Characterization of In-Cone Logic Locking Resiliency Against the SAT Attack

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

***Abstract*—The resiliency of in-cone logic locking techniques to the satisfiability (SAT) attack is characterized in this paper. An analysis of the parameters of the SAT solver that impact security and a characterization of the effect netlist topology has on the security of the circuit is presented. The analysis of SAT solver parameters and logic structure is used to develop three novel logic locking gate selection algorithms based on maximum fanout free cones (MFFCs) and gate controllability for circuits implementing XOR, look-up table (LUT), and 2 × 1 MUX-based logic obfuscation. The XOR, LUT, and MUX MFFC-based algorithms resulted in an average increase of, respectively, 61.8%, 123.6%, and 38.5% in the minimum number of iterations required to complete the SAT attack across 1,000 different variable orderings of the netlist while applying the locking techniques to 5% of the gates within the netlist. In addition, the SAT attack resiliency and output corruption of the developed algorithms are compared with out-of-cone locking techniques.**

***Index Terms*—Hardware security, key gate selection, logic locking, satisfiability (SAT) attack.**

I. Introduction

THE root of trust in the computing stack is the hardware layer, with the software layer requiring a consistent protocol for interaction. The hardware layer is often inherently assumed to be trusted, which is no longer valid as integrated circuit (IC) fabrication transitions toward a horizontal model. The transition from a vertical manufacturing model, where every step of the IC design and manufacturing process is executed in-house, to a horizontal model is driven by the multibillion dollar investment required to develop an advanced fabrication process [1] and the increased utilization of third-party intellectual property (IP) as a means to reduce design time, add functionality, and reduce costs [2].

Third-party IP, fabrication and test facilities, and the end-user of an IC are all security risks as each represents an untrusted entity in the IC design and fabrication flow. The security threats posed by untrusted third-parties include IP theft, counterfeiting, and overproduction of ICs, and the insertion of harmful circuit modifications (hardware Trojans). Reported cases of counterfeit ICs [3], [4] and ICs with backdoors [5] reveal an increasing urgency and concern of threats attributed to the hardware layer.

One primary research direction to mitigate the threats of untrusted third-parties is obfuscation, which aims to limit the amount of circuit information an adversary is able to recover from an IC. Split manufacturing [6], IC camouflaging [7]–[9], and logic encryption/locking [10]–[14] are three prominent obfuscation methodologies. The use of an active key when applying logic locking protects against untrusted foundries and end-users, as opposed to split manufacturing and camouflaging which allow for black-box use of the IC.

While logic locking provides protection against a wide variety of threats, the satisfiability (SAT) attack [15] has severely limited the effectiveness of logic locking techniques. The SAT attack efficiently determines the key used for logic locking by applying a miter circuit to discern input–output pairs that are then added as additional constraints to the solver. A detailed description of the SAT attack is provided in Section II.

To protect against the SAT attack, a variety of techniques were developed that insert additional logic to artificially control the corruption of the outputs, as described in [16]–[19]. While the proposed out-of-cone techniques allow for a provable methodology to increase the number of required iterations of the SAT attack to decrypt a circuit, a variety of new attack vectors have emerged that subvert the provable security provided against SAT-based attacks [20]–[24].

Instead of adding circuitry to control the level of corruption at the outputs, this paper addresses logic locking as applied within the original logic cone of the IC, which is referred to as in-cone logic locking for the remainder of this paper. In-cone modifications provide increased resiliency against removal attacks as an incomplete netlist remains once the in-cone logic is removed. However, logical masking allows for efficient execution of the SAT attack.

The contributions of this paper include a characterization of in-cone logic locking techniques while analyzing parameters that are uniquely sensitive to execution on a given SAT solver. In addition, an analysis of the impact logic structure has on the efficacy of the SAT attack is performed. Guiding principles to limit the generation of superfluous keys are developed. The analysis of in-cone logic locking is utilized to develop a gate selection strategy based on maximum fanout free cones (MFFCs) and gate controllability for the insertion of XOR gates, heterogeneously sized look-up tables (LUTs), and/or 2 × 1 MUXes. The developed in-cone MFFC-based

Manuscript received January 15, 2019; revised April 12, 2019; accepted June 12, 2019. Date of publication June 27, 2019; date of current version July 17, 2020. This work was supported in part by the Air Force Office of Scientific Research, National Defense Science and Engineering Graduate Fellowship under Grant 32 CFR 168a, in part by the Drexel Ventures Innovation Fund, and in part by the National Science Foundation under Grant CNS-1648878 and Grant CNS-1751032. This paper 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; isavidis@coe.drexel.edu).

Digital Object Identifier 10.1109/TCAD.2019.2925387

Algorithm 1: SAT Attack Algorithm [15]

```
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
$\overrightarrow{K_C} := sat\_assignment_{\overrightarrow{K_1}}(F_i)$
```

selection algorithms are compared to the out-of-cone stripped functionality logic locking (SFLL) technique [18] with regard to SAT security and corruption of circuit outputs.

The rest of the paper is organized as follows. An introduction to the SAT attack is provided in Section II. An overview of the methodologies developed to thwart the SAT attack is provided in Section III. An analysis of parameters that result in a variation in the number of iterations to execute the SAT attack is described in Section IV. A characterization of the effect the logical structure of the netlist has on the provided security against the SAT attack is described in Section V. Design criteria and constraints to limit the number of generated working keys for a locked circuit are described in Section VI. A logic locking gate selection algorithm based on the results provided in Sections IV–VI is described in Section VII. Results comparing the proposed algorithm with prior in-cone logic selection algorithms and the SFLL out-of cone logic locking technique are described in Section VIII. A discussion of trends in logic locking methodologies is provided in Section IX. Concluding remarks are provided in Section X.

II. SAT Attack Overview

The SAT attack requires as input a reverse engineered netlist of the locked circuit in conjunction with an activated IC. The attack, when applied to logic locked ISCAS'85 benchmark circuits, is able to decrypt a majority of the benchmarks within 10 hours [15]. Pseudocode of the algorithm implementing the SAT attack described in [15] 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.

The reverse engineered locked netlist is modified to form a miter circuit, as shown by the example circuit in Fig. 1. The duplicate versions of the circuit are referred to as A and B. Outputs of the duplicated circuits A and B are XORed together, resulting in a logic 1 when a difference between the two is observed. All of the XOR outputs are then connected to an OR gate that evaluates to logic 1 when any of the outputs of circuits A and B differ.

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

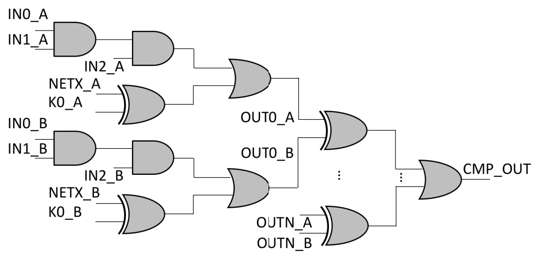

Fig. 1. Miter circuit with replicated A and B versions of the reverse engineered netlist that includes obfuscated gates. Corresponding outputs between the A and B versions of the circuit are XORed. Each XORed signal is then passed to an OR gate to check for any differentiating output.

The Tseytin transformation [25] is applied to convert the logical representation of the circuit into conjunctive normal form (CNF). A Tseytin transformation of a 2-input AND gate is provided as (1), where A and B represent inputs and C represents the output of the gate. Once converted to CNF form, typical SAT solvers are able to solve for the logical conditions of the circuit that generate a logic 1 at the output of the miter circuit. As the only varying signals between the A and B copies of the circuit are the key nets, whenever a condition that generates a logic 1 at the output of the miter circuit is observed, at least one key value is pruned from the key-space. The inputs to the miter circuit that result in such a condition are called distinguishing input patterns (DIPs) and are represented by $\overrightarrow{X_i^d}$ in Algorithm 1.

Once a DIP is generated, the DIP is applied to an activated IC to determine the correct output $\overrightarrow{Y_i^d}$. The resulting $\overrightarrow{X_i^d}$ and $\overrightarrow{Y_i^d}$ are included as additional constraints to the SAT solver by applying constant Boolean propagation, which generates a new set of CNF clauses that are then added as constraints. The process is repeated until no further DIPs are found, resulting in a working key for the locked circuit.

III. SAT Attack Resistant Architectures

The greatest cost, with regard to the computing efficiency of an adversary, occurs when only a single key is eliminated with each DIP while executing the SAT attack. To produce such a computational cost, techniques described in [16]–[19] add additional circuitry to ensure the output is strategically corrupted such that a DIP only eliminates a single key vector.

The typical circuit resembles a structure similar to that shown in Fig. 2, where the flip function limits the amount of data corruption observed at the OUT net. The flip function varies for the different implementations of the circuit and includes: 1) the generation of complementary logic blocks that may or may not result in 1 when an incorrect key is applied [16]; 2) a mask function that generates an incorrect output for each incorrect key for only a single input pattern [17]; and 3) the Hamming distance and LUT-based circuitry that corrects the corrupted functional output of a logic cone based on the applied input [18]. Techniques that artificially control the corruption of the circuit [16]–[19] allow for an algorithmic determination of the number of iterations required to execute the SAT attack. However, out-of-cone methodologies exhibit multiple vulnerabilities

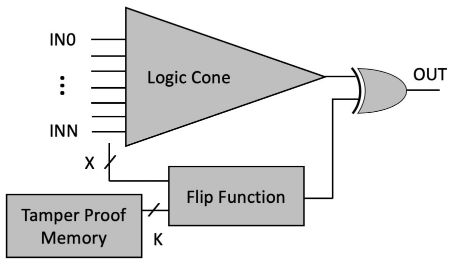

Fig. 2. Generalized circuit topology to defend against the SAT attack utilized in [16]–[19]. Signals represented by *X*, which are extracted from the inputs to the logical cone or the internal signals of the logical cone, are combined with key signals *K* to generate a function that controls the assertion of the flip function.

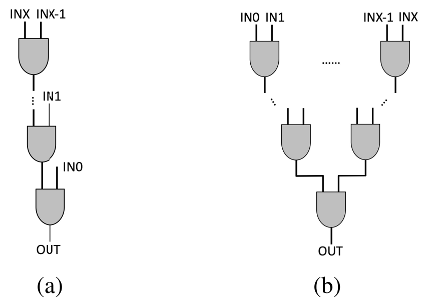

Fig. 3. AND-tree topologies with (a) input at every level and (b) inputs at the level farthest from the output.

to non-SAT-based attacks, including removal attacks [22], approximation attacks [20], [21], and structural attacks [23].

Removal attacks [22] are successful on out-of-cone techniques when the flip function in Fig. 2 is removed and the original logic cone is unaltered. Many of the SAT resistant methodologies, therefore, rely on in-cone logic locking as a means to provide security if the SAT resistant circuitry is removed. However, the addition of in-cone logic locking to the SAT resistant circuit topologies is susceptible to approximation attacks as described in [20] and [21], which are capable of determining the logic locking key without removal of the SAT resistant logic. The attacks utilize separate approaches, but apply a common principal, specifically the use of certain I/O pairs that prune the key space of the in-cone logic locking elements. The algorithm in [20] applies an error calculation after a certain interval of iterations and utilizes random I/O patterns to eliminate in-cone keys. The strategy of using random I/Os is beneficial as the implemented in-cone logic locking techniques generate a large number of corrupted logical outputs. A separate technique described in [21] forces the DIPs to eliminate at least two keys, which results in generated I/O pairs that prune the key space of the in-cone logic locking elements. The SFLL method proposed in [18] alters the logical minterms within the logic cone, which requires a correction of the logical function by the flip signal and, therefore, results in a reduced vulnerability to a removal attack. The attack proposed in [23] utilizes the properties of the modified logic cone and the Hamming distance-based error correction circuitry to efficiently determine the key without the need for an oracle circuit.

While there are a variety of attacks against out-of-cone techniques, the two primary attacks against in-cone logic locking techniques are sensitization attacks [26] and SAT-based attacks [15]. Inserting key gates in a pairwise fashion prevents sensitization-based attacks from significantly pruning the key space [26], leaving SAT-based attacks as the dominant attack vector for in-cone logic locking techniques. The analysis of SAT attack resiliency is, therefore, the focus of this paper, with the goal of increasing the resiliency of the circuit to a SAT attack. The improved resiliency of in-cone logic locking is also of particular benefit to techniques that rely on modifications to the original logic cone, such as IC camouflaging. In addition, the characterization of in-cone logic locking provides a baseline to quantify the security of out-of-cone techniques against non-SAT-based attack vectors.

IV. Avoiding SAT Solver Specific Security

Each developed SAT solver includes differences that effect the solution time