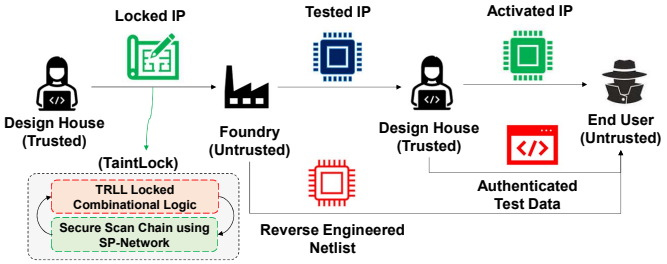

Fig. 1: A scan-protected IP through the supply chain.

generated from specified bits embedded within the test pattern, the attacker needs only a single authenticated pattern to extract the authentication keys for existing static and dynamic key-based authentication architectures. This motivates the need for a more secure authentication mechanism.

III. TaintLock: A Generalized Architecture

A. Overview

TaintLock supports both *authentication* and *encryption*. Under the assumptions made by the threat model, as both trusted and untrusted users may have access to the scan chains, authentication is performed on a per-pattern basis. Scan responses for unauthenticated test patterns are encrypted.

Authentication: TaintLock uses a challenge-response mechanism to validate the authenticity of each test pattern. For each pattern P_i, there exists a corresponding signature value, S_i, computed using taint value, T_i, embedded in it. A Feistel structure-based substitution permutation (SP) network with a dynamic key schedule is used to compute $S_i = F_{auth}(T_i, K_i)$, where K_i is a dynamically changing key generated from a free-running reconfigurable block, and F_{auth} is a lightweight block cipher implemented by the SP-network. The same P_i can have a different S_i, depending on the order in which the pattern is scanned. Since an authenticated end-user is aware of the key-schedule, they can compute S_i for any P_i.

Encryption: TaintLock utilizes a stream cipher to dynamically encrypt scan data using XOR gates placed in the scan chain. The scan encryption key, K_{encr}, is generated by re-purposing F_{auth} as a stream cipher in counter mode [17]. Thus, $K_{encr} = F(T_i, K_i) \oplus \phi_i$, where ϕ_i is a dynamically changing key, also generated from a free-running reconfigurable block. The encrypted response is $R_{encr} = R_i \oplus K_{encr}$, where R_i is the original response.

For each test pattern, taint bits are selected from the specified locations within the test pattern while signature bits are embedded within the test pattern by replacing carefully selected don't-care bits ($X's$). As the locations of X-bits used for embedding S_i may change for each pattern, an integer linear programming (ILP)-based scheme selects the minimum number of S_i locations required to cover all ATPG-patterns in the test set. The number of bits used in T_i and S_i along with their corresponding allocation is determined by the architecture of F_{auth}, as discussed later in this section. Fig. 2 illustrates dynamic authentication using TaintLock. Note that the locations of the signature-bits change for each pattern.

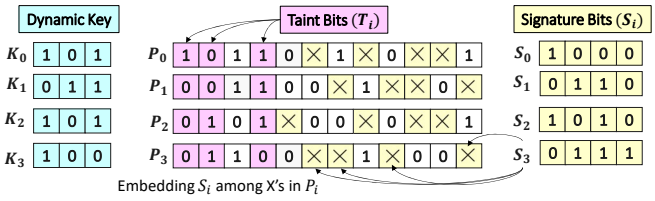

Fig. 2: Dynamic scan authentication using taint bits.

B. System Architecture

TaintLock consists of the following components (Fig. 3):

- **Taint and Signature Cells:** These scan cells are identical to regular scan cells and are responsible for storing the taint and selection bits associated with each pattern.
- **Selection MUXes:** Used to specify which signature cells contain the signature bits for every pattern. If there exist r signature cells, with each signature S_i being q-bits wide, then there will exist at most q such $r:1$ MUXes. The MUX select signals are scanned into a separate register in parallel with the test pattern.
- **Authentication Block (F_{auth}):** Computes the authentication signature using the supplied taint bits and a dynamically generated key through an SP-network.
- **Reconfigurable Block:** Consists of several LFSRs with reconfigurable feedback responsible for generating the dynamic keys used for signature computation and scan encryption. The LFSR seeds and feedback settings are supplied from tamper-proof memory and only known to authenticated users.
- **Comparator:** Performs comparison between the authentication signature embedded in the test pattern and the signature computed on-board using F_{auth}.
- **Encryptor:** In case of mismatch between the computed and embedded signatures, it generates and stores the key, K_{encr}, used to encrypt the test response.

During an authenticated test access, P_i is scanned in with the appropriate S_i embedded in it, leading to unobfuscated scan operation. As S_i can only be computed by a trusted user with access to the dynamic keys used for authentication and encryption, an untrusted user will be able to access only the encrypted responses for the supplied pattern.

C. Feistel Structure-based SP-Network Design

The authentication block consists of five layers (Fig. 4), alternating between permutation, key-whitening, and non-linear diffusion layers. The baseline authentication block supports two rounds of permutation (L1, L4) and key-whitening (L2, L5) while performing one round of non-linear diffusion (L3) using a dynamic key schedule.

Layer Architecture and Key Sizes: The sizes of L1–L5 and k_1–k_4 are determined based on the number of signature bits (q bits per pattern) and the size of MUXes used in the permutation layers (2^α:1 MUX). Given q and α, the parametric values of the layer and key sizes are shown in Fig. 4. The permutation layers (L1, L4) use a network of 2:1 MUXes ($\alpha = 1$) to selectively propagate half the incoming bits to the next layer. Thus, each

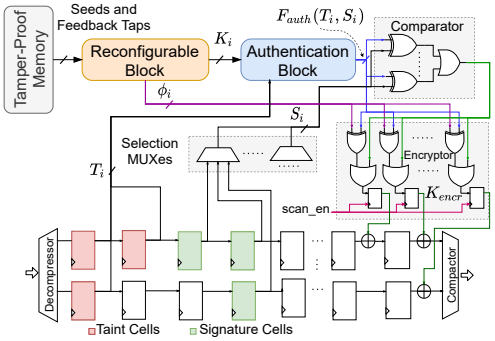

Fig. 3: TaintLock system architecture.

incoming bit has an equal probability of selection based on the dynamic key used to drive its corresponding MUX select line. The key-whitening layers (L2, L5) are used to encrypt intermediate data and increase the size of the key space. The diffusion layer (L3) uses several 6×4 combinational S-Boxes to perform non-linear mapping between input and output. The number of S-Boxes is given by $q \times 2^{\alpha-2}$. As $\alpha = 1$ and we can only have an integer number of S-Boxes, q must be an even value. Finally, for $\alpha = 1$, the number of taint bits, $t = 6q$.

Key Schedule: The keys (k_1, k_2, k_3, k_4) driving the permutation and key-whitening layers are generated from on-chip LFSRs with reconfigurable feedback (RB). The key schedule is determined by the feedback configuration and seed of each RB. An RB of size λ consists of $\lambda - 1$ MUXes in its feedback path. The size of the seed for such an RB is $2\lambda - 1$. The key update frequency is determined by the clock signal of the RB. The clock input to the RB is driven by the *scan_en* signal, thereby updating the LFSR state, and consequently the keys, dynamically for each pattern. Fig. 5 shows the illustration of the timing diagram of Taintlock's internal signals. When an authenticated pattern, P_a is scanned in, the comparator's output is 0, indicating that the generated signature matched the embedded signature, leading to an authenticated response. When an unauthenticated pattern, P_b is scanned in, comparator's output becomes 1, indicating a signature mismatch leading to an encrypted output response. As the initialization seed of each RB can be changed for each IC/device, Taintlock supports a unique key per device.

Permutation Layer Design: This layer rearranges l incoming bits into m new locations using m different $n:1$ MUXes ($m < l$). To maintain uniform permutation of incoming bits, we must ensure that each incoming bit has an equal probability of occurring at the MUX output. Thus, all the $m \times n$ MUX inputs must be evenly distributed across the l incoming bits leading to each incoming bit being connected to nm/l MUXes on average. Note that $nm \geq l$. For e.g., consider $l = 8$, $m = 5$, and $n = 4$. Using our strategy, it follows that each incoming bit is connected to at least $\lfloor \frac{20}{8} \rfloor = 2$ and at most $\lceil \frac{20}{8} \rceil = 3$ unique MUXes. This ensures that every incoming bit can occur at a given output location with an average probability of $20/8^2 = 0.31$.

Scan Authentication & Encryption: Let k_1, k_2, k_3, and k_4 represent the dynamic keys used in different layers, $p_i(k_i, t)$ rep-

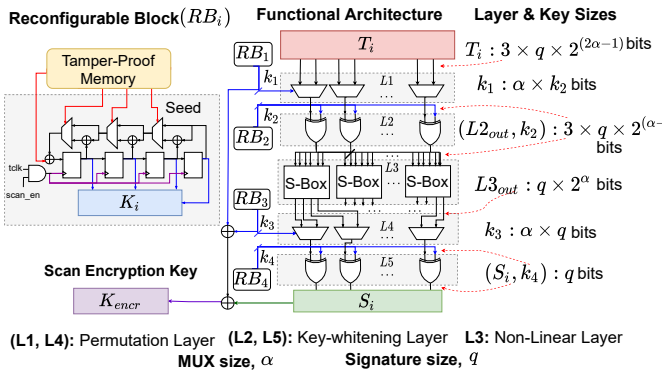

Fig. 4: Feistel structure-based SP-network used for authentication.

