Increased Output Corruption and Structural Attack Resilience for SAT Attack Secure Logic Locking

Kyle Juretus, *Student Member, IEEE*, and Ioannis Savidis, *Senior Member, IEEE*

***Abstract*—Current out-of-cone logic locking methodologies provide resilience against the satisfiability (SAT) attack with minimal corruption of the outputs when comparing an activated and locked integrated circuit (IC). In addition, the structure of the modifications to the original logic leaks functional information of the circuit, which allows an adversary to determine the correct key. A novel logic locking methodology, CORruption adaptable logic locking (CORALL), is introduced in this article that provides increased security against the SAT attack for modified logic cones that require a large corruption of the primary outputs of the circuit, where the corruption is quantified by comparing between an activated and locked state of the IC. In addition, the modifications to the logic cone utilized by CORALL provide increased resilience against structural attacks. The CORALL architecture increases the number of iterations required to successfully execute a SAT attack for a flip function with 20 inputs by 34.41× over SFLL-HD n/4 and 82.36× over SFLL-Flex. In addition, a protected-cube selection process based on iterative cofactors is introduced, which provides varying logical functions of the perturb unit and maps portions of the logic of the perturb unit into the look-up tables (LUTs) of the CORALL architecture. The variation in the logical functions implemented by the perturb unit and the mapped functionality of the perturb unit into a LUT provide resistance to all current structural attacks on out-of-cone logic locking techniques.**

***Index Terms*—Hardware security, logic locking, SAT attack, stripped functionality logic locking.**

I. Introduction

With a majority of the focus of integrated circuit (IC) design driven by optimizations in power, performance, and area (PPA), the security of the IC has been largely overlooked. Attacks, such as Spectre [1] and Meltdown [2], which resulted from vulnerabilities to the architecture of the IC, demonstrate that IC security must be considered as an optimization criteria along with PPA when designing next-generation circuits.

Without proper IC security, threats including denial of service, theft of information, and/or corruption of circuit functionality are all possible [3]. In addition, threats to the security of an IC are compounded by the increasing reliance on third-party intellectual property (IP), which has expanded in use to satisfy time-to-market constraints and to reduce monetary cost [4]. Furthermore, third-parties have become more prevalent in IC manufacturing due to the multibillion-dollar investment required to implement an advanced fabrication process [5]. The threat landscape of untrusted third parties, therefore, includes third-party IP vendors, fabrication facilities, test facilities, and end-users.

One area of research that explores methods to mitigate the security threats to an IC posed by untrusted third-parties is logic locking/obfuscation. Logic locking introduces a key dependency that is used to logically configure an IC, which prevents unauthorized third-parties from having access to the complete netlist of the circuit.

A variety of logic locking techniques exist. Early techniques implemented in-cone logic locking, which insert additional gates within the logic cone of the circuit to generate a functional dependency based on an applied key [6]–[9]. In addition, a variety of techniques exist that alter the functionality of a finite state machine (FSM) within an IC [10]–[16], which are termed sequential logic locking methods.

One of the major threat vectors to in-cone logic locking techniques is oracle guided attacks, where an adversary uses responses from an activated IC to iteratively determine the key. The most prominent logic locking oracle guided attack introduced in [17], is based on a satisfiability (SAT) solver. The SAT attack utilizes a miter circuit with two copies of the locked circuit each supplied with a different key to generate a distinguishing input pattern (DIP) that is then applied to an active oracle IC. The process iteratively continues until no additional DIPs are generated.

Part of the reason SAT-based attacks execute efficiently on in-cone logic locking methodologies is due to the masking of certain key gates when applying generated DIPs [18]. In order to increase the resilience of logic locked circuits to the SAT attack, out-of-cone logic locking techniques strategically corrupt the outputs of an IC when an incorrect key is applied [19]–[22]. However, additional attacks have been introduced that reduce the security of out-of-cone techniques [23]–[26].

A majority of the vulnerabilities of out-of-cone techniques do not apply to out-of-cone architectures that modify the original logic cone, such as stripped-functionality logic locking (SFLL) [21]. However, the structure of the circuit that modifies the function of the logic cone is vulnerable to identification, which allows an adversary to determine the original logic cone [26]–[28]. In addition, to provide resilience to the SAT attack, SFLL requires that the corruption of the outputs of the circuit is limited when comparing the modified logic cone to the original cone.

Manuscript received October 7, 2019; revised February 16, 2020; accepted March 28, 2020. Date of publication April 17, 2020; date of current version January 11, 2021. This work was supported in part by the Air Force Office of Scientific Research, in part by the National Defense Science and Engineering Graduate Fellowship, 32 CFR 168a, in part by the Drexel Ventures Innovation Fund, and in part by National Science Foundation under Grant CNS-1751032. This article was recommended by Associate Editor S. Ghosh. *(Corresponding author: Kyle Juretus.)*

The authors are with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104 USA (e-mail: kjj39@drexel.edu; is338@drexel.edu).

Digital Object Identifier 10.1109/TCAD.2020.2988629

0278-0070 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: SAT Attack Algorithm [17]

Input: C and eval
Output: $\overrightarrow{K}_C$
$i := 1$;
$F_1 = C(\overrightarrow{X}, \overrightarrow{K_1}, \overrightarrow{Y_1}) \wedge C(\overrightarrow{X}, \overrightarrow{K_2}, \overrightarrow{Y_2})$;
while $sat[F_i \wedge (\overrightarrow{Y_1} \neq \overrightarrow{Y_2})]$ **do**
  $\overrightarrow{X_i^d} := sat\_assignment_{\overrightarrow{X}}[F_i \wedge (\overrightarrow{Y_1} \neq \overrightarrow{Y_2})]$;
  $\overrightarrow{Y_i^d} := eval(\overrightarrow{X_i^d})$;
  $F_{i+1} := F_i \wedge C(\overrightarrow{X_i^d}, \overrightarrow{K_1}, \overrightarrow{Y_i^d}) \wedge C(\overrightarrow{X_i^d}, \overrightarrow{K_2}, \overrightarrow{Y_i^d})$;
  $i := i + 1$;
end
$\overrightarrow{K_C} := sat\_assignment_{\overrightarrow{K_1}}(F_i)$

A novel logic locking methodology, CORuption adaptable logic locking (CORALL), is introduced in this article that results in increased resilience against the SAT attack when significant output corruption is needed between the modified and original logic cones. In addition, a protected cube selection algorithm based on iterative cofactors is proposed to generate varying topologies of the logical circuit that modifies the functional behavior of the original logic cone. The developed protected cube selection algorithm also allows for the storage of a portion of the functionality of the perturb unit, which implements the modifications to the logic cone, in the look-up tables (LUTs) of the CORALL flip function circuitry. Storing a portion of the perturb unit in the LUTs prevents an adversary from extracting the entire logic of the perturb unit, even in the worst case when the logic of the perturb unit is structurally separate from the original logic cone. With the addition of 1) varying flip-function topologies, 2) better logical mixing of the perturb unit with the original cone, and 3) the ability to store a portion of the flip-function in LUTs, the CORALL architecture is secure against all currently developed structural attacks.

This article is organized as follows. Oracle guided attacks are discussed in Section II. An overview of SAT resistant out-of-cone logic locking methodologies is provided in Section III. A description and a comparison of SFLL-HD, SFLL-Flex, and CORALL is provided in Section IV. An overview of structural-based attacks is provided in Section V. The CORALL protected cube selection algorithm based on iterative cofactors is described in Section V-A. An analysis of the SAT attack resilience, structural attack resilience, corruption of the modified logic cone, and overhead in PPA is also provided in Section V. Concluding remarks are provided in Section VI.

II. SAT Attack Overview

The most prominent and efficient oracle guided attack against a logic obfuscated IC makes use of a SAT solver to unlock the circuit [17]. The pseudocode of the SAT attack is provided as Algorithm 1. The notation utilized to describe the combinational logic space is $C(\overrightarrow{X}, \overrightarrow{K}, \overrightarrow{Y}) \subseteq \mathbb{B}^{M+L+N}$, where $\mathbb{B}$ is the binary domain of $\{0, 1\}$, $\overrightarrow{X} \in \mathbb{B}^M$ represents the M primary inputs, $\overrightarrow{Y} \in \mathbb{B}^N$ represents the N primary outputs, and $\overrightarrow{K} \in \mathbb{B}^L$ represents the L key inputs.

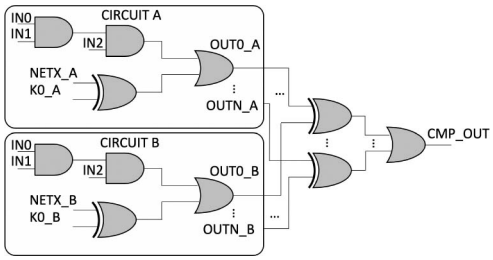


Fig. 1. Miter circuit with replicated A and B versions of the reverse engineered netlist that includes obfuscated gates. Corresponding outputs between the A and B versions of the circuit are XORed. Each XORed signal is then passed to an OR gate to check for any differentiating output.

The primary steps of the algorithm include 1) generation of a DIP, 2) evaluation of the DIP on an oracle IC, and 3) adding the constraints generated from the oracle response to the SAT model of the circuit. The first step, the generation of a DIP, is completed through the use of a miter circuit of the locked netlist obtained by reverse engineering the IC. A miter circuit logically represents two copies of a locked reverse engineered circuit, with each copy supplied a distinct key. The outputs of the two copies of the circuit are XORed to check for equivalence, and the outputs of the XOR gates are connected to an OR gate, as shown in Fig. 1. The miter circuit is efficiently converted into conjunctive normal form (CNF) by applying the Tseytin transformation [29]. As an example, the Tseytin transformation of an AND gate with inputs A and B and output C is provided as (1). The CNF representation of the circuit is provided to a SAT solver to generate the DIPs

$$\left(\bar{A} \vee \bar{B} \vee C\right) \wedge \left(A \vee \bar{C}\right) \wedge \left(B \vee \bar{C}\right). \tag{1}$$

The generated DIPs are then provided to an activated IC as inputs, which yields the correct outputs of the circuit for the given input pattern. The input and output pairs that are generated are applied to the CNF equivalent representation of the locked circuit netlist, leaving a set of logical constraints that are appended to the original SAT formulation. The process of DIP generation, oracle IC evaluation, and adding input/output constraints to the SAT model of the circuit is repeated until no further generation of DIPs is possible. The SAT attack model is extremely efficient when applied to in-cone logic locked ISCAS'85 [30] and MCNC [31] benchmark circuits, decrypting a majority of the evaluated circuits within 10 hours [17].

III. SAT Attack Resistant Architectures

Reducing the efficiency of SAT-based attacks requires limiting the amount of information leaked on the correct key for each generated DIP. From the perspective of an adversary, the worst-case scenario occurs when only a single key is eliminated for each DIP, which results in a SAT attack efficiency equivalent to that of a brute force attack. In order to control both the output corruption and the elimination of keys for each DIP, the techniques described in [19], [20], [21], and [22] add additional circuitry that produces output corruption that is independent of the original structure of the logic cone.

The typical circuit resembles a structure similar to that shown in Fig. 2, where the flip function ensures that corruption at the *OUT* net is strategically generated to prevent a single DIP from significantly pruning the key space. The flip

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

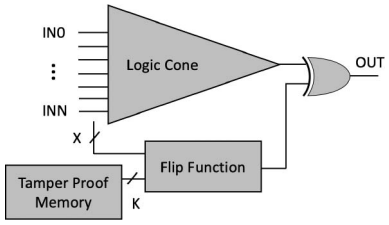


Fig. 2. Generalized circuit topology to defend against the SAT attack utilized in [19], [20], [21], and [22]. Signals represented by X, which are extracted from the inputs to the logical cone or the internal signals of the logical cone, are combined with key signals K to generate a function that controls the assertion of the flip function.

functions utilized across the developed SAT resistant methodologies include 1) generating complementary logic blocks that produce a logic 1 when an incorrect key is applied to the circuit for a subset of the inputs and keys [19], 2) utilizing a mask function that corrupts circuit outputs when an incorrect key and input pattern is applied [20], and 3) Hamming distance and LUT-based circuitry that corrects the corrupted functional output of a logic cone based on the applied input [21]. A benefit of implementing out-of-cone techniques that control the corruption of the outputs of a circuit [19]–[22] is a more reliable prediction of the resilience of the circuit to the SAT attack as the original logic structure does not impact the resilience, unlike for in-cone logic locking methodologies.

However, attacks to out-of-cone logic locking methodologies have been developed that do not rely on the execution of the SAT attack to ascertain the original functionality of the circuit. The threats include removal attacks [25], approximation attacks [23], [24], and structural attacks [26]–[28]. Removal attacks [25] are applicable when the added flip function is easily determined within the IC and is removed. As the output corruption of the techniques described in [19] and [20] is only dependent on the flip function, once removed, the original functionality of the circuit is exposed.

In order to modify the functionality of the original cone, in-cone logic locking techniques are implemented as a means to increase the resilience of the out-of-cone techniques to removal attacks [20]. However, the added in-cone logic locking techniques are also vulnerable to attack due to the disparity in the corruption of the outputs produced by the in-cone techniques and the SAT resistant flip function. The SAT resistant flip function corrupts a very small number of the circuit outputs, whereas in-cone logic locking achieves close to a 50% Hamming distance when comparing the difference in the values of the outputs of the locked circuit to the outputs of the original unlocked circuit [32]. The disparity in output corruption allows for attacks such as described in [23] and [24] that ignore the limited number of corrupted outputs produced by the flip function while determining the in-cone logic locking key, which results in a minimal degradation in the efficiency of the SAT attack due to the implemented flip function. The algorithm described in [23] applies random I/O patterns after a set number of iterations of the SAT solver and provides an estimate of the error when returning an approximate functional netlist of the IC. The method developed in [24] forces DIPs to eliminate at least two keys, which bypasses the artificial corruption of the outputs induced by the flip function and targets the key of the in-cone logic locking elements.

In order to prevent the removal of the flip function by the attacks described in [23] and [24], the SFLL method proposed in [21] applies modifications to the original logic cone of the circuit. The modified logic cubes (minterms) then serve as the protected input cubes, which are corrected by a flip function. The modification of the logic cone prevents removal, bypass, and sensitization attacks as the adversary is left with the modified logic cone if the circuit implementing the flip function is removed [21]. The technique in [33] generates a modified logic cone with a flip function implemented as a diversified logic tree, which produces a varying level of output corruption by replacing AND gates within the flip function AND-tree with gates of other logical functions. The resulting key space of the diversified logic tree is similar to SFLL-HD as the number of key inputs is limited to the number of circuit inputs. Additional details regarding the architectures of the SFLL-HD and SFLL-Flex flip functions are provided in Section IV. However, two concerns regarding the security of modified logic cone-based techniques remain: 1) the resilience to SAT attack is dependent on low levels of corrupted outputs when comparing an activated IC and a locked IC that is provided an incorrect key and 2) the techniques are vulnerable to structural attacks that target the architecture of the perturb unit within the modified logic cone [26]–[28].

IV. SAT Attack Resistant Architectures With Increased Corruption

Limitations in the maximum achievable output corruption of current techniques that modify the logic cone [21], [33], [34] are described in this section. An analysis of SFLL-HD and SFLL-Flex [21] is performed as SFLL-HD shares a similar key space to [33] and an SFLL-Flex-based flip function is utilized in [34]. The proposed CORALL architecture is then introduced, which provides resilience against the SAT attack while generating a significant error rate (ER), as defined in Section V-A, in the primary outputs when comparing the modified and original logic cone of the circuit.

A. SFLL-HD

The resilience of SFLL-HD is dependent on the number of protected cubes c, the number of key bits k, and the number of queries q the adversary performs on the oracle IC. The provided security of SFLL-HD is estimated through calculation of (2) [21]. As c increases, the probability an adversary finds a protected input improves, which results in a decrease in the average number of queries needed to determine the key bits. When the Hamming distance parameter is small, which results in a small c, the number of iterations required for an adversary to successfully execute a SAT attack, for a sufficient key size k, becomes large on average.

$$\frac{c}{2^k} + \frac{c}{2^k - 1} \cdots + \frac{c}{2^k - q} \approx \frac{c * q}{2^k} \tag{2}$$

The strongest resilience against the SAT attack occurs when the Hamming distance parameter is set to 0, as the number of protected cubes is 1. Therefore, the number of iterations to determine the protected input cube is estimated as approximately uniform from calculation of (2). In addition, nonprotected input cubes only eliminate a single key for each DIP, which results in SFLL-HD^0 providing a high resilience

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

to the average SAT attack case. However, the error rate of the SFLL-HD modified logic cone as compared to the original circuit is only $1/2^n$, where n is the total number of inputs to the logic cone. The very low level of output corruption raises concern that an adversary essentially has access to a working IC, especially if the circuit is tolerant to rare errors.

Increasing the value of the Hamming distance parameter of SFLL-HD results in an increase in the output corruption of the circuit as the number of protected cubes is expanded to k choose h combinations $\binom{k}{h}$, where h is the chosen Hamming distance and k is the number of key bits. Instead of producing a single incorrect output, each nonprotected cube now generates an incorrect output for any key that is a Hamming distance h away from the applied input vector. However, the protected cubes now produce correct responses for $\binom{k}{h}$ combinations of keys, instead of the single correct key present with SFLL-HD^0. The total error rate of the circuit is now given by (3), where $\binom{k}{h}/2^k$ defines the corruption probability of a nonprotected cube and $(2^k - \binom{k}{h})/2^k$ denotes the corruption probability of a protected cube.

$$\mathrm{ER}_{\mathrm{total}} = \frac{\binom{k}{h}}{2^k} + \frac{2^k - \binom{k}{h}}{2^k} \tag{3}$$

While the corruption increases for larger HD values, DIPs generated by the SAT attack for nonprotected cubes now eliminate a greater number of keys, which results in greater efficiency when executing a SAT attack. The primary cause of the degradation in the resilience against the SAT attack is that the key space is limited to 2^n combinations, where n is the number of inputs to the circuit. Increasing the error rate for a key space of the same size, therefore, results in the elimination of more keys per DIP.

B. SFLL-Flex

The SFLL-Flex [21] flip circuit differs from SFLL-HD as the flip function of SFLL-Flex is implemented as a LUT that inverts the logical value of protected input cubes as opposed to the Hamming distance comparator utilized in SFLL-HD. The estimated security (ES) of SFLL-Flex is given by (4) [35]. Similar to SFLL-HD, a large k provides the greatest resilience against the SAT attack on average.

$$\mathrm{ES}_{\mathrm{avg}} = 2^k * \frac{c}{c+1} \tag{4}$$

To increase the output corruption provided by the SFLL-Flex technique, the number of protected cubes c is increased or the number of key inputs k protecting each cube is reduced. While increasing the value of c does not negatively effect the average SAT resilience, the number of cubes added to produce a significant error rate in the outputs of the modified logic cone as compared to the original cone results in a significant increase in area and power. Reducing the number of key bits increases the output corruption of the circuit without resulting in a large cost in area and power. However, as the length of the key is reduced, the number of possible key combinations and, therefore, $\mathrm{ES}_{\mathrm{avg}}$ also significantly decrease. For example, if 50% of the outputs of a cone are to be corrupted, then a single input serves as a dependency for the protected cube. An adversary is, therefore, able to easily brute force the value of the protected cube as the cube only has a single input dependency of two possible values.

A	B	K0	K1	K2	K3
0	0	X	✓	✓	✓
0	1	X	✓	X	X
1	0	✓	✓	X	✓
1	1	✓	✓	✓	X

(a)

A	B	K0	K1	K2	K3
0	0	X	✓	✓	X
0	1	X	✓	✓	X
1	0	X	✓	✓	X
1	1	X	✓	✓	X

(b)

A	B	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15
0	0	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	X	X	X	X
0	1	✓	✓	✓	✓	X	X	X	X	✓	✓	✓	✓	X	X	X	X
1	0	✓	✓	X	X	✓	✓	X	X	✓	✓	X	X	✓	✓	X	X
1	1	X	✓	X	✓	X	✓	X	✓	X	✓	X	✓	X	✓	X	✓

(c)

A	B	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9
0	0	✓	✓	✓	✓	✓	X	X	✓	X	X
0	1	✓	✓	✓	✓	X	✓	X	X	✓	X
1	0	✓	✓	X	X	✓	✓	✓	✓	X	X
1	1	X	✓	X	✓	X	X	X	✓	X	✓

(d)

Fig. 3. Truth tables of a two input AND gate locked with (a) SFLL-HD^0 with a correct key of K1, (b) SFLL-HD^1 with a correct key of K1 and K2, (c) LUT-based correction circuit with a correct key of K1, and (d) CORALL $n2l1c1$ with a correct key of $K1$.

C. CORALL

Current SAT resistant techniques that are based on logic cone modifications provide limited output corruption. The developed CORALL architecture addresses limitations to current techniques by utilizing an expanded key space that allows for increased output corruption while providing high SAT attack resilience. The expansion is denoted as an increase in the number of unique columns of the truth table representing the key space, which implies more functions for an oracle guided attack to constrain. For example, consider the output corruption tables shown in Fig. 3 of the flip function circuitry of a two-input AND gate implemented with SFLL-HD^0, SFLL-HD^1, and a LUT. The modification of the logic cone when implementing SFLL-HD^0 on a two-input AND flips the output of the circuit when $A = 0$ and $B = 1$, which results in a modified cone with an output corruption probability of 25%. In comparison, 50% of the cubes, $AB = 00$ and $AB = 11$, are corrupted when implementing the AND with SFLL-HD^1. However, as discussed in Section IV-A, less DIPs are required to execute a successful SAT attack on the obfuscated AND gate, with an average of two DIPs required for SFLL-HD^0 and only one DIP for SFLL-HD^1. As opposed to the SFLL-HD implementations, the LUT flip function shown in Fig. 3(c) implements up to 2^4 possible functions. Since there are 2^4 possible functions, four iterations of the SAT attack are required to decrypt the key independent of the corruption applied to the modified logic cone. In general, a LUT with n select lines generates 2^{2^n} unique columns. Since each input pattern applied to the LUT generates an equally balanced logic one and logic zero in the key space, as shown in Fig. 3(c), the key space is cut in half each iteration. The 2^{2^n} unique columns, therefore, force an oracle guided attack to apply 2^n input patterns to determine the correct key.

While a LUT allows for the maximum amount of unique columns, and therefore, the largest number of functions for an adversary to search, the use of a LUT as the flip-function is limited by the significant overhead in PPA required to

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

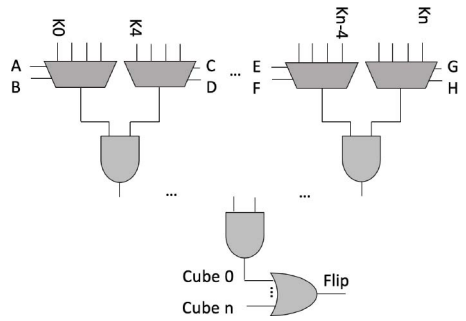


Fig. 4. Implementation of the flip function circuit of CORALL. The first layer consists of l-input LUTs, where l is two for the example shown, with the inputs to the modified cone provided to the LUTs. The outputs of the LUTs are inputs to an AND tree, which forms a single protected cube. Multiple protected cubes are then provided as inputs to an OR gate, which generates the flip signal to correct any errors in the modified logic cone.

implement the circuit. The objective of CORALL is to generate a large number of unique columns in the key space without resulting in a large overhead in PPA.

The CORALL architecture is described as CORALL(n, l, c), where n defines a subset of the circuit inputs, l is the number of LUT select lines, and c is the number of protected cubes. The circuit structure when l equals 2 is shown in Fig. 4, where each logical input of the circuit is connected to a select line of a two input LUT that requires four key bits and the output of each LUT is then connected to an AND tree. The structure is easily modified to account for multiple protected cubes, where each output of a protected cube is connected to an OR gate, as shown in Fig. 4. In addition, the size of the LUTs is adaptable. For example, setting l to 4 is possible, which combines inputs A, B, C, and D into a single LUT instead of utilizing two LUTs to group inputs A and B and inputs C and D when l equals 2. Rather than requiring an exponential increase in the number of keys, as needed for a standard LUT topology, the CORALL architecture only requires a linear increase in the key size. The number of keys required for circuits with 1 to 12 inputs is provided in Fig. 5, where a linear increase in the number of keys for an l of 2, 3, 4, 6, and 8 inputs is shown for the CORALL architecture. For example, securing a circuit with inputs A to H ($n = 8$) requires 2^8 key inputs for a standard LUT topology as compared to 16 key bit inputs for CORALL $n8l2c1$, which significantly reduces the area and power overhead when l is less than the total number of inputs to a given logic cone. The number of keys required to implement CORALL is given by (5), where num_lut $= \lceil n/l \rceil$.

$$c * \sum_{i=1}^{\text{num\_lut}} 2^{l_i} \tag{5}$$

1) SAT Attack Resilience of CORALL: When the corruption of the modified logic cone equals $1/2^n$, the resilience of CORALL $nl1c1$ to a SAT attack is nearly identical to SFLL-HD^0 as the functionality of CORALL inherently contains the point flip function produced by SFLL-HD^0. The only difference is that applying CORALL results in an increase in the minimum number of iterations required by the SAT attack to return the correct key since guessing the protected cube does not eliminate all but one key value as is the case with SFLL-HD^0. For example, consider the resulting key space when securing the input pattern $AB = 11$ with CORALL $n2l1c1$,

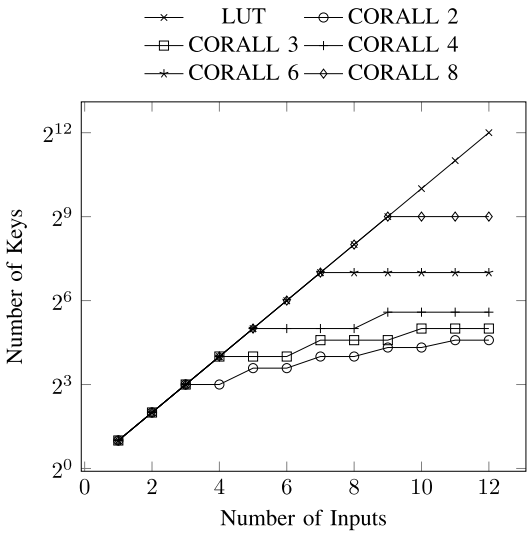


Fig. 5. Number of keys required for a standard LUT-based logic locking topology and the CORALL methodology as the number of primary inputs to the circuit is increased from 1 to 12. The CORALL architecture is evaluated for LUT sizes of 2, 3, 4, 6, and 8 inputs.

which is shown in Fig. 3(d). Even when the DIP $AB = 11$ is selected, multiple correct functions still exist. In the case of CORALL $n2l1c1$, three DIPs are required to determine the correct functionality of the circuit as the technique generates 10 out of a possible 16 unique column values. Note that when the correct constraints (check marks) and incorrect constraints (X's), as shown in Fig. 3(d), are roughly evenly distributed, the number of iterations of the SAT attack is approximated by $\log_2(u)$, where u is the number of unique columns. For the approximation of $\log_2(u)$ to hold, the remaining valid columns for each iteration must also have an approximately even distribution of correct and incorrect output possibilities. If, for example, a DIP of $AB = 10$ was chosen after $AB = 00$ in Fig. 3(d), there are still four correct and two incorrect possible functions remaining, which implies that the adversary is only able to eliminate $^2/_6$ of the possible key values.

As the amount of output corruption is increased, which is quantified by comparing the outputs of the original and modified logic cones, the number of unique columns in the truth table of the key space dictates the extent to which the CORALL topology is resilient to an oracle guided attack. An increased number of unique columns provides greater resilience to the SAT attack as an added constraint to the SAT solver when a protected cube is chosen as a DIP is the logical AND of the key bits exposed by the LUTs. For example, the circuit shown in Fig. 4 with $c = 1$ results in the constraint $K0 \cdot K4 \cdot (Kn - 4) \cdot Kn$ when the DIP input of A to H equals 00001111 produces an output of 1 from the flip function. The extreme case occurs when the modified logic cone is an inverse function of the original logic cone, which results in a constant one at the output of the flip function. In the case where the modified logic cone is completely corrupted and $c = 1$, which implies a single protected cube, the CORALL architecture requires a minimum number of SAT iterations of 2^l, where l is the number of LUT select lines. The minimum number of iterations is a result of revealing a single key bit of each LUT for each determined protected minterm pattern.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

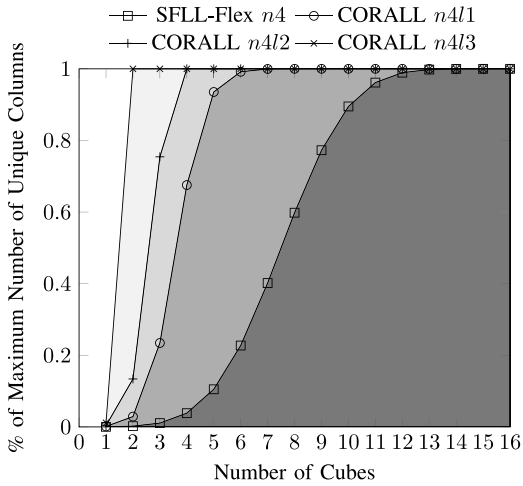


Fig. 6. Number of unique columns of the key space truth table as the number of protected cubes c is increased for CORALL and SFLL-Flex when $n = 4$. The l size of CORALL is varied from 1 to 3.

Increasing l results in a greater minimum number of iterations required to fully execute an oracle guided attack. The greater number of unique columns added to the truth table of the key space by increasing l, therefore, provides an improved resilience to oracle guided attacks, with the number of unique columns when $c = 1$ given by $(2^{2^l} - 1)^{n/l} + 1$.

The number of unique columns is also increased by providing a greater number of cubes c to the CORALL circuitry. Whereas the number of unique columns for SFLL-Flex is given by $\sum_{i=1}^{c} \binom{n}{i}$, CORALL generates more unique columns as multiple minterms are flipped per protected cube. The number of unique columns for CORALL and SFLL-Flex when the number of inputs to the flip function n equals 4 is provided in Fig. 6. The results shown in Fig. 6 indicate that CORALL provides a much faster increase in the number of unique columns than SFLL-Flex, generating SAT attack resilience with increased modified logic cone error rates without requiring the large overhead of additional cubes needed with SFLL-Flex. Using the notation defined in [36], the CORALL circuit is able to achieve best-possible approximation-resilient locking (BPARL) with regard to oracle guided attacks as the hypothesis space, which equals the number of unique columns, exponentially increases until the space saturates at 2^{2^n} columns. As the required level of output corruption for a given modified logic cone decreases, the value of both the c and l parameters is reduced as the SAT attack resilience of SFLL given by (2) applies.

2) Simulation of Benchmark Circuits: The SFLL-HD, SFLL-Flex, and the proposed CORALL architectures are compared for 5, 10, 15, and 20 input flip functions implemented for the b17 and b18 ITC'99 [37] benchmark circuits and the s38584 and s35932 ISCAS'89 [38] benchmark circuits, with results provided in Fig. 7. The CORALL architecture is evaluated for 2 and 4 protected input cubes c and for 2 and 4 inputs to the LUT l. The number of constrained inputs for the analysis of CORALL is 1, which implies that $n-1$ inputs are free, where n is the total number of inputs to the flip function. Constraining only a single input per cube results in modified logic cones that produce a large amount of output corruption as compared to an activated IC.

TABLE I
OUTPUT CORRUPTION FROM THE MODIFIED LOGIC CONES OF THE B17, B18, S38584, AND S35932 BENCHMARK CIRCUITS. THE SECURED CIRCUITS INCLUDE A 20 INPUT FLIP FUNCTION AND ARE EVALUATED FOR 1000 RANDOM INPUT PATTERNS, WITH EACH RANDOM INPUT PATTERN EVALUATED FOR 100 RANDOM KEY VALUES.

Methodology	**Benchmark**			
	b17	**b18**	**s38584**	**s35932**
CORALL $c4l4f19$	83.60%	82.90%	80.80%	84.70%
CORALL $c2l2f19$	76.30%	73.00%	73.60%	74.30%
SFLL HD $n/3$	6.80%	7.90%	6.70%	6.90%
SFLL HD $n/2$	29.60%	31.60%	28.60%	30.40%
SFLL HD $n/4$	3.20%	3.30%	2.50%	3.30%
SFLL Flex $c2n5$	10.10%	10.60%	13.20%	11.70%
SFLL Flex $c4n5$	21.70%	20.20%	22.00%	20.70%

The analysis of SFLL-HD with results as shown in Fig. 7 is completed for Hamming distances of $n/4$, $n/3$, and $n/2$ to ensure significant output corruption within the modified logic cone. The analysis of SFLL-Flex for all trials is completed with 2 and 4 protected cubes c and 1 and 5 inputs n for each cube.

The results shown in Fig. 7 confirm that CORALL significantly outperforms SFLL-HD and SFLL-Flex in both the number of iterations of the SAT solver and the CPU execution time required to complete the SAT attack. When implementing a 20 input flip function, CORALL *c4l4*, on average, outperforms SFLL-HD $n/4$ by 34.41× and SFLL-Flex by 82.36× in the number of iterations required to execute a successful SAT attack. The execution time of the SAT attack on circuits secured by CORALL is, on average, 111.96× that of SFLL-HD and 430.05× that of SFLL-Flex for a flip function with 20 inputs.

In addition to increasing the number of SAT attack iterations, CORALL also provides greater corruption of outputs as compared to SFLL-HD and SFLL-Flex. The estimated error rate (output corruption) for the CORALL, SFLL-HD, and SFLL-Flex techniques is analyzed for a 20 input circuit implementation of the flip function with results listed in Table I. The analysis is performed by comparing the outputs of an activated and locked IC for 1000 random input patterns. For each random input pattern, 100 random key values are applied to the locked IC. The results listed in Table I indicate that the CORALL architecture corrupts a minimum of 73% of the total input-output pairs. The maximum output corruption observed for SFLL-HD and SFLL-Flex is, respectively, 31.60% and 21.70%. CORALL, therefore, significantly improves the resilience of a circuit to a SAT attack as compared to both SFLL-HD and SFLL-Flex for cases where a large corruption of the primary outputs is required.

An additional consideration when characterizing the resilience of a circuit to the SAT attack is the variation in the number of iterations required to execute the attack, which is analyzed on the c1908 ISCAS'85 benchmark circuit. The c1908 circuit is evaluated for 500 random netlist orders after applying the SFLL-HD and CORALL techniques with a 10 input circuit of the flip function. Results of the analysis are provided in Fig. 8. Varying the order of the netlist results in variation in the number of iterations required to successfully execute the SAT attack [18].

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

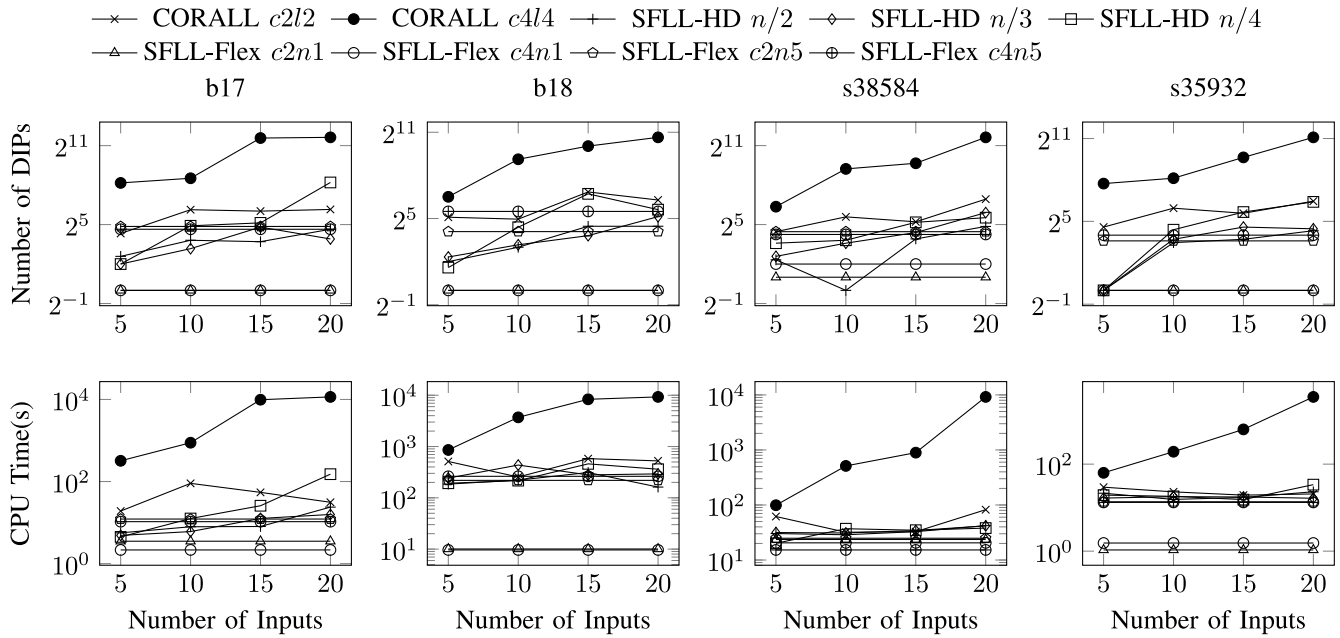


Fig. 7. Comparison of the SFLL-HD, SFLL-Flex, and proposed CORALL out-of-cone logic locking techniques when the modified logic cone requires a high corruption of outputs as compared to an activated IC. The techniques are evaluated for circuit implementations of the flip function with 5, 10, 15, and 20 inputs. The number of cubes *c* and LUT-size *l* are varied for CORALL. The SFLL-HD architecture is evaluated for varying Hamming distances computed based on the number of inputs *n*. The SFLL-Flex architecture is analyzed for a varying number of cubes *c* and number of inputs per cube *n*.

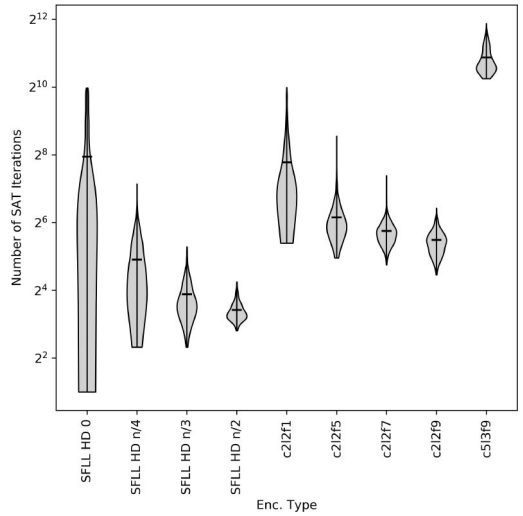


Fig. 8. Number of SAT attack iterations required to decrypt the ISCAS'85 c1908 benchmark circuit when the circuit is locked with a 10 input flip function using the SFLL-HD and CORALL methodologies. Each secured circuit is evaluated for 500 netlist orders [18] to characterize the variation in the number of iterations. The SFLL-HD technique is evaluated for Hamming distances of 0, *n*/4, *n*/3, and *n*/2, where *n* is the number of inputs to the circuit of the flip function. The CORALL methodology is evaluated for *c2l1f*1, *c2l2f*5, *c2l2f*7, *c2l2f*9, and *c5l3f*9, where *c* represents the number of cubes, *l* the LUT size, and *f* the number of free inputs.

SFLL-HD^0 and CORALL *c2l2f*1 are also characterized, which include a single free input *f* of 1 as shown in Fig. 8 and represent the case where the provided output corruption is limited for each methodology. Note that the free inputs *f* are a subset of *n* and represent the inputs a protected cube is independent of. The results shown in Fig. 8 indicate that the mean number of iterations produced by SFLL-HD^0 is slightly greater than CORALL *c2l2f*1. However, the minimum number of iterations for SFLL-HD^0 is much lower, and the distribution is uniform across much of the range of the number of iterations required to complete the SAT attack. The uniform nature of SFLL-HD^0 matches the expected numerical results given by (2). A much larger minimum number of iterations is required for CORALL *c2l2f*1, which matches the results from the analysis provided in Section IV-C1. CORALL, therefore, provides nearly equivalent resilience to the SAT attack as that of SFLL-HD^0 for cases that require a low output corruption while also requiring a much larger minimum number of iterations of the SAT attack.

As the Hamming distance parameter of SFLL is increased, the mean number of iterations required to execute the SAT attack is reduced significantly as indicated by the results shown in Fig. 8. However, the minimum number of iterations needed to successfully execute the SAT attack increases. In contrast, the number of iterations to complete the SAT attack decreases as the number of free inputs increase for CORALL. However, the mean and minimum number of iterations are greater for CORALL than SFLL-HD. Note that the resilience of CORALL to the SAT attack increases as the number of cubes *c* and the LUT size *l* increase. Increasing the number of cubes to 5 and the LUT size to 3 while 9 of 10 inputs are unconstrained to produce greater output corruption results in an increase in the number of iterations to over 2^{10}. The trade-off for increasing the number of cubes and LUT size is the additional overhead in power, performance, and area (PPA).

The overhead in PPA to implement the SFLL-HD and SFLL-Flex techniques with a flip function of 20 inputs is listed in Table II, while the characterization of the overheads for CORALL is listed in Table III. The listed data indicates that for the large ITC'99 (b17 and b18) and ISCAS'89 (s35932 and s38584) benchmark circuits, the implementations of all three techniques result in low overheads, with the PPA increasing

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

TABLE II
ANALYSIS OF THE PPA OVERHEAD OF IMPLEMENTING SFLL-HD AND SFLL-FLEX ON THE B17, B18, C432, C1908, S35932, AND S38584 BENCHMARK CIRCUITS. SFLL-FLEX IS EVALUATED FOR 2 AND 4 PROTECTED CUBES c AND A FLIP FUNCTION WITH 5 INPUTS ($n = 5$). ALL EVALUATED CIRCUITS ARE LIMITED TO A MAXIMUM OF 20 INPUTS TO THE FLIP FUNCTION.

Benchmark	Original			SFLL-HD			SFLL-Flex					
							c2n5			c4n5		
	Area (μm^2)	Power (mW)	Timing (ns)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)
b17	2.47E+05	102.81	23.73	1.15	5.15	5.65	-0.62	-1.28	2.44	-0.42	-1.54	2.44
b18	7.11E+05	402.93	27.67	0.43	1.89	4.26	-0.06	0.79	2.35	0.05	0.98	2.02
c432	1.71E+03	0.48	3.85	249.01	659.26	24.42	30.77	22.92	20.26	55.82	40.09	13.25
c1908	4.15E+03	2.12	4.64	95.83	145.82	8.41	10.44	10.41	3.88	22.60	17.22	-1.29
s35932	9.72E+04	38.04	3.04	4.47	11.18	114.47	0.54	1.01	-0.33	1.04	1.75	0.66
s38584	1.18E+05	40.56	5.22	4.25	2.11	24.71	0.93	-7.91	1.53	1.31	-7.76	5.94

TABLE III
ANALYSIS OF THE PPA OVERHEAD OF THE CORALL ARCHITECTURE FOR THE B17, B18, C432, C1908, S35932, AND S38584 BENCHMARK CIRCUITS. CORALL IS EVALUATED FOR 2 AND 4 PROTECTED INPUT CUBES c AND LUT SIZES OF 2 AND 4 INPUTS. ONLY A SINGLE CONSTRAINED INPUT IS UTILIZED FOR THE GENERATION OF THE CORALL CIRCUITS. ALL EVALUATED CIRCUITS ARE LIMITED TO A MAXIMUM OF 20 INPUTS TO THE FLIP FUNCTION.

Benchmark	CORALL					
	c2l2			c4l4		
	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)
b17	-2.12	-1.70	3.37	0.14	1.06	7.12
b18	-2.50	-0.59	12.94	-1.30	0.68	9.65
c432	117.80	103.21	7.79	494.95	530.18	23.90
c1908	46.64	36.56	-0.65	203.99	152.38	10.56
s35932	2.33	0.46	-3.95	8.62	8.26	-3.95
s38584	0.81	-6.83	8.05	6.88	8.86	3.64

by no more than 10% as compared to the unobfuscated version of the circuit. Applying the three techniques on all of the large benchmark circuits results in no more than a 3% increase in area, with both SFLL-Flex and CORALL providing further reductions in occupied area after resynthesis of the modified logic cone. Implementations of all three techniques result in large overheads in area for the small ISCAS'85 benchmark circuits, where the c1908 and c432 circuits increase in area by over 100%. On average, a 20% reduction in area and a 2.14% increase in performance was observed as compared to SFLL-HD when implementing the CORALL architecture utilizing four protected cubes c and a LUT size l of four. The largest variation occurs in the timing overhead, as different cones are secured by each technique for each benchmark circuit.

The results listed in Table III indicate that the implementation of CORALL results in similar overheads in PPA as SFLL while providing a significantly higher resilience to the SAT attack for circuits that require a large percentage of corruption between the activated and locked IC. In addition, CORALL provides greater flexibility than SFLL, as the number of cubes and LUT size are adaptable parameters that tune between the security of a circuit, as measured by the number of iterations of the SAT attack, and the overhead in area, power, and performance of implementing CORALL on the circuit. For example, the timing overhead of the c432 benchmark circuit secured with CORALL *c4l4* is 23.90%, as listed in Table III. As a four input LUT contributes to the delay of the flip function, decreasing the LUT size to three inputs reduces the logical depth of the critical path of the circuit implementing the flip function, which reduces the overhead in delay to 0%. If increased resilience to the SAT attack is required, an additional cube is added instead to limit the impact on the timing of the circuit.

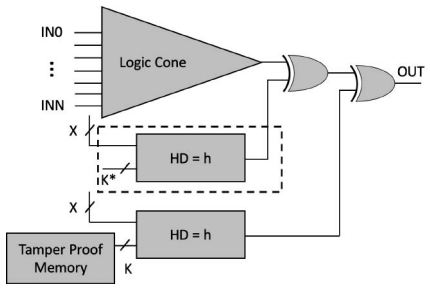


Fig. 9. Block diagram of the SFLL-HD logic locking technique, where the perturb unit is shown within the dotted box and with the correct key labeled as K*. The restore unit, composed of the tamper proof memory and the bottom HD block, corrects any modifications introduced by the perturb unit.

V. INCREASING THE RESILIENCE OF MODIFIED LOGIC CONES AGAINST STRUCTURAL ATTACKS

There are a variety of proposed structural attacks to determine the key of a logic locked IC [26]–[28] that target the fixed and predictable structure of the modifications made to the original, unobfuscated, logic cones of the circuit. While out-of-cone techniques provide a means to generate SAT attack resistant circuits, targeting the structure of the modified logic cone allows an adversary to determine the key without access to an oracle IC, which indicates a significant security vulnerability.

The majority of the structural attacks are developed for the SFLL-HD^h topology. A block level schematic of the SFLL-HD^h flip function (restore unit) and the modified logic cone (perturb unit) is shown in Fig. 9. The current structural attacks are capable of identifying the perturb unit and extracting the hard coded key without access to an oracle IC when a resynthesis of the original logic cone to integrate the perturb unit does not produce significant topological mixing.

The work described in [26] proposes two different structural attacks. The first structural attack is based on the unnateness of the perturb unit of SFLL-HD^0, where the perturb unit simplifies to a single comparator. A Boolean function is described as positive unate with regard to x if switching x from 0 to 1, while leaving all other variables unchanged, never results in the function switching from 1 to 0. A negative unate function behaves exactly the opposite, where x transitioning from 1 to 0 never results in the function switching from 0 to 1 when all other variables are held constant. The single minterm cube

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

utilized in SFLL-HD^0 is either positive or negative unate with regard to every input to the perturb circuit, which results in the determination of the protected cube for 18 of the 20 circuits analyzed in [26].

In addition, an attack on SFLL-HD^h that utilizes properties of the Hamming distance checker is proposed in [26]. The attack exploits the observation that protected cube inputs that are a distance of $2h$ from one another provide information regarding the key. Any bits where the two protected input cubes agree must be the correct key bits [26]. The use of the $2h$ protected cube distance is further expanded, where the remaining key bits are iteratively searched for with a SAT solver.

The work described in [27] assumes that an attacker is able to find a protected cube and then use a bit coloring attack to determine the key in $n-1$ queries. The grouping of bits is based on whether the modified logic cone is equal to the original logic cone. When it is not, the two input patterns are $2h$ distance apart. The property is similar to the separation in the Hamming distance observed in [26].

A functional reverse engineering approach is utilized in [28] instead of the separation in the Hamming distance. The BSIM tool described in [39] is applied to identify comparators and adders within the circuit, which constitute components of the circuit that implement the Hamming distance perturb and restore unit.

Structural attacks are prevented using cryptographic techniques, including one-way functions and an AES block as described in [27] and [40], respectively. Both one-way functions and AES increase the difficultly of determining the input conditions of the circuit from the output constraints, but do not integrate well with the original logic cone. Therefore, modified approaches such as described in [28] allow for identification of the perturb unit, which is then simply removed, exposing the original logic cone to an adversary.

The work in [34] prevents structural attacks by inducing a stuck-at fault into the original logic circuit. The introduction of the stuck-at fault guarantees modification to the original logic cone without requiring resynthesis to implement the changes to the original circuit. The technique is referred to as subtractive since elements are no longer added to the original circuit. The primary drawback of the SFLL-Fault methodology is the run-time required to select a fault that is correctable with a limited flip function circuit, as over an hour was required per fault on the evaluated benchmark circuits [34]. CORALL instead provides additive modifications to resist structural attacks while requiring less computational resources to implement the technique.

A. Structural Attack Resilience of CORALL

Increasing the resilience of out-of-cone logic locking techniques to structural attacks requires 1) better mixing of the original logic cone with the perturb function and 2) nonidentical modifications to the logic cone that prevent identification and exploitation of the logical structure of the perturb function due to the possession of knowledge of the logic locking algorithm. The first modification to the locking mechanism of the circuit when implementing CORALL is the elimination of the XOR gate in the perturb unit, which reduces the complexity of integrating the perturb function with the original logic cone. To remove the XOR, the algorithm must determine a set of input constraints that result in a constant logic 0 or logic 1 output from the original logic cone. The output of the circuit is now flipped with standard OR and AND gates, depending on the desired output of the modified cone.

Algorithm 2: Protected Cube Generation

```
Input: FSM netlist netlist, LUT Size lut_size,
Common Probability common_prob,
Number of Cubes num_cubes;
curr_func = netlist;
num_cone_inputs = length(curr_func.inputs);
flip_funcs = [];
for num_cubes do
    while num_cone_inputs > lut_size do
        if common_inputs then
            /* Set cf_var, cf_val from common
               inputs                          */
        else
            /* Choose random cf_var and cf_val
               */
        func_cfs = get_cfs(curr_func, cf_var);
        if common_inputs then
            curr_func = func_cfs[cf_val];
        else if num_ins_cf0 < num_ins_cf1 then
            curr_func = func_cfs[0];
        else if num_ins_cf1 < num_ins_cf0 then
            curr_func = func_cfs[1];
        else
            curr_func = func_cfs[cf_val];
        /* Add set input to constraints     */
        if rand(0, 1) <= common_prob then
            /* Append to common inputs      */
        if num_cone_inputs <= 2 * lut_size then
            /* Check for 2-LUT function
               criteria                        */
    end
    flip_funcs.append(curr_func)
end
/* Add correction circuitry for flip_funcs
   */
```

In order to determine the modified logic cubes for use in CORALL, an approach based on iterative cofactors is applied. For a Boolean function, the positive cofactor of a variable x_2 is given by (6), while the negative cofactor is given by (7). The positive cofactor sets x_2 to logic 1 and returns the remaining logic function, while the negative cofactor sets x_2 to logic 0 and, similarly, returns the remaining logic function.

$$f_{x_2} = f(x_0, x_1, 1, x_3, \ldots, x_n) \tag{6}$$

$$f_{\bar{x}_2} = f(x_0, x_1, 0, x_3, \ldots, x_n) \tag{7}$$

The formulaic description of utilizing the iterative cofactor to determine the modified logic cubes is provided as Algorithm 2. The while loop included in Algorithm 2 is the primary pseudo-code describing the generation of the protected cubes. The algorithm begins by checking if there are previous common inputs to utilize as the branching variable for the cofactor. If not, a random variable and logic value are generated for the branching of the cofactor. The positive (f_x) and negative ($f_{\bar{x}}$) cofactors are determined for the given cofactor input variable *cf_var*. If the branching direction is already chosen by the common input, then the respective positive or negative cofactor is selected as the current function *curr_func*.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

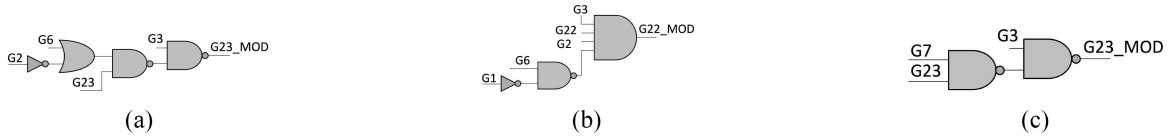


Fig. 10. Three different perturb functions generated with the developed algorithm that selects protected cubes through iterative cofactors. Note the topological differences between the three generated circuits and the variance in the support set of the functions.

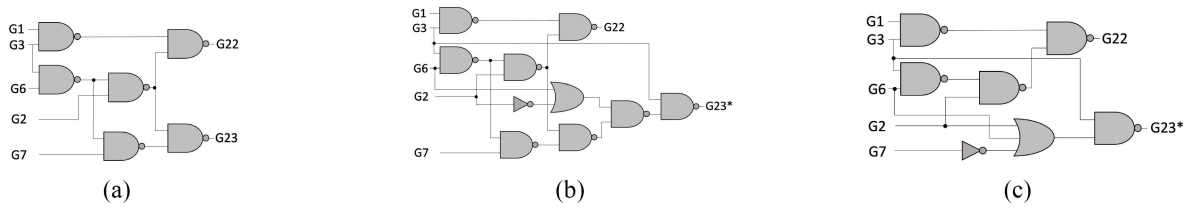


Fig. 11. Procedural stages of the modifications to the original logic cone to include the logic of the perturb unit shown in Fig. 10(a). The schematics depict (a) an unmodified c17 circuit, (b) a modified logic cone before resynthesis, and (c) a modified logic cone after resynthesis.

Otherwise, the number of inputs to the positive and negative cofactor functions are considered for the branching direction. The polarity of the cofactor with the smallest support set is chosen as the *curr_func*. Utilizing the cofactor with the smallest support set reduces the number of branches taken, which results in fewer constrained inputs and, therefore, the generation of more input–output pairs with corrupted outputs. If, however, the objective is to lower the output corruption, then the cofactor with the largest support set is selected. If the positive and negative cofactors include an equal number of inputs, then the cofactor polarity is chosen at random. The set variable and the polarity of the given set variable are added to a hash map of the constrained inputs, which are utilized when generating the logic of the perturb unit.

The computational complexity, as given by the runtime of executing Algorithm 2, is bounded by the cofactor operation and the number of iterations of the while loop required to determine the protected cube. In the worst case, the cofactor algorithm needs to update the entire graph structure to a constant value, which is bounded by $O(|V|*|E|)$, where $|V|$ is the set of graph vertices and $|E|$ the set of graph edges. However, in most cases, the cofactor operation does not have to update the entire graph. Rather, the cofactor operation typically only updates a small portion of the graph corresponding to an average runtime complexity on the order of $O(\log(|V|*|E|))$. The number of iterations is limited to n, where n is the number of primary inputs to the circuit. The total runtime complexity of the operation is then bounded by $O(n*\log(|V|*|E|))$. Note that the number of iterations of the algorithm and the cofactor complexity are correlated. As a result, a cofactor iteration that significantly modifies the graph also eliminates a subset of the n input variables, which reduces the total number of executed iterations of the cofactor algorithm.

To produce increased variation in the logical structure of the generated perturb unit, a term described as *common probability* is introduced to the algorithm. The common probability is the likelihood that a variable from a previous protected input cube is applied again on another cube, which results in the generation of noncomparator-based logical functions and a significantly large amount of variation within the logic of the perturb unit. The resulting variation increases the difficulty of successfully executing a structural attack on the modified cone. Three logical topologies of the perturb unit generated using the iterative cofactor technique on the c17 ISCAS'85 benchmark circuit are shown in Fig. 10. The variation in the logic of the perturb unit provides increased difficulty to an adversary examining the netlist to determine the structure of a given perturb unit. The procedural stages of implementing the modifications to the logic cone that integrate the perturb circuit shown in Fig. 10(a) with the ISCAS'85 c17 benchmark circuit are shown in Fig. 11. The modified logic cone with no resynthesis is shown in Fig. 11(b), which is the worst case structural topology as the additional perturb logic is not well integrated with the original cone. However, the similarity of the logic of the perturb unit to the original logic cone of the circuit increases the difficulty of detecting the perturb unit, even when resynthesis does not produce an optimal mixing of the perturb logic with the original logic cone. The resynthesized logic cone of c17 modified to include the perturb unit shown in Fig. 10(a) is depicted in Fig. 11(c), which demonstrates that resynthesis of the modified logic cone results in a logical topology that is much more difficult to subdivide and extract.

In addition, for each iteration of the while loop, a check is performed to determine if the number of remaining inputs to the cube-generation function is less than $2*lut\_size$. If the number of inputs is less than $2*lut\_size$, the algorithm explores the feasibility of mapping the remaining logical function to two LUTs connected with an AND gate. The remaining inputs must allow for independent groupings of literals. As an example, for a function with literals *ABCD*, one possible grouping of the literals is *AB* and *CD*, which are then mapped to two 2-input LUTs. For the standard CORALL architecture, independent groups with last level functions of AND/NOR are searched for first, as AND/NORs are readily mapped to two LUTs followed by an AND gate. If more flexibility in the logical topology of the circuit is needed, the first level of AND gates connected to the LUTs, as shown in Fig. 4, are converted to AND/OR gates [8]. The conversion to AND/OR gates allows for the use of AND, OR, NAND, and NOR gates as last level logic.

If the number of remaining inputs to the cube-generation function is greater than $2*lut\_size$, then the algorithm continues to iterate until the remaining cube-generation function fits into a single LUT, or the remaining function is a constant logic 0 or logic 1. The generated function is then added to a

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
OUTPUT CORRUPTION OF THE MODIFIED LOGIC CONE FOR THE CORALL ITERATIVE COFACTOR PROTECTED CUBE SELECTION ALGORITHM AND SFLL-HD^0. THE NUMBER OF INPUTS TO THE SECURED CONE OF THE CORALL ARCHITECTURE IS PROVIDED. THE CORRUPTION IS MEASURED AS THE NUMBER OF ERRORS PER SECOND, ASSUMING AN IC OPERATING AT 1 GHZ.

Benchmark	Number of Cone Inputs	CORALL Corruption	SFLL-HD^0 Corruption
ex5	7	6.51E+08	3.97E+06
apex4	9	4.97E+08	9.80E+05
ex1010	10	9.10E+07	4.89E+05
c1908	33	1.00E+06	5.82E-02
c432	36	4.00E+04	7.28E-03
apex2	36	1.28E+08	7.28E-03
c499	41	1.80E+00	2.27E-04
seq	38	4.80E+08	1.82E-03
c1355	41	1.00E+04	2.27E-04
k2	29	7.00E+04	9.31E-01
c3540	50	4.00E+06	4.44E-07
c880	36	2.20E+07	7.28E-03
dalu	27	1.10E+05	3.73E+00
i9	13	4.10E+07	6.11E+04
i8	17	2.20E+07	3.81E+03
c5315	67	7.00E+06	3.39E-12
i4	47	4.00E+06	3.55E-06
i7	7	2.35E+08	3.97E+06
c7552	80	1.43E+03	4.14E-16
c2670	107	8.40E+07	3.08E-24
des	19	8.60E+07	9.54E+02
b17	150	9.54E+02	2.80E-36
s38584	49	6.00E+07	8.88E-07
s35932	14	5.00E+07	3.05E+04
b18	141	4.77E+02	8.35E-44

list of flip functions (*flip_funcs*) and the algorithm is repeated until the target number of cubes is reached. By utilizing a LUT to capture some of the logic of the perturb unit, even if modifications to the logic cone are identifiable, the adversary does not possess the necessary information required to determine the key.

The iterative cofactor is utilized until the logic of the protected cube is mapped to a LUT, is a constant logic 0, or is a constant logic 1 to ensure that the generated perturb function is of similar logical depth to the original logic cone. The goal is to avoid large differences in the depth of logic between the original circuit and the perturb unit. For example, if the logic of the perturb unit is only reliant on a single input, then the output corruption is high but the logic of the perturb unit is discernible from the original logic cone. In that case, the perturb unit does not integrate with the original logic cone during resynthesis as the logical path only has a depth of one. In contrast, if all inputs to the logical cone are constrained within the perturb unit, the adversary is able to isolate the perturb unit by the logical structure of the circuit. While applying the iterative cofactor of the logical cone reduces the disparity in the logical depth between the logic of the perturb unit and the logic of the original cone, additional constraints on the inputs reduce the corruption of the outputs of the circuit. For an IC operating at 1 GHz, an error rate of one corrupted output every second deems a device unusable. The corruption of the outputs of the benchmark circuits modified using the iterative cofactor-based algorithm is listed in Table IV, where all benchmark circuits secured with CORALL produced more than 4.77E+02 errors per second when locked. The resulting error rate after implementing SFLL-HD^0 is much lower, which provides an adversary with a much more functional IC when an incorrect key is applied. CORALL allows for additional corruption by adding more protected cubes and/or increasing the size of the LUTs. However, both increasing the LUT size and/or the number of cubes results in an increase in the PPA overheads. CORALL is also capable of utilizing cubes with less constraints, which generate greater output corruption. The tradeoff is increased risk to structural attack if the resynthesis of the circuit does not integrate the perturb unit into the original logic cone.

TABLE V
COMPARISON OF THE RESILIENCE OF SFLL-HD, SFLL-FLEX, AND CORALL TO THE STRUCTURAL ATTACKS PROPOSED IN [26]–[28], [41]. A ✔ DENOTES THE TECHNIQUE IS RESILIENT TO THE ATTACK VECTOR AND A ✗ DENOTES THE METHODOLOGY IS SUSCEPTIBLE TO THE ATTACK VECTOR. A ✔* INDICATES THE TOPOLOGY IS PARTIALLY RESILIENT TO THE ATTACK VECTOR.

Attack Vector	Logic Locking Methodology		
	SFLL-HD [21]	SFLL-Flex [21]	CORALL
[26] Unateness	✗	✔*	✔
[26] 2H	✗	✔	✔
[27]	✗	✔	✔
[28]	✗	✔*	✔

By 1) replacing the XOR gate as the output of the perturb unit, 2) generating variation in the structure of the perturb logic, and 3) placing the logic of the perturb unit in the top level LUTs of the CORALL architecture, CORALL protects against all current structural attacks. The unateness check described in [26] no longer applies as the perturb unit is not required to utilize all inputs of the logic cone. The properties of the Hamming distance checker exploited in [26] and [27] also no longer apply as the perturb unit does not utilize a Hamming distance checker and does not have a set logical structure. The mixing of multiple correlated comparators of varying lengths by executing CORALL results in the generation of logic structures similar to the original logic cone, which prevents the execution of the functional reverse engineering attack described in [28]. Even if a comparator is utilized, the support set of the comparator is unknown, which forces the BSIM tool to examine a majority of the logic within the original cone of the IC. With CORALL, removing the topological distinction between the original logic cone and the perturb logic significantly increases the computational cost of a BSIM attack. A summary of the vulnerabilities of SFLL-HD, SFLL-Flex, and CORALL to structural attacks is provided in Table V. While the SFLL-Flex technique is not described as providing cubes of variable length, the flip function does allow for modifications to the size of cubes. Therefore, SFLL-Flex offers partial protection against the unateness attack described in [26] and the functional reverse engineering attack described in [28]. Full protection against both attacks is not possible as the circuit implementing the flip function reveals the variable length of the cubes, which permits the unateness and functional reverse engineering attacks to search a smaller subset of the original logic cone.

The primary challenge in performing a structural attack on CORALL is verifying that the original cone has been restored. Consider the worst case scenario where the added logic is

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
SAT ATTACK ANALYSIS OF CORALL IMPLEMENTED WITH TWO PROTECTED CUBES AND A LUT SIZE OF TWO INPUTS. A TO DESIGNATES A TIMEOUT, WHICH WAS SET TO FIVE DAYS (432 000 S) FOR THE ANALYSIS. THE NUMBER OF NEEDED KEY BITS IS DETERMINED BY UTILIZING (5) AND THE NUMBER OF INPUTS TO THE LOGIC CONE FROM TABLE IV.

Benchmark	# of Keys	# of Iterations	# of Cubes	CPU Time(s)
apex2	144	1.38E+05	3.67E+08	TO
apex4	34	7.00E+02	1.52E+07	146.992
b17	586	5.69E+04	1.20E+09	TO
b18	688	4.27E+04	1.92E+09	TO
c1355	162	7.51E+04	3.52E+08	TO
c1908	130	7.96E+04	3.21E+08	TO
c2670	426	4.17E+04	4.02E+08	TO
c3540	200	7.30E+04	6.31E+08	TO
c432	144	1.02E+05	1.93E+08	TO
c499	162	9.66E+04	4.51E+08	TO
c5315	266	4.07E+04	6.15E+08	TO
c7552	320	4.73E+04	8.28E+08	TO
c880	144	9.11E+04	3.43E+08	TO
dalu	106	5.36E+04	4.83E+08	TO
des	74	3.76E+04	1.24E+09	TO
ex1010	40	6.62E+02	1.37E+07	69.1274
ex5	26	1.86E+02	6.19E+05	1.50104
i4	186	1.20E+05	3.23E+08	TO
i7	26	6.20E+01	3.20E+05	1.0908
i8	66	6.33E+04	5.98E+08	TO
i9	50	1.09E+04	6.77E+07	8144.68
k2	114	7.69E+04	6.42E+08	TO
s35932	56	1.78E+04	1.66E+09	TO
s38584	194	1.94E+04	2.12E+09	TO
seq	152	5.07E+04	7.19E+08	TO

found within the netlist, which is equivalent to extracting the added logic from Fig. 11(b). While other techniques produce a static structural signature of the perturb unit, CORALL yields no such distinct signature. Without a set structural signature, the only means for an adversary to verify that the original logic cone has been extracted is to apply an oracle guided attack.

B. Results of Simulation

The analysis of circuit resilience against the SAT attack is performed for two protected cubes and a LUT size of two inputs when utilizing the developed algorithm that selects protected cubes using iterative cofactors, with results as listed in Table VI. The SAT attack was performed on a Xeon E52687W CPU running at 3 GHz and with 95 GB of RAM. A timeout of 5 days (432,000 s) was set for all circuits when executing the SAT attack. The results listed in Table VI further support the expectation that larger cone sizes, as listed in Table IV for each benchmark circuit, provide greater resilience to the SAT attack as discussed in Section IV-C1. In addition, the output corruption is lower for the algorithm that selects protected cubes using iterative cofactors as compared to the results shown in Table I, which also increases the resilience against the SAT attack as the probability of generating a DIP that causes the flip function to evaluate to logic 1 on the activated IC decreases. The circuits that the SAT attack decrypted the key within 5 days include a small number of inputs to the logic cone, such as the ex1010 benchmark circuit that contains only 10 inputs. Other than the four circuits that were expected to fail due to smaller size, all other circuits timed out after 5 days of executing the SAT attack.

In order to increase the resilience of a circuit to the SAT attack when the number of inputs to a single cone is small but the total number of inputs to the circuit is much larger, as is the case for the i9 benchmark circuit that includes 88 total inputs, securing multiple cones with a single implementation of the CORALL circuit is possible by applying protected cubes from both cones. For merged cones, the logic that interfaces between the two or more cones is prioritized. Combining cones allows for greater resilience against the SAT attack. For example, the key for i7 was determined in 1.09 seconds when executing the SAT attack as the cone included only 7 inputs. However, a larger number of inputs to the flip function is generated by combining three of the smaller cones of the i7 benchmark circuit, which results in a cone with 11 total inputs. The execution time of the SAT attack increases to 192.48 seconds from 1.09 seconds, and the number of iterations of the SAT solver increases to 2048 from 32. The cost of combining the multiple cubes is an increase in the logic of the flip function, which is a tradeoff that must be considered.

The resilience against the SAT attack is also improved by increasing the number of cubes and the LUT size of the CORALL architecture, as discussed in Section IV-C1. The cost of increasing the number of cubes and/or the LUT size is an increase in the power consumption and area of the circuit as well as a potential drop in performance.

The overhead in power, performance, and area (PPA) to secure the ISCAS'85, MCNC, and ISCAS'89 benchmark circuits by CORALL with two protected cubes and a LUT size of two inputs is provided in Table VII. The analysis of the benchmark circuits indicates that the overhead in PPA of a given topology is highly dependent on the original size of the circuit being secured. For example, the i4 circuit includes a total of 120 gates, which results in a greater percentage of the total gates added to implement the CORALL architecture and a corresponding increase in area of 347.04%. However, for the s38584 circuit that includes 11,448 gates, the overhead in area is only 0.23%. The size of the circuit must, therefore, be considered when choosing a logic locking implementation. The CORALL architecture provides a similar overhead in PPA to SFLL-HD as indicated by the results listed in Table II.

In addition, certain benchmark circuits, including apex4 and ex1010 resulted in reductions in the area and/or power consumption and improvements in the performance over the original unobfuscated circuit. The improvements are due to a significant modification to the original logic cone and indicate a strong integration of the logic of the perturb unit and the original logic of the circuit.

The runtime of executing the algorithm that generates protected cubes through iterative cofactors for each benchmark circuit is also listed in Table VII. All of the benchmarks complete within approximately 1 minute or less, with most of the benchmarks requiring less than 1 second. The low runtimes demonstrate the usability of the algorithm to quickly secure a circuit netlist.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
OVERHEAD IN AREA, POWER, AND PERFORMANCE OF IMPLEMENTING CORALL ON A SUBSET OF ISCAS'85, MCNC, ISCAS'89, AND ITC'99 BENCHMARK CIRCUITS WHEN UTILIZING THE ALGORITHM THAT SELECTS PROTECTED CUBES THROUGH ITERATIVE COFACTORS. THE NUMBER OF INPUTS AND THE NUMBER OF GATES ARE PROVIDED FOR EACH BENCHMARK CIRCUIT. THE RUNTIME OF THE ITERATIVE COFACTOR ALGORITHM IS ALSO LISTED FOR EACH BENCHMARK CIRCUIT.

Benchmark	Number of Inputs	Number of Gates	Original			CORRALL			
			Area (um^2)	Power (mW)	Timing (ns)	Area (%)	Power (%)	Timing (%)	Runtime (s)
i4	192	120	2.03E+03	0.29	1.19	347.04	567.22	62.18	0.01
c432	36	160	1.71E+03	0.48	3.85	319.12	259.87	22.86	0.11
c499	41	202	3.90E+03	2.42	2.78	172.49	79.58	20.14	0.03
ex5	8	256	4.64E+03	1.30	1.97	20.03	19.09	3.55	0.02
c880	60	383	3.59E+03	0.89	4.49	42.08	50.17	-4.68	0.05
apex4	9	438	3.00E+04	11.46	4.08	-1.74	-0.25	-0.74	0.07
i7	199	471	5.08E+03	1.53	1.33	25.98	13.50	34.59	0.03
i9	88	522	5.61E+03	1.83	2.61	29.83	20.24	7.66	0.04
c1355	41	546	3.90E+03	2.43	2.78	171.14	99.92	22.30	9.68
ex1010	10	810	2.94E+04	11.11	3.63	-1.24	-0.54	-2.20	0.05
c1908	33	880	4.15E+03	2.12	4.64	118.78	83.43	29.74	4.96
apex2	39	1035	2.78E+03	0.59	2.84	182.70	208.16	-1.41	0.01
c2670	233	1193	5.79E+03	2.44	3.79	28.22	7.93	-2.37	16.46
k2	45	1201	1.04E+04	1.54	4.07	29.93	58.78	1.23	0.07
seq	41	1459	1.87E+04	5.59	3.38	21.34	23.41	16.86	0.05
c3540	50	1669	1.01E+04	4.42	6.63	53.99	29.58	-11.46	6.90
dalu	75	1697	7.49E+03	1.67	5.51	74.81	81.02	-29.40	0.26
i8	133	1831	8.55E+03	1.78	3.01	22.76	34.84	11.63	0.01
c5315	178	2307	1.37E+04	7.13	4.70	55.42	30.70	4.89	3.80
c7552	207	3512	1.67E+04	8.48	8.62	13.54	13.70	4.18	0.02
des	256	4000	3.48E+04	19.03	3.76	2.26	0.91	4.26	0.01
s38584	1464	11448	1.18E+05	40.56	5.22	0.23	-0.02	-2.11	6.56
s35932	1763	12204	9.72E+04	38.04	3.04	-2.84	-6.05	-20.07	5.91
b17	1452	30777	2.47E+05	102.81	23.73	-0.79	-1.43	-0.17	66.75
b18	3357	111241	7.11E+05	402.93	27.67	-0.29	1.24	4.92	20.86

VI. CONCLUSION

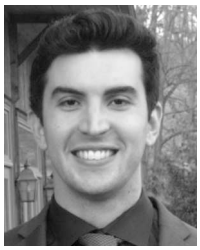
This article introduces a novel out-of-cone logic locking methodology described as CORALL. CORALL provides increased resilience to the SAT attack for circuits that require a large corruption of outputs between the original logic cone and the modified logic cone of an obfuscated version of the circuit. The CORALL architecture results in an increase in the number of iterations required to complete a SAT attack for a 20 input circuit implementing a flip function by 34.41× over SFLL-HD $n/4$ and 82.36× over SFLL-Flex. CORALL provides a large increase in the resilience against the SAT attack while also increasing the percentage of output corruption by at least 40% over SFLL-HD $n/4$ and SFLL-Flex. The improvement in the resilience of a circuit to the SAT attack is provided with similar overheads in area and timing as compared to SFLL-HD. A 20% reduction in area and a 2.14% increase in performance on average was observed as compared to SFLL-HD when implementing the CORALL architecture utilizing four protected cubes and a LUT size of four. An algorithm that generates protected cubes by applying iterative cofactors is also developed to increase the resilience of the circuit to execution of structural attacks on the modified logic cone. The varying implementations of the logic structures of the perturb unit and the mapping of the logic of the perturb unit into the LUTs of the CORALL architecture provide protection against all known structural attacks.

REFERENCES

[1] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *Proc. IEEE Symp. Security Privacy*, San Francisco, CA, USA, 2019, pp. 1–19.
[2] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *Proc. USENIX Security Symp.*, Aug. 2018, pp. 973–990.
[3] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
[4] *Trusted Integrated Chips (TIC) Program*, document IARPA-BAA-11-09, IARPA, Riverdale Park, MD, USA, Oct. 2011.
[5] *Trends in the Global IC Design Service Market*, DigiTimes, Taipei, Taiwan, Mar. 2012. [Online]. Available: http://www.digitimes.com/news/a20120313RS400.html?chid=2
[6] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. IEEE/ACM Design Autom. Test Eur. Conf.*, Munich, Germany, Oct. 2008, pp. 1069–1074.
[7] J. Rajendran *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015.
[8] K. Juretus and I. Savidis, "Reduced overhead gate level logic encryption," in *Proc. IEEE/ACM Great Lakes Symp. VLSI*, Boston, MA, USA, May 2016, pp. 15–20.
[9] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.
[10] S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.
[11] Y. Alkabani and K. Farinaz, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security Symp.*, Aug. 2007, pp. 291–306.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.

[12] K. Juretus and I. Savidis, "Time domain sequential locking for increased security," in *Proc. IEEE Int. Symp. Circuits Syst.*, Florence, Italy, May 2018, pp. 1–5.
[13] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall, "Interlocking obfuscation for anti-tamper hardware," in *Proc. Cyber Security Inf. Intell. Res. Workshop*, Jan. 2013, pp. 1–4.
[14] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit sequential logic obfuscation: Attacks and defenses," in *Proc. IEEE Int. Conf. Circuits Syst.*, Baltimore, MD, USA, May 2017, pp. 1367–1370.
[15] J. Dofe and Q. Yu, "Novel dynamic state-deflection method for gate-level design obfuscation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 2, pp. 273–285, Feb. 2018.
[16] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2007, pp. 674–677.
[17] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, Washington, DC, USA, May 2015, pp. 137–143.
[18] K. Juretus and I. Savidis, "Characterization of in-cone logic locking resiliency against the SAT attack," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Jun. 27, 2019, doi: 10.1109/TCAD.2019.2925387.
[19] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, Jun. 2016, pp. 127–146.
[20] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, McLean, VA, USA, May 2016, pp. 236–241.
[21] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Nov. 2017, pp. 1601–1618.
[22] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 8, pp. 1399–1412, Aug. 2019.
[23] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, McLean, VA, USA, May 2017, pp. 95–100.
[24] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. Great Lakes Symp. VLSI*, May 2017, pp. 179–184.
[25] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerg. Topics Comput.*, early access, Aug. 21, 2017, doi: 10.1109/TETC.2017.2740364.
[26] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, Florence, Italy, Mar. 2019, pp. 936–939.
[27] H. Zhou, Y. Shen, and A. Rezaei, "Vulnerability and remedy of stripped function logic locking," Dept. Elect. Comput. Eng., Northwestern Univ., Evanston, IL, USA, Rep. 2019/139, 2019, [Online]. Available: https://eprint.iacr.org/2019/139
[28] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, and M. Al-Qutayri, "Functional reverse engineering on SAT-attack resilient logic locking," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sapporo, Japan, May 2019, pp. 1–5.
[29] G. S. Tseytin, "On the complexity of derivations in the propositional calculus," in *Studies in the Mathematics and Mathematical Logic, Part 2*, New York, NY, USA: Consultants Bureau, 1968, pp. 115–125.
[30] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 1985, pp. 677–692.
[31] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectron. Center North Carolina, Durham, NC, USA, Rep., 1991.
[32] J. Rajendran, Y. Pino, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. IEEE/ACM Design Autom. Test Eur. Conf. Exhibit.*, Dresden, Germany, Oct. 2012, pp. 953–958.
[33] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the approximation resiliency of logic locking and IC camouflaging schemes," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 347–359, Feb. 2019.
[34] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "ATPG-based cost-effective, secure logic locking," in *Proc. IEEE 36th VLSI Test Symp.*, San Francisco, CA, USA, Apr. 2018, pp. 1–6.
[35] S. Phatharodom, N. Kandasamy, and I. Savidis, "Closed-form estimation of SAT-attack resilience," in *Proc. Gov. Microcircuit Appl. Critical Technol. Conf.*, Mar. 2020, pp. 1–6.
[36] K. Shamsi, D. Z. Pan, and Y. Jin, "On the impossibility of approximation-resilient circuit locking," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, McLean, VA, USA, May 2019, pp. 161–170.
[37] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Trans. Design Test Comput.*, vol. 17, no. 3, pp. 44–53, Jul./Sep. 2000.
[38] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, USA, May 1989, pp. 1929–1934.
[39] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, Grenoble, France, Mar. 2013, pp. 1277–1280.
[40] M. El Massad, S. Garg, and M. Tripunitara, "The SAT attack on IC camouflaging: Impact and potential countermeasures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Jul. 3, 2019, doi: 10.1109/TCAD.2019.2926478.
[41] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "SigAttack: New high-level SAT-based attack on logic encryptions," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit.*, Florence, Italy, Mar. 2019, pp. 940–943.



Kyle Juretus (Student Member, IEEE) received the Bachelor of Science degree in computer and electrical engineering and the Masters of Science degree in computer engineering from Drexel University, Philadelphia, PA, USA, in 2014 and 2016, respectively, where he is currently pursuing the Ph.D. degree with Integrated Circuits and Electronics Lab.

He is a Research Assistant with the Integrated Circuits and Electronics Lab, Drexel University. He is currently a National Defense Science and Engineering Fellow. His research interests include circuit level techniques to prevent intellectual property theft and counterfeiting, mitigating side-channel leakage of integrated circuit designs, and design automation for hardware security.

Ioannis Savidis (Senior Member, IEEE) received the B.S.E. degree in electrical and computer engineering and biomedical engineering from Duke University, Durham, NC, USA, in 2005, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2007 and 2013, respectively.

He joined the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA, in 2013, where he is currently an Associate Professor and directs the Integrated Circuits and Electronics Design and Analysis Laboratory. His current research interests include analysis, modeling, and design methodologies for high performance digital and mixed-signal integrated circuits, power management for SoC and microprocessor circuits, hardware security, including digital and analog obfuscation and Trojan detection, and electrical and thermal modeling and characterization, signal and power integrity, and power and clock delivery for heterogeneous 2-D and 3-D circuits.

Dr. Savidis was a recipient of the 2018 National Science Foundation Early Faculty (CAREER) Award. He also serves on the editorial boards for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *Microelectronics Journal*, and the *Journal of Circuits, Systems and Computers*. He serves on the organizing committees of the IEEE International Symposium on Hardware Oriented Security and Trust, the ACM Great Lakes Symposium on VLSI, and the VLSI Systems and Applications Technical Committee. He is a member of the Institute of Electrical and Electronic Engineers, the Association of Computing Machinery, the IEEE Circuits and Systems Society, the IEEE Communications Society, and the IEEE Electron Devices Society.

Authorized licensed use limited to: Drexel University. Downloaded on June 29,2021 at 20:17:17 UTC from IEEE Xplore. Restrictions apply.