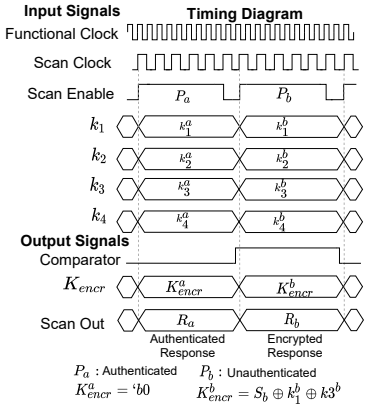

Fig. 5: Timing diagram for the different internal signals of TaintLock's authentication block.

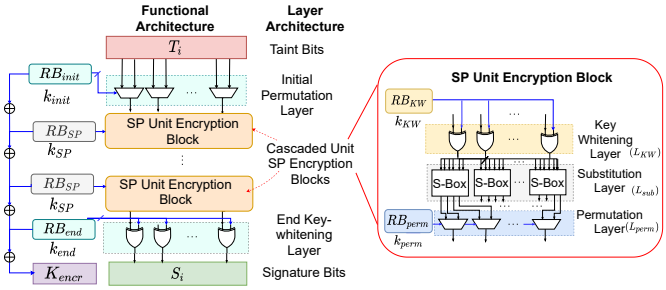

Fig. 6: General-purpose TaintLock authentication block comprising multiple SP-unit encryption blocks cascaded in an end-to-end fashion.

resent the output of the i^{th} permutation layer, and $s(t)$ represent the output of the substitution block. Then, the authentication signature is expressed as $s = k_4 \oplus p_2(k_3, s(k_2 \oplus p_1(k_1, t)))$. Scan responses are encrypted through XOR gates distributed uniformly across the scan chain. The encryption key is generated by re-purposing the block cipher to a stream cipher in counter mode, so that $k^{encr} = k1 \oplus k3 \oplus s$. The keys k_1 and k_3 are truncated to match s. If q is the size of encryption key, then the j^{th} bit of the encrypted response is encrypted with all the key bits following it, i.e., $r^{encr}_j = r_j \bigoplus_{i=j}^{q} k^{encr}_j$. A working example is presented below:

Example: Consider an authentication architecture consisting of four signature-bits. Thus, for $q = 4$, t is 24-bits wide, k_1 and k_2 are 12-bits wide, k_3 and k_4 are four-bits wide. Consider the taint-bits embedded in the test pattern to be T such that, T=`0xB12D09` (in hexadecimal). Also, consider the keys generated by the RB for two consecutive test patterns be as follows, k_1=`[0xC2F, 0xB4C]`, k_2=`[0x56A, 0x824]`, k_3=`[0x7, 0x3]`, k_4=`[0x9, 0xD]`. The signatures for both test patterns are obtained by passing the above values through F_{auth}, such that $S = F_{auth}(T, k_1, k_2, k_3, k_4) =$ `[0x4, 0xE]`. These values are compared with the signature embedded within the corresponding test patterns through the on-board comparator. In case of a mismatch, the binary key used for encryption is generated as follows: $k^{encr} = S \oplus k_1 \oplus k_3$=`[1100, 0001]`.

D. Extending TaintLock for General-Purpose Encryption

The previous sub-sections describe TaintLock's key architectural components, with the Feistel-inspired SP-network being at the heart of the authentication and encryption scheme. The overall architecture of the constituent layers associated with the authentication block allow designers to extend the baseline SP-network to support multiple rounds of encryption. This is achieved by cascading a single SP unit encryption block end-to-end multiple times, as illustrated in Fig. 6. A single SP unit encryption block consists of three layers which include a key-whitening layer (L_{KW}), a non-linear substitution layer (L_{sub}), and a permutation layer (L_{perm}) in that order. The encryption keys for both layers, L_{KW} and L_{perm}, are generated from independent RBs, namely RB_{KW} and RB_{perm}, respectively. Both the RBs are represented together as RB_{SP} to denote the RB component of a single SP unit encryption block as shown in Fig. 6. Multiple such SP unit encryption blocks can be cascaded together to support multiple rounds of encryption. Note that the general-purpose authentication block is always initialized with an initial permutation layer and a terminal key-whitening layer in the end, irrespective of the number of SP unit encryption blocks in between. It is important to note that the sizes of all the multiplexers and S-Boxes used in the authentication block is uniform across the entire architecture.

E. Optimizing Taint and Signature Cell Allocation using ILP

TaintLock promises per-pattern dynamic scan-authentication and encryption support without any impact on overall test cost. This is achieved by embedding authentication information within the don't care bits of the test pattern set. Since the percentage of don't care bits in a single test pattern is significant for large designs [18], we formulate an ILP model that minimizes the number of signature cells, r, while covering all patterns in the test set. Let us consider a pattern set with n patterns, each being m bits wide. We assume a single scan chain. However, this analysis is also applicable for multiple scan chains without any loss of generality. We use a binary decision variable x_{ij} such that $x_{ij} = 1$ if the j^{th} bit of the i^{th} pattern is selected, and $x_{ij} = 0$ if vice-versa. We next pre-compute the location of the don't care bits, i.e., the X-locations across all patterns in the pattern set and store them as matrix coefficients, where $c_{ij} = 1$ if the j^{th} bit of the i^{th} pattern is an X and $c_{ij} = 0$ otherwise.

The reward score is computed next for each bit position j, and is given as follows: $d_j = \sum_{i=1}^{n} c_{ij}\ \forall 1 \leq j \leq m$.The larger the number of X's in a bit location across all patterns in the test

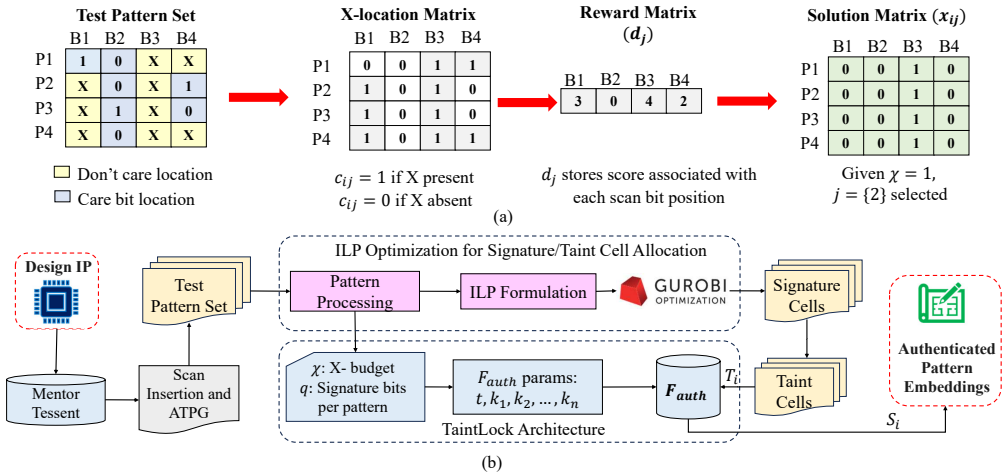

Fig. 7: (a) Illustrative example showing the different steps in the ILP-based selection of signature cells for a given test pattern set and X-budget (χ). (b) Methodology for signature/taint cell allocation using ILP and generation of authenticated test pattern embeddings in TaintLock.

set, more likely is the selection of that location as a signature cell. We also determine χ, which is the minimum number of X's present among all patterns in the test set, also referred to as the X-budget. We impose the constraint that we must select exactly q X's, i.e., q signature bits per pattern, where $q \leq \chi$. Note that n such constraints exist, one for each pattern, and is given by: $\sum_{j=1}^{m} x_{ij} = q \; \forall 1 \leq i \leq n$. Our objective is to maximize the reward function: $\sum_{j=1}^{m} d_j \cdot \sum_{i=1}^{n} x_{ij} c_{ij}$. This objective function ensures that bit locations with larger number of X's per pattern are given priority. It also ensures that all X-bits in a column are selected during scoring. An example illustrating this process for a representative test pattern set containing four patterns generated for a scan chain consisting of four scan cells is shown in Fig. 7(a).

After the r signature cells are allocated, the remaining $(m-r)$ cells associated with the specified bits in the test pattern are considered for taint cell allocation. The number of taint locations (t) is fixed for all patterns in the test set. It is also ensured that $(m-r) \geq t$, where $t = 6q$. For each remaining scan cell location, k, let a_k and b_k denote the number of 0's and 1's occurring at that location across all test patterns, respectively. As taint bits are used as inputs to the authentication block, we select taint cells that have similar numbers of 1's and 0's. This ensures that the input to the authentication block is balanced. Thus, for each k, we evaluate $\Delta_k = |a_k - b_k|$ and $\mu_k = (a_k + b_k)/2$. Subsequently, we choose k with the highest μ_k and lowest Δ_k, thereby ensuring a balanced selection of taint bits, leading to better authentication performance for the block cipher.

Fig. 7(b) shows the overall methodology for integrating the optimization process for signature and taint cell allocation while co-selecting the different parameters for a baseline TaintLock architecture to generate a set of authenticated test patterns. We evaluate the ILP model on several IPs from the Common Evaluation Platform (CEP) [2] protected using different configurations of F_{auth}. We compute the number of signature cells for IPs containing F_{auth} with $q = 12$ bits and $q = 16$ bits, respectively. These results are shown in Table I.

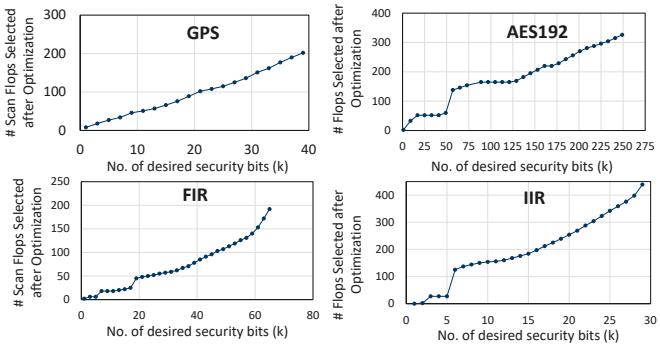

Fig. 8: Signature cell selection through ILP as a function of the required number of security bits.

As taint bits drive the block cipher, F_{auth}, taint-cell allocation is done to ensure that the input to cipher is balanced. Thus, for each bit position j, we evaluate the number of 1s and 0s across all patterns i, and select bits with close to equal number of both. We also ran incremental optimization experiments, where we increased the number of signature bits required per pattern from zero to the X-budget and evaluated the minimum number of signature cells obtained to cover all the test patterns; see Fig. 8. It can be seen that as the number of security bits required per pattern increases, the corresponding increase in the number of signature cells selected per pattern also increases in order to cover all test patterns in the test set. This is because as the bit locations with most number of $X's$ are already selected, it becomes increasingly rare to find additional X bit locations that are shared across all the test patterns.

IV. Security Analysis

A. Encryption Quality and Adversarial Indistinguishability

As TaintLock supports dynamic per-pattern authentication and encryption, based on the dynamic key schedule the same pattern can have different signatures and encrypted responses. For a test set containing n patterns, each pattern will have at least n different signatures and encrypted responses, respectively. For a given signature size and key schedule, we evaluate the *authentication quality* of a pattern P_i by analyzing

TABLE I: Evaluating the number of signature cells required to support different TaintLock versions across CEP benchmarks.

IP	Pattern Count	Pattern Length	χ	r (from ILP)	
				q=12	q=16
GPS	230	347	70	51	57
FIR	160	448	66	20	25
IIR	183	672	219	160	197
SHA256	306	1040	267	27	32
AES192	598	6854	4504	20	25
Rocket	2183	43140	39856	23	23

q: # of signature bits, r: # of signature cells, t: # of taint cells, $t = 6q$, thus $t = 72$ and 96 for 12 and 16 bit signatures, respectively, χ: X-budget.

the distribution of 1's and 0's associated with each signature bit location j ($1 \leq j \leq q$) across all n signatures. For each P_i, let the pair $(\mu_{i,1}^j, \mu_{i,0}^j)$ denote the average fraction of 1s and 0s associated with the j^{th} signature bit across all n signatures. We next evaluate the mean $(\mu_{i,1}^{sign}, \mu_{i,0}^{sign})$ and standard deviation $(\sigma_{i,1}^{sign}, \sigma_{i,0}^{sign})$ of these j ordered pairs. Here, $\mu_{i,1}^{sign}$ is the average fraction of 1's in all the signature bits for all n signatures. Similarly, $\sigma_{i,1}^{sign}$ is the standard deviation in the average fraction of 1's across the j signature bits for all n signatures. Note that because $\mu_{i,1}^{sign} = 1 - \mu_{i,0}^{sign}$, it follows that $\sigma_{i,1}^{sign} = \sigma_{i,0}^{sign}$. A value of $\mu_{i,*}^{sign}$ close to 0.5 and a very low value of $\sigma_{i,*}^{sign}$ are indicative of a balanced authentication function. Next, we use the Hamming distance between a scan response R_i, and its encrypted counterpart $R_{encr,i}$, to evaluate *encryption quality*. Given R_i, let $R_{i,encr}^j$ represent its j^{th} encrypted response such that $1 \leq j \leq n$. Then, μ_i^{HD} represents the average Hamming distance between the true response and all other n encrypted responses. A Hamming distance of 0.5 is indicative of a balanced encryption function. The authentication quality ($\mu_{i,*}^{sign}$, $\sigma_{i,*}^{sign}$) and encryption quality (μ_i^{HD}) values are averaged across all patterns in the test set and presented for IPs in Table II. We observe that the baseline version of TaintLock performs balanced authentication and encryption.

From Table II, we observe that the baseline version of TaintLock is adversarially indistinguishable, i.e., given the encrypted ciphertext, $R_{i,encd}$, an adversary can learn no partial information about the plaintext, R_i. For a given private-key encryption scheme Π, consider an experiment $PrivK_{\mathcal{A},\Pi}$ in which a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ outputs two messages m_0, m_1 out of which one of them is encrypted at random. We define $PrivK_{\mathcal{A},\Pi}$ as follows: (1) An adversary $\mathcal{A}$ is given an input 1^n and generates a pair of output messages m_0, m_1 with $|m_0| = |m_1|$. (2) A key is generated at random by running $Gen(1^n)$, and either message m_0 or m_1 is chosen at random depending on a uniform bit $b \in \{0, 1\}$. The challenge ciphertext $c \leftarrow Enc_k(m_b)$ is generated. (3) $PrivK_{\mathcal{A},\Pi} = 1$ if $\mathcal{A}$ identifies correct m_b for given c. We next introduce the following definitions [19].

Definition 1. *A function f from the natural numbers to non-negative real numbers is negligible if for every positive polynomial p, there is an N such that $\forall\, n > N$; $f(n) < \frac{1}{p(n)}$.*

Definition 2. *A given private-key encryption scheme Π is considered adversarially indistinguishable if for a PPT adversary $\mathcal{A}$, there is a negligible function $negl$ such that for all n,*

TABLE II: Evaluating the authentication and obfuscation quality for 12- and 16-bit baseline TaintLock versions across CEP benchmark IPs.

IP	12-bit			16-bit		
	$\mu_{avg,1}^{sign}$	$\sigma_{avg,1}^{sign}$	μ_{avg}^{HD}	$\mu_{avg,1}^{sign}$	$\sigma_{avg,1}^{sign}$	μ_{avg}^{HD}
FIR	0.506	0.030	0.485	0.497	0.036	0.509
IIR	0.494	0.049	0.480	0.505	0.032	0.492
SHA256	0.501	0.026	0.486	0.501	0.033	0.487
AES192	0.502	0.022	0.488	0.503	0.018	0.496
RocketCore	0.501	0.010	0.495	0.495	0.012	0.495

TABLE III: Evaluating the authentication and obfuscation quality for 12- and 16-bit cascaded TaintLock versions consisting of two SP unit encryption blocks across CEP benchmark IPs.

IP	12-bit			16-bit		
	$\mu_{avg,1}^{sign}$	$\sigma_{avg,1}^{sign}$	μ_{avg}^{HD}	$\mu_{avg,1}^{sign}$	$\sigma_{avg,1}^{sign}$	μ_{avg}^{HD}
FIR	0.490	0.089	0.498	0.513	0.016	0.512
IIR	0.487	0.079	0.487	0.504	0.052	0.508
SHA256	0.499	0.045	0.495	0.499	0.053	0.494
AES192	0.510	0.098	0.502	0.509	0.034	0.510
RocketCore	0.489	0.045	0.512	0.488	0.043	0.505

$$P[PrivK_{\mathcal{A},\Pi} = 1] \leq \tfrac{1}{2} + negl(n).$$

From the definitions above, we note that $\mu_{avg,1}^{HD}$ provides us with an indirect measure of the adversarial indistinguishability of an encryption scheme. From Table II, we observe that $\mu_{avg,1}^{HD} \approx 0.5$ for all IPs analyzed. This implies that there exists an equal probability of the scan response bit being encrypted or not, thus indicating the adversarial indistinguishablity of the baseline version of TaintLock.

We next also compute the same metrics for an extended version of the baseline TaintLock architecture comprising of two cascaded SP unit encryption blocks for the same signature sizes. These results are also presented in Table III. Both the baseline version of TaintLock and the cascaded version comprising of more rounds of substitution and permutation perform well in terms of their quality of encryption and adversarial indistinguishability. The difference in μ^{sign} and μ^{HD} between both versions is statistically insignificant. As a result, the baseline version of the TaintLock architecture may be sufficient for most security applications.

B. Security Analysis

TaintLock prevents Oracle access by using a decentralized and non-intrusive method for cryptographically authenticating and encrypting scan access. Pairing it with a low overhead and high output corruptibility IP obfuscation method such as truly random logic-locking (TRLL) makes it immune to existing netlist analysis-based removal attacks [5]. In this section, we evaluate TaintLock's ability to resist a wide range of attacks such as cryptanalysis attacks, Oracle-guided attacks, and Oracle-free attacks including state-of-the-art scan deobfuscation attacks.

1) Brute-force Resilience

In this sub-section, we compare the brute-force attack efforts for different different signature sizes of the baseline TaintLock architecture. Given q signature bits per pattern, an attacker can attempt to guess S_i for a given P_i. The probability of success $P(S_i|P_i)$ for the attacker is $1/2^q$. However, due to dynamic per-pattern authentication, S_i will change not only based on the pattern itself but also based on the order of patterns in the test

TABLE IV: Comparison of different brute-force attack efforts for various versions of the baseline TaintLock architecture, determined by q.

q	(k_1, k_2, k_3, k_4)	λ_{sum}	$n \times 2^q$	$t_{bf} = 2^{\lambda_{sum}}$
10	$(30, 30, 10, 10)$	156	1.23×10^5	9.13×10^{46}
12	$(36, 36, 12, 12)$	188	4.91×10^5	3.92×10^{56}
14	$(42, 42, 14, 14)$	220	1.96×10^6	1.68×10^{66}
16	$(48, 48, 16, 16)$	252	7.86×10^6	7.23×10^{75}

For this analysis, the average number of patterns n is 120, which is an optimistic choice in the context of oracle-guided attacks.

set. For n patterns, the attacker must try all $n \times 2^q$ signature combinations to find the correct S_i for each P_i. Furthermore, this does not *unlock* the IP as the attacker will be forced to repeat this process in case of a new pattern set. Thus, the attacker is forced to guess the seed used to generate the correct keys, (k_1, k_2, k_3, k_4), to unlock the IP. Note that due to the use of RBs, if λ_i is the size of the i^{th} key, then the size of the RB seed used to generate that key is $2\lambda_i - 1$. Brute-force attack resilience is quantified by the number of attempts required by the attacker to uncover the correct key. For TaintLock, this is given by $t_{bf} = 2^{\lambda_{sum}}$, where $\lambda_{sum} = 2(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) - 4$ is the total seed size. The attack effort is prohibitively high for the different brute-force approaches discussed above (For e.g., $t_{bf} = 3.92 \times 10^{56}$ for $q = 12$). More data is presented in Table IV. Note that cascaded SP networks provide significantly more security against brute-force attacks. This is because the key size used for encryption increases with the number of intermediate SP networks. Furthermore, since brute-force attack effort increases exponentially with the key size, increasing the layer count linearly may give significantly better security against such attacks

2) Cryptographic Attacks

Given access to authenticated debug patterns, P_{auth}, the attacker may apply known-plaintext attacks [20]. However, TaintLock is resilient to both known-plaintext (KPA) and chosen-plaintext attacks (CPA). If the attacker can collect 2^m plaintext-ciphertext pairs, i.e., the number of patterns in P_{auth} is 2^m, the resilience of the system will be reduced to $2^{\lambda_{sum}-m}$, where $\epsilon = \lambda_{sum} - m$. However, given the small size of P_{auth}, $m << \lambda_{sum}$. This makes such attacks infeasible in practice. Additionally, given that TaintLock obscures the circuit test responses, the attacker does not have control over the original patterns being scanned out of the design, making chosen-plaintext attacks also infeasible. We first define the notion of security in context of a PPT adversary $\mathcal{A}^{\mathcal{PPT}}$ that can make queries in polynomial time [21].

Definition 3. *An encryption scheme Π is ϵ-secure if a PPT adversary $\mathcal{A}^{\mathcal{PPT}}$ after $p(\epsilon)$ queries cannot unlock the scheme with a probability greater than $\frac{p(\epsilon)}{2^\epsilon}$.*

In the above definition, $p(\epsilon)$ represents a polynomial number of queries. The following theorems characterize the security guarantees provided by TaintLock.

Theorem 1. *TaintLock is ϵ-secure against known-plaintext attacks.*

Proof. If the attacker can collect 2^m plaintext-ciphertext pairs, i.e., the number of patterns in P_{auth} is 2^m, the resilience of the system will be reduced to $2^{\lambda_{sum}-m}$, where $\epsilon = \lambda_{sum} - m$. However, given the small size of P_{auth}, $m << \lambda_{sum}$ implying that ϵ is a very large value. The probability that the attacker can unlock the encryption scheme after $p(\epsilon)$ queries is $\frac{1}{2^\epsilon - p(\epsilon)}$. Therefore, the probability P_l that the attacker cannot unlock the encryption scheme after $p(\epsilon)$ queries is: $P_l = 1 - \frac{1}{2^\epsilon - p(\epsilon)} = \frac{2^\epsilon - p(\epsilon) - 1}{2^\epsilon - p(\epsilon)} > \frac{2^\epsilon - p(\epsilon) - 1}{2^\epsilon}$. As $p(\epsilon)$ is polynomial in the number of queries, $p(\epsilon) < 2^{\epsilon - 1} \implies 2^\epsilon - p(\epsilon) > p(\epsilon)$. Consequently, $P_l > \frac{2^\epsilon - p(\epsilon) - 1}{2^\epsilon} \approx \frac{2^\epsilon - p(\epsilon)}{2^\epsilon} > \frac{p(\epsilon)}{2^\epsilon}$. This makes TaintLock ϵ-secure against known-plaintext attacks, rendering such attacks infeasible in practice. □

We illustrate the implication of the theorem above through a real-world example. Let us consider the number of patterns in a large IP such as RocketCore. From Table I, we observe that RocketCore has close to 2200 test patterns when rounded up to the nearest hundred. The closest whole value of m for that case is 12. Assuming that RocketCore is protected using the baseline version of the TaintLock architecture while supporting a 10-bit signature, the value of $\epsilon = 156 - 12 = 144$. In practice, the attacker will need close to 2^ϵ patterns to mount an effective attack, which is in the order of 10^{43}, a very large number.

Theorem 2. *The TaintLock encryption scheme is resilient against chosen-plaintext attacks.*

Proof. By construction, the TaintLock architecture does not support scan-out of unauthenticated test patterns. Thus, unauthenticated test responses are encrypted dynamically based on the pattern order. As a result, plaintext $m \leftarrow R$ remains unavailable to the adversary $\mathcal{A}^{\mathcal{PPT}}$. As only the ciphertext $c \leftarrow R_{encr}$ remains available, chosen-plaintext attacks become infeasible in practice. □

More recent attacks mounted on dynamically obfuscated scan chains, such as DynUnlock and ScanSAT, aim to model the obfuscation logic consisting of the LFSRs into a logic-locked combinational design, with the key being the obfuscation key (LFSR seed). They can then utilize combinational SAT attack using the obfuscated test responses. This is because: (1) The attackers can create obfuscation logic does not (2) The use of non-linear cryptographic functions make

3) Oracle-guided and Oracle-free Attacks

We next show that Oracle-guided attacks such as the SAT attack are ineffective in case of TaintLock.

Theorem 3. *TaintLock makes the Oracle dysfunctional in unauthenticated mode.*

Proof. Oracle-guided attacks rely on scan chains to provide functionally correct input-output pairs. For unauthenticated test patterns, $\mathcal{A}^{\mathcal{PPT}}$ will only have access to R_{encr}. Since TaintLock is adversarially indistinguishable, almost half the output bits are flipped at random. Thus, the Oracle response becomes dysfunctional. □

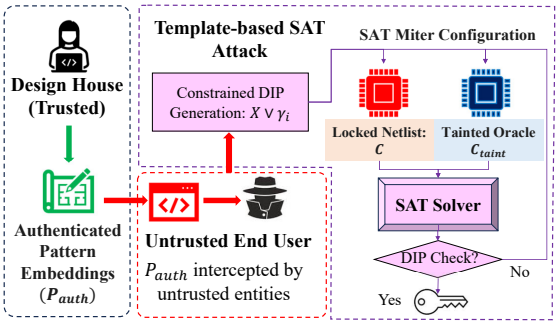

Fig. 9: Illustration of Template-based SAT attack with constrained DIPs generated from leaked authenticated test patterns (P_{auth}) using a tainted Oracle (C_{taint}).

Input: $C, eval$ ▷ $eval$ generates scan response
Output: K_c
$i \leftarrow 1$;
$F_1 \leftarrow C((\overrightarrow{X} \vee \gamma_i), \overrightarrow{K_1}, \overrightarrow{Y_1}) \wedge C((\overrightarrow{X} \vee \gamma_i), \overrightarrow{K_2}, \overrightarrow{Y_2})$;
while $sat[F_i \wedge (\overrightarrow{Y_1} \neq \overrightarrow{Y_2})]$ **do**
  $\overrightarrow{X_i} \leftarrow sat\_assignment_{\overrightarrow{X}}([F_i \wedge (\overrightarrow{Y_1} \neq \overrightarrow{Y_2})])$;
  $\overrightarrow{Y_i} \leftarrow eval(\overrightarrow{X_i})$;
  $F_{i+1} \leftarrow F_i \wedge C((\overrightarrow{X_i} \vee \gamma_i), \overrightarrow{K_1}, \overrightarrow{Y_1}) \wedge C((\overrightarrow{X_i} \vee \gamma_i), \overrightarrow{K_2}, \overrightarrow{Y_2})$;
  $i \leftarrow i + 1$;
end
$\overrightarrow{K_c} \leftarrow sat\_assignment_{\overrightarrow{K_1}}(F_i)$;
return K_c

Algorithm 1: Procedure for template-based SAT attack.

We show in [22] that a dysfunctional IP guarantees resilience against all forms of Oracle-guided attacks. TaintLock encrypts test response data thereby making it resilient against attacks. We later show that the attacker can use a *tainted Oracle*, by utilizing partial authentication information, to attempt to reconstruct locking keys. However, such an attack no longer fits under the purview of an Oracle-guided attack.

Scan deobfuscation attacks such as ScanSAT and DynUnlock belong to a class of Oracle-free that aim to extract the LFSR seed used for dynamic scan obfuscation. These attacks model the obfuscation logic responsible for pattern transformation into a combinational cone of logic-locked netlist activated using the LFSR seed. Applying the SAT attack on this locked netlist allows such attacks to reverse engineer the LFSR seed. These attacks are successful because: (1) the feedback polynomial of the LFSR used for dynamic obfuscation is known to the attacker from the RE'd netlist, allowing them to model the pattern transformation as a function of the LFSR seed; (2) the obfuscation logic does not contain non-linear blocks, allowing easier construction of their combinational logic equivalent. Both of these drawbacks are addressed by TaintLock. The use of LFSRs with reconfigurable feedback (RBs) makes it impossible for the attacker to extract the correct feedback polynomial used for key generation from the RE'd netlist. Moreover, the attacks described above cannot model the encryption blocks used in the SP-network (permutation layers and S-boxes) into a combinational design logic-locked using the RB seed due to their non-linear nature [7], [8]. Finally, resetting the IP clears data in scan cells including signature values, triggering encryption. This makes TaintLock resilient against state-of-the-art scan deobfuscation attacks.

4) Template-based SAT Attack

Although traditional SAT attack remains ineffective against TaintLock, we consider the scenario where the attacker leverages the already available set of authenticated test patterns, P_{auth}, to mount a variation of the SAT attack aimed at recovering partial keys to the combinationally locked netlist. Based on the strongest version of the threat model, we assume that the attacker not only has access to the reverse engineered netlist of the locked design but is also aware of the location of the taint and signature bits in P_{auth}. As a result, the attacker has the ability to modify specific bit locations in every pattern while keeping the taint and signature locations within those patterns unchanged. This allows the attacker to partially control certain bit locations of each pattern in the test set thereby providing them with partial scan access. As the attacker must maintain the taint and signature bits for each test pattern along with the order in which the pattern is scanned in, there are additional constraints imposed on the generation of DIPs (distinguishing input patterns) thereby making such an attack more difficult.

Suppose $\mathcal{A}^{\mathcal{SAT}}$, has access to the locked netlist C and a tainted Oracle, C_{taint} protected using TaintLock. C_{taint} only produces the correct output when supplied with the embedded security information. Assuming that the attacker has access to P_{auth}, let Γ_{mask} represent that set of all patterns (in the same order as in P_{auth}) with non-security bits set to 0. Thus, every pattern in Γ_{mask} when OR'ed with the corresponding test pattern in P_{auth} will generate the same test pattern. Thus, $\mathcal{A}^{\mathcal{SAT}}$ aims at finding the key vector $\overrightarrow{K_c}$ by solving the following QBF (quantified Boolean formula): $\exists \overrightarrow{K_c} \forall \overrightarrow{X_i} \in I : C((\overrightarrow{X_i} \vee \gamma_i), \overrightarrow{K_c}, \overrightarrow{Y_i}) = C_{taint}((\overrightarrow{X_i} \vee \gamma_i), \overrightarrow{Y_i})$, where $\overrightarrow{X_i}$ and $\overrightarrow{Y_i}$ are the i^{th} inputs and outputs of the circuit and γ_i is the i^{th} pattern in Γ_{mask}. The overall procedure is shown in Algorithm 1 and illustrated in Fig. 9. The attack creates a miter circuit indicated by the SAT formula, F_i, where the same $\overrightarrow{X}$ when applied with the constraint γ_i produces two different outputs for two different keys. The SAT solver tries to create a SAT assignment, iterating over more values of $\overrightarrow{X}$, until all incorrect keys are eliminated.

We implemented this attack on CEP benchmark IPs locked using the baseline version of the TaintLock architecture and considered a signature size of 12 bits and 16 bits respectively. The combinational netlist was locked using 128 embedded TRLL key gates. The RANE platform was repurposed to mount the template-based SAT attack [23]. All experiments were run on an Intel Xeon CPU. We evaluated the effectiveness of three powerful SMT solvers including Yices [24], Z3 [25], and MSAT [26] solvers. Table V summarizes the results. We ovserve that none of the solvers are able to converge to the correct keys. This can be explained by comparing the QBF for $\mathcal{A}^{\mathcal{SAT}}$ with that of the regular SAT attack. As compared to the regular SAT attack, template-based SAT attack imposes additional constraints on the miter configuration, making the process of

TABLE V: Evaluating the effectiveness of template-based SAT attack on CEP benchmark IPs protected using 128 bit TRLL keys and the baseline version of the TaintLock architecture.

IP	12-bit			16-bit		
	Yices	Z3	MSAT	Yices	Z3	MSAT
FIR	TO	TO	TO	TO	TO	TO
IIR	TO	TO	TO	TO	TO	TO
SHA256	TO	TO	TO	TO	TO	TO
AES192	TO	TO	TO	TO	TO	TO
RocketCore	TO	TO	TO	TO	TO	TO

TO: Timeout due to non-convergence of the SAT solver, limited to 72 hours.

TABLE VI: Number of SAT clauses generated by applying the internal KC2 solver with template-based SAT attack constraints on CEP benchmark IPs locked using 128 bit TRLL keys and 12-bit signature.

Benchmark IP	Number of SAT Clauses
FIR	1.7×10^4
IIR	1.8×10^4
SHA256	2.8×10^4
AES192	2.3×10^5
RocketCore	3.5×10^5

finding DIPs to solve the SAT assignment in Algorithm 1 very challenging. In addition to finding equivalence classes of keys that satisfy $(\overrightarrow{Y_1} \neq \overrightarrow{Y_2})$, every DIP must also satisfy the masking constraint to maintain authenticated Oracle access, i.e. $C(\overrightarrow{X_i} \vee \gamma_i)$. Moreover, as $i \leftarrow i+1$, the new masking state γ_{i+1}, also changes (due to changing location of the security bits) and is added to the original SAT assignment.This severely limits the ability of the SAT solver to eliminate large categories of equivalence key classes with the addition of every new pattern, eventually exhausting all patterns in the authenticated test set, P_{auth}. We also evaluated the number of SAT clauses that are generated until time out when a sequential deobfuscation attack such as the KC2 attack is applied on CEP benchmark IPs locked with TaintLock under the constraints imposed by template-based SAT attack [27]. This data is presented in Table VI and shows the increasing complexity of the template-based SAT attacks with the size of the IP as the number of clauses increases significantly for larger IPs. Recent prior work has also shown that increasing the number of SAT clauses leads to a further increase in complexity of SAT-based deobfuscation, showing that exponential growth in the number of SAT clauses also increases deobfuscation time exponentially [28]. In [28], the SAT solver is unable to converge to a solution with less than 2×10^4 clauses even after 10 days. This shows the increasing difficulty associated with applying the template-based SAT attack to IPs protected with TaintLock.

5) Oracle Reconstruction Attacks

A new class of machine learning-based attacks that are aimed reconstructing the Boolean functionality of the locked combinational part of the netlist have been proposed. Such attacks include LORAX [11], which relies on gaining access to a limited number of functionally correct IO-pairs from the circuit and utilizing them to train ensemble ML models to predict the Boolean functionality of the locked cone of combinational logic. Such attacks assume that only a limited number of IO-pairs are available for training ($<< 2^m$, where m is the number of primary inputs in the circuit).

Attacks such as LORAX can be potentially applied against TaintLock as follows:

- As this attack is only applicable to the part of the netlist containing combinational logic locking, the attacker will have to gain access to the different cones of logic protected using TRLL in a design protected using TiantLock.
- In order to gain access to these combinational logic cones, the attacker will be forced to utilize scan chains. This is because not all combinational cones in a design are accessible exclusively through the PIs of the design.
- Let $\mathcal{T}$ denote the set of training samples used to train LORAX on a locked cone of combinational logic. As the scan chains are protected though TaintLock, every response $P_{res} \in \mathcal{T}$ will be encrypted. As a result, LORAX will be trained on encrypted responses to scanned in patterns.

To evaluate the effectiveness of a LORAX-based Oracle reconstruction attack on IP protected using TaintLock, we encrypted the combinational responses of the scanned in test patterns for a wide variety of IWLS 2020 benchmark circuits locked using TRLL and TaintLock [29]. The IWLS 2020 benchmarks were used for analysis in [11], and thus were also used in this study to perform a fair comparison. Fig. 10 compares the effectiveness of the Oracle-reconstruction attack using decision tree and random forest-models when applied to different cones of logic protected using the TaintLock architecture. We observe that the reconstruction accuracy for majority of the benchmark circuits after passing through the TaintLock encryption scheme is close to 50%, which is equivalent to having a random guess.

V. Testability Analysis

In this section, we describe how TaintLock can support test and debug during different stages of the product lifecycle including manufacturing/fabrication, activation, and test during in-field deployment.

A. Testability Support during Product Lifecycle

Manufacturing Test: TaintLock can support manufacturing test using scan-based ATPG in the presence of dummy keys. The IP remains deactivated without loading the tamper-proof memory. There is no impact on test coverage due to the extraction of taint bits directly from scan-based test patterns generated from ATPG. Furthermore, signature bits can be pre-computed and embedded in each test pattern by replacing the don't-care bits (X) within the ATPG patterns. The original scan architecture remains unchanged. In the next sub-section, we discuss how TaintLock can support EDT-based compressed test pattern generation. In Section VI, we also show that the combinational path delay of the authentication block responsible for on-chip signature generation is negligible, especially when compared to other logic paths and the clock frequency of the design. Moreover, TaintLock supports two-pattern delay tests such as launch-on-shift (LOS)/launch-on-capture(LOC) in encrypted mode. For these tests, the first pattern is applied with the embedded signature. However, the second pattern is obtained either from the circuit's response

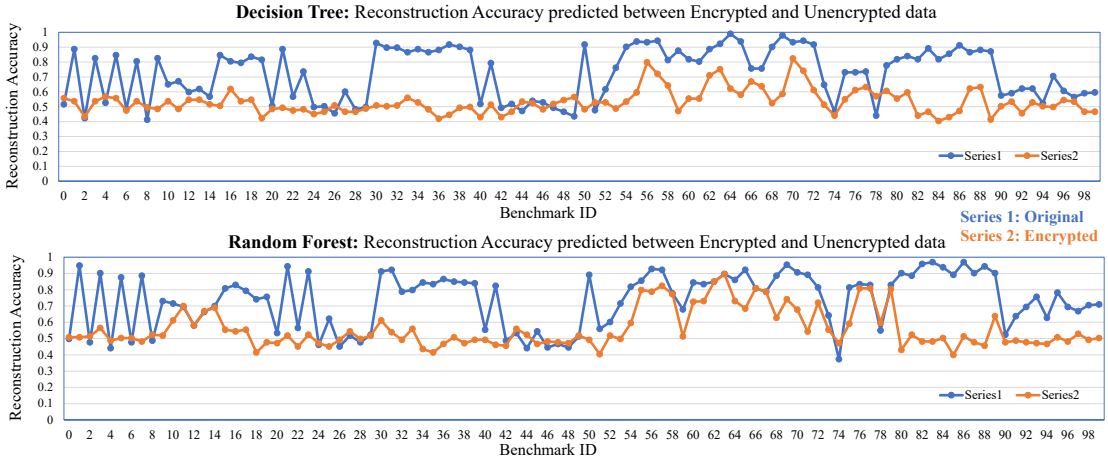

Fig. 10: Reconstruction accuracy for Oracle reconstruction using LORAX-based decisions trees (top) and random forest (bottom) ensemble models for benchmark logic cones protected using TaintLock architecture.

(LOC), or by shifting a single bit (LOS). Due to the lack of control over the bits in the second pattern, it is unlikely that the correct signature will be embedded for that pattern. Thus, the scan responses for LOC/LOS will be encrypted. However, because TaintLock uses a symmetric cryptographic scheme, a trusted user can decrypt the scan responses using the encryption key. Furthermore, the use of unique keys per pattern ensures that the encrypted response can be uniquely mapped to the expected response. This allows the user to pre-compute the expected fault-free encrypted responses for LOC/LOS tests.

Testability of TaintLock-Specific Peripherals: The process to test TaintLock's peripheral blocks, namely selection MUXes, authentication blocks, comparator, and encryption units will not be any different than the process discussed above. As there are no intermediate unscanned flops in TaintLock's peripheral logic, the path used for on-chip signature generation remains covered by scan flops. Since manufacturing test is performed without IP activation, the tamper proof memory is not loaded until after manufacturing, when the fabricated product is shipped back to the design house for placement and assembly of the tamper-proof memory. Thus, inputs to the tamper-proof memory can also be utilized as test points for structural testing. Furthermore, due to the purely combinational nature of TaintLock's authentication block, it will receive full scan coverage. The registers used to store K_{encr} are also scan driven. Finally, all RBs can be tested as part of scan chain integrity testing by loading a combination of test patterns identical to march testing through the input test points directly driving the interface of the tamper-proof memory.

IP Activation: TaintLock supports seamless IP activation process after the tested IP is shipped back to the design house. The activation process is done in a trusted site. The design house can generate additional test patterns to ascertain IP integrity during functional mode. This also involves additional steps to diagnose issues in manufacturing that lead to functional failure, which include specifically targeting functionally critical areas of logic using targeted tests [30], [31]. During this step, the designers also have the opportunity to identify malicious modifications in the IP, including scan chains, control logic, or hardware Trojans embedded within the IP.

In-field Test and Diagnosis: TaintLock can be integrated with built-in self test (BIST) techniques. Since BIST requires the use of pre-generated test patterns, authenticated BIST patterns and their corresponding signatures can be pre-computed depending on the keys used for IP activation. Furthermore, to maximize test coverage, the designers can choose to operate BIST in obfuscation mode, where the input test patterns are pre-generated to cover maximum number of faults. Since the designers also know the scan encryption key, K_{encr}. for every test pattern, they can also compute the golden signatures accordingly. Furthermore, in the next sub-section, we show that TaintLock's scan encryption mechanism does not introduce additional aliasing in the computed output signatures, thereby indicating that it will have no impact on fault diagnosis as compared to a baseline BIST implementation without TaintLock. As TaintLock is a plug-and-play solution, note that designers can integrate authentication-aware test-plans for both BIST and ATPG-based structural testing during IP design and development stage itself.

B. Test Compression Support

TaintLock supports advanced test compression architecture including embedded deterministic test (EDT) [32]. EDT-based methods achieve effective compression by supplying EDT test patterns through a limited number of scan channels which are then decompressed on-chip to concurrently drive a large number of scan patterns through multiple scan chains in the design. The effective compression achieved by EDT is limited by the chain-to-channel ratio or compression factor, Ω, which is the ratio between the the number of scan chains in the design and the number of input scan channels supplying test patterns to the design. Higher compression is achieved if a small number of input scan channels supply compressed patterns to a large number of scan chains in the design. Aggressive EDT-based compression schemes may compete with TaintLock's signature embedding, therefore leading to some drop in test coverage. However, unlike other scan obfuscation methods that require test compression engines such as EDT to always be

set to bypass mode [12], TaintLock is compatible with test compression due to the following reasons:

1) Test cubes in large designs today have a significant percentage of don't care bits to support large compression ratios along with security bit embedding.
2) EDT-based compression schemes provide designers with the ability to place constraints on specific locations of scan cells, providing flexibility with signature embedding while achieving high compression ratios with minimum drop in test coverage.
3) TaintLock's obfuscated responses do not cause aliasing when paired with EDT. This implies that any loss in test coverage due to TaintLock's signature embedding with EDT compression can be recovered by operating TaintLock in obfuscation mode.

Note that commercial EDA tools such as Siemens Tessent allow EDT-based test patterns to be generated alongside baseline ATPG, which do not have compression enabled by default. TaintLock can be utilized in both these scenarios. Designers have the ability to apply constraints to specific scan cells in the IP during EDT-based compressed test pattern generation, thereby enabling them to embed security bits during the compressed test pattern generation process itself. Adding constraints for signature embedding during EDT-based test generation can have some impact on the test coverage. In Table VII, we evaluate the test coverage impact of embedding signature bits of varying sizes across a wide range of EDT-driven compression ratios for IPs of varying sizes. We observe that increasing compression ratios (from 10x to 100x) leads to some drop in the baseline test coverage. This is because aggressive compression leads to a higher utilization of don't care bits, thereby targeting less number of faults. We also observe that increasing the size of signature embedding (from 8 bits to 16 bits), for the same compression ratio, can also lead to some drop in test coverage. However, note that the average drop in test coverage across all compression ratios and all IPs is less than 2%. Also note that for very large IPs such as AES192, which contain hundreds of thousands of gates, the average drop in test coverage is less than 0.6%. This is because of the large number of don't-care bits associated with large designs, as is also evident from Table I.

There may exist scenarios where extremely large compression ratios are necessary. Furthermore, situations may arise where the baseline test pattern set is updated to target new faults and not all don't care locations overlap for each pattern. In such situations, TaintLock can remain operational in obfuscation mode. Through Theorem 4 below, we show that TaintLock can continue operating in obfuscation mode, i.e., no signature bits embedded in the supplied test patterns, without causing any aliasing when the design's obfuscated output response is passed through the EDT-based response compactor. Theorem 4 shows that the obfuscated test responses generated by TaintLock will maintain their uniqueness in the signature so long as the uniqueness of the golden (non-obfuscated) test responses is maintained.

TABLE VII: Impact of embedding TaintLock's signature bits of various sizes on test coverage across different compression ratios.

IP	# Gates	# Scan Cells	Ω	Test Coverage (%)			
				Baseline	8-bit	12-bit	16-bit
IIR	16145	672	10	99.71	99.70	99.70	99.69
			25	99.05	98.14	97.82	97.85
			50	98.32	97.74	97.60	96.84
			100	97.93	97.25	96.79	95.29
FIR	11193	448	10	99.72	99.62	99.57	99.56
			25	99.41	98.68	98.58	98.24
			50	99.10	98.45	97.97	97.22
			100	98.85	98.07	97.58	94.83
SHA-256	28440	1040	10	99.63	99.49	99.40	99.36
			25	97.34	96.79	95.88	93.74
			50	94.23	94.22	92.97	91.21
			100	94.01	92.33	91.89	90.51
AES-192	469321	6854	10	98.97	98.97	98.97	98.87
			25	98.93	98.87	98.87	98.60
			50	98.89	98.87	98.54	97.90
			100	98.86	98.84	98.37	97.47

Ω: Compression ratio, i.e., ratio between number of scan chains in the design and number of input scan channels for EDT.

Theorem 4. *For EDT-based response compaction using XOR trees, obfuscated test responses do not lead to aliasing if unobfuscated test responses do not lead to aliasing.*

Proof. Consider an XOR-based test compactor with N scan chains feeding M compactor outputs ($N > M$). Consider an n bit scan slice feeding an n:1 XOR-tree in the compactor ($n \leq N$). Let E denote the set of bits in V that are identical between the unobfuscated fault-free (R_u) and faulty responses (R'_u), and D denote the set of bits in R_u that are complimentary to those in R'_u due to fault propagation. Consequently, D' denotes the corresponding set of bits with complementary values in R'_u. Let (e_0, e_1) be the ordered pair of 0- and 1-counts in E and (d_0, d_1) be the ordered pair of 0- and 1-counts in D. Therefore, 0- and 1-counts for D' are (d_1, d_0).

The output of the XOR-tree is 0 (1) if the 1-count of the input bits is even (odd). Without scan obfuscation and aliasing, R_u produces a different output signature than R'_u after compaction. Thus, $(e_1 + d_1)$ must have a different parity than $(e_1 + d_0)$ $\implies$ d_0 and d_1 must have different parities $\implies$ $(d_0 + d_1)$ is an odd number, as the sum of two numbers with different parities is always odd. Upon enabling obfuscation, both R_u and R'_u undergo the same transformations, leading to some bits in E and D undergoing even or odd number of flips.

For R_u, let the 1-count in the component of D undergoing odd (even) bit-flips be d_1^o (d_1^e) and let the 0-count in the component of D undergoing odd (even) bit-flips be d_0^o (d_0^e). Note that for R'_u, the 1-count in the component of D' undergoing odd (even) bit-flips is equal to the 0-count in the corresponding component of D. Without scan obfuscation, there is no aliasing and $d_{sum} = d_0 + d_1 = (d_0^o + d_0^e) + (d_1^o + d_1^e)$ is odd. With scan obfuscation, the 1-count in D is $d(R_u) = d_1^e + d_0^o$. This is because the 1-count in the obfuscated response is determined by the 1-count of the original response affected by even number of bit-flips and 0-count of the original response affected by odd number of bit-flips. Similarly, the 1-count in D' of transformed R'_u is $d(R'_u) = d_1^o + d_0^e$. As E contains equal 1-counts for R_u and R'_u after transformation, and $d(R_u) + d(R'_u)$ is odd, it implies that the if the 1-count in the transformed R_u is odd, then the corresponding 1-count in the transformed R'_u must be even, and vice-versa. Therefore, R_u produces different

compacted signature than R'_u. □

VI. Overhead Analysis

To evaluate its power, performance, and area (PPA) overhead, TaintLock is integrated with different IPs from the CEP benchmark circuit suite [2]. The overhead is computed for 12-bit and 16-bit signature versions of the TaintLock architecture. It has been shown in previous sections that 12-bit and 16-bit signature versions of TaintLock provide strong security metrics and are paired with 128-bit TRLL logic-locked netlist. For the experiments, all benchmarks were synthesized with the Nangate 45 nm standard cell library using Synopsys Design Compiler. Timing closure was achieved at a frequency of 500 MHz. Place and route was then run on the synthesized designs, followed by parasitic extraction using Cadence Innovus to generate the final design layout. To evaluate circuit timing impact and power consumption, Cadence Tempus was used.

Table VIII shows TaintLock's PPA overhead on five CEP benchmark IPs. We observe that TaintLock has minimal impact on area and timing overhead of the designs. This is because the TaintLock authentication functions are purely combinational in nature, thereby leading to no impact on scan frequency and thus, supporting at-speed testing. TaintLock's authentication and obfuscation logic have fixed area for a given security architecture. Thus, the 12-bit or 16-bit signature architecture when paired with 128-bit TRLL keys will produce the same logical area irrespective of the size of the IP being protected. There may be changes in the wire length due to variability in routing, but this will not have any impact on the logic overhead. As a result, for the same architectural configuration selected for TaintLock, both area and power overhead will decrease with an increase in the size of the IP under protection.

RocketCore is the largest IP among the CEP benchmark suite, (over 200,000 gates). TaintLock has an area overhead of less than 0.2% for RocketCore. Although TaintLock does not modify the functional implementation of the design, it may impact the floorplan of the IP due to optimizations in the routing paths available for scan stitching, thereby affecting wire delay. From Table VIII, we observe that this impact on wire delay is minimal. Finally, the latency of generating the encryption keys, K_{encr}, is evaluated at ~ 1.3 ns, which is less than the functional clock period of 2 ns. The results show that the area and the timing impact of TaintLock are negligible, and the authentication function can compute the signatures from test patterns in real-time without impacting scan clock frequency and also support at-speed testing.

Table IX presents a comparison of the TaintLock architecture with other similar scan obfuscation methods. We compare TaintLock with methods such as LCSS [9], SSTKR [10], scan encryption using block ciphers (SEBC) [14], scan encryption using stream ciphers (SESC) [33], DOSC [12], and parallel latch-based lock [15]. TaintLock offers significant advantages over other methods in the following categories: (1) Security against SAT-based deobfuscation attacks such as ScanSAT and DynUnlock. LCSS, DOSC, and SSTKR do not offer resistance against these class of attacks. (2) Support for dynamic per pattern authentication. Existing encryption based methods like SEBC, SESC, and parallel lath-based lock do not support dynamically changing signatures for authentication. (3) Support for test compression unlike DOSC, LCSS, and SSTKR. (4) Support for multi-pattern tests such as LOC/LOS with no timing overhead unlike parallel latch-based lock. In summary, TaintLock offers a cryptographically strong method to not only authenticate but also encrypt scan data. Further, it offers a decentralized and non-intrusive way to embed scan security with any of the existing logic locking techniques.

TABLE VIII: PPA overheads associated with TaintLock and PRESENT.

IP	Security configuration	Cell area (μm^2)	Wirelength (μm)	Power consumption (mW)	Δ_{crit} (ns)
FIR	Baseline	6010.5	28103.1	2.88	1.98
	Δ_{prsnt} (%)	30.64	29.03	17.69	0.71
	Δ_{12} (%)	6.38	10.47	3.13	0.22
	Δ_{16} (%)	7.5	14.65	3.94	0.72
IIR	Baseline	9864.6	46255.5	7.02	1.98
	Δ_{prsnt} (%)	18.66	17.64	7.26	8.78
	Δ_{12} (%)	4.19	6.07	3.52	0.27
	Δ_{16} (%)	4.83	11.78	4.24	0.42
SHA256	Baseline	15077.6	114915.1	7.26	1.98
	Δ_{prsnt} (%)	12.21	7.1	7.02	0.05
	Δ_{12} (%)	2.64	5.42	2.16	0.12
	Δ_{16} (%)	3.57	7	4.4	0.05
AES192	Baseline	267278.4	3794342.7	71.91	1.99
	Δ_{prsnt} (%)	0.69	0.21	0.71	0.45
	Δ_{12} (%)	0.13	1.66	0.04	0.6
	Δ_{16} (%)	0.16	2.61	0.54	0.13
RocketCore	Baseline	375386.6	2979859.8	71.1	1.68
	Δ_{prsnt} (%)	0.49	0.28	0.72	0.45
	Δ_{12} (%)	0.11	2.05	0.06	0.85
	Δ_{16} (%)	0.13	2.03	0.54	0.42

Δ_{prsnt}: percentage overhead associated with the insertion of PRESENT; $\Delta_{12}(\Delta_{16})$: percentage overhead associated with the insertion of 12-bit (16-bit) signature version of TaintLock; Δ_{crit}: critical path delay; Units are for Baseline values only.

To present a head to head comparison with another state-of-the-art encryption scheme, we compare TaintLock with a lightweight cryptographic encryption scheme that may be re-purposed for scan authentication, called PRESENT [34]. Unlike PRESENT, TaintLock provides security against not only known- and chosen-plaintext attacks, but also against Oracle-guided attacks. Moreover, it does so at over $3\times$ lower area overhead. As shown in Table VIII, the overall PPA footprint of TaintLock is lower as compared to PRESENT. In addition, PRESENT requires multiple rounds of encryption that take up significant time, thereby adversely impacting the scan shift frequency. PRESENT also has a fixed block size, thereby is not parameterizable and hence not useful for small IP. Our results show that a lightweight scan authentication scheme using taint bits, such as TaintLock, can successfully offer security against state-of-the-art attacks.

VII. Conclusion

We have presented the vulnerabilities associated with existing scan data authentication techniques and how TaintLock, a low overhead authentication and encryption method that uses embedded signature and taint bits, can address them. We have evaluated the effectiveness of TaintLock against a variety of SAT-based Oracle-guided attacks and machine learning-based Oracle-free attacks. We have also shown that TaintLock does not have any impact on circuit testing and can support existing test compression schemes.

TABLE IX: Comparing prior secure scan methods with TaintLock.

Metrics	LCSS [9]	SSTKR [10]	SEBC [14]	SESC [33]	DOSC [12]	Parallel Latch-based Lock [15]	TaintLock
Scan deobfuscation attacks [7], [8]	✗	✗	✓	✓	✗	✗	✓
Removal attacks [4]	✗	✗	✗	✗	✗	✗	✓
Dynamic per-pattern authentication	✗	✓	✗	✗	✗	✗	✓
Support response encryption	✗	✗	✓	✓	✓	✗	✓
Support LOC/LOS	✗	✗	✓	✗	✓	✓	✓
Support test compression	✗	✗	✓	✓	✗	✓	✓
Total test-time overhead (cycles)	$p \times d$	$p \times d$	$4N$	None	None	$k * p$	None

k: key size in [15]; p: Pattern count; d: # of dummy flops; N: Round register size in [14].

REFERENCES

[1] J. Talukdar et al., "TaintLock: Preventing IP theft through lightweight dynamic scan encryption using taint bits," in *IEEE European Test Symp. (ETS)*, 2022, pp. 1–6.

[2] B. Tan *et al.*, "Benchmarking at the frontier of hardware security: lessons from logic locking," *arXiv preprint arXiv:2006.06806*, 2020.

[3] J. Talukdar et al., "Securing Heterogeneous 2.5 D ICs Against IP Theft through Dynamic Interposer Obfuscation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–2.

[4] A. Chakraborty *et al.*, "Keynote: A disquisition on logic locking," *IEEE Trans. CAD*, vol. 39, pp. 1952–1972, 2019.

[5] N. Limaye *et al.*, "Thwarting all logic locking attacks: Dishonest oracle with truly random logic locking," *IEEE Trans. CAD*, vol. 40, pp. 1740–1753, 2020.

[6] K. Z. Azar et al., "From cryptography to logic locking: A survey on the architecture evolution of secure scan chains," *IEEE Access*, 2021.

[7] N. Limaye et al., "DynUnlock: unlocking scan chains obfuscated using dynamic keys," in *IEEE DATE*, 2020, pp. 270–273.

[8] L. Alrahis *et al.*, "ScanSAT: Unlocking static and dynamic scan obfuscation," *IEEE Trans. on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1867–1882, 2019.

[9] J. Lee et al., "A low-cost solution for protecting IPs against scan-based side-channel attacks," in *IEEE VLSI Test Symp. (VTS)*, 2006, pp. 1–6.

[10] M. A. Razzaq et al., "SSTKR: Secure and testable scan design through test key randomization," in *IEEE Asian Test Symp.*, 2011, pp. 60–65.

[11] W. Zeng et al., "LORAX: Machine learning-based oracle reconstruction with minimal I/O patterns," in *IEEE ISVLSI*, 2021, pp. 126–131.

[12] D. Zhang et al., "Dynamically obfuscated scan for protecting IPs against scan-based attacks throughout supply chain," in *IEEE VTS*, 2017.

[13] R. Karmakar et al., "A scan obfuscation guided design-for-security approach for sequential circuits," *IEEE TCAS-II: Express Briefs*, 2019.

[14] M. Da Silva et al., "Preventing scan attacks on secure circuits through scan chain encryption," *IEEE Trans. CAD*, vol. 38, 2018.

[15] W. Wang *et al.*, "A secure scan architecture using parallel latch-based lock," *Integration*, 2023.

[16] H. Woo et al., "A Secure Scan Architecture Protecting Scan Test and Scan Dump Using Skew-Based Lock and Key," *IEEE Access*, 2021.

[17] M. Dworkin, "Block cipher modes of operation: The CCM mode for authentication and confidentiality," *NIST Special Publication*, 2003.

[18] K. J. Balakrishnan et al., "Relationship between entropy and test data compression," *IEEE Trans. CAD*, vol. 26, pp. 386–395, 2007.

[19] A. Sahai et al., "How to use indistinguishability obfuscation: deniable encryption, and more," in *ACM Symp. on Theory of Comp.*, 2014.

[20] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Workshop on the Theory and App. of Cryptographic Tech.*, 1993, pp. 386–397.

[21] J. Katz et al., *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

[22] J. Talukdar et al., "A BIST-based dynamic obfuscation scheme for resilience against removal and oracle-guided attacks," in *IEEE Intl. Test Conference*, 2021, pp. 170–179.

[23] S. Roshanisefat *et al.*, "RANE: An open-source formal de-obfuscation attack for reverse engineering of logic encrypted circuits," in *IEEE Great Lakes Symp. on VLSI*, 2021, pp. 221–228.

[24] B. Dutertre, "Yices 2.2," in *Intl. Conference on Computer Aided Verification*, 2014, pp. 737–744.

[25] L. De Moura *et al.*, "Z3: An efficient SMT solver," in *Intl. conf. on Tools and Algorithms for the Construction and Analysis of Systems*, 2008.

[26] R. Bruttomesso *et al.*, "The MATHSAT 4 SMT solver: Tool paper," in *20th Intl. Conference on Computer Aided Verification*, 2008, pp. 299–303.

[27] K. Shamsi et al., "KC2: Key-condition crunching for fast sequential circuit deobfuscation," in *IEEE DATE*, 2019, pp. 534–539.

[28] M. S. Rahman *et al.*, "Security assessment of dynamically obfuscated scan chain against oracle-guided attacks," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, pp. 1–27, 2021.

[29] S. Rai *et al.*, "Logic synthesis meets machine learning: Trading exactness for generalization," in *IEEE DATE*, 2021, pp. 1026–1031.

[30] A. Chaudhuri et al., "Fault-criticality assessment for ai accelerators using graph convolutional networks," in *IEEE DATE*, 2021.

[31] ——, "Functional criticality analysis of structural faults in AI accelerators," *IEEE Trans. CAD*, vol. 41, pp. 5657–5670, 2022.

[32] J. Rajski et al., "Embedded deterministic test," *IEEE Trans. CAD*, vol. 23, pp. 776–792, 2004.

[33] E. Valea et al., "Stream vs block ciphers for scan encryption," *Microelectronics Journal*, vol. 86, pp. 65–76, 2019.

[34] A. Bogdanov et al., "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems-CHES*, 2007, pp. 450–466.

Jonti Talukdar received the B. Tech. degree in ECE from Nirma University, Ahmedabad, India, in 2018, and M.S.and Ph.D. degrees in ECE from Duke University, Durham, in 2020 and 2024, respectively. His research interests lie at the intersection of hardware security, test, and applied machine learning for silicon security, health, and lifecycle management. His Ph.D. thesis is on IP security for 2.5D/3D HI Systems. He currently works as a senior DFX engineer at NVIDIA in Santa Clara, CA.

Arjun Chaudhuri received the B.Tech. degree in ECE from IIT Kharagpur, Kharagpur, India, in 2017. He received both the M.S. and Ph.D. degrees in ECE from Duke University, Durham, NC, in 2022. He currently works as a senior DFX Engineer at NVIDIA in Santa Clara, CA and also holds an appointment as an adjunct faculty at Arizona State University.

Eduardo Ortega Eduardo Ortega received the B.A./B.S. degrees in Integrated Engineering from the University of San Diego (2020). He is a Fulton Fellow of the Ira A. Fulton Schools of Engineering, School of Electrical, Computer, and Energy Engineering at Arizona State University, Tempe, AZ, where he is also currently pursuing the Ph.D. degree.

Krishnendu Chakrabarty (Fellow, IEEE) received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and M.S.E. and Ph.D. degrees from University of Michigan, Ann Arbor, in 1992 and 1995, respectively. He is now the Fulton Professor of Microelectronics in the School of Electrical, Computer and Energy Engineering at Arizona State University (ASU). He is also the Director of the ASU Center on Semiconductor Microelectronics and CTO of the Department of Defense Microelectronics Commons Southwest Advanced Prototyping (SWAP) Hub. Prof. Chakrabarty's current research projects include: design-for-testability of 2.5D/3D integrated circuits and heterogeneous integration; hardware security; AI accelerators; microfluidic biochips; AI for healthcare. He is a Fellow of ACM, IEEE, and AAAS, and a Golden Core Member of the IEEE Computer Society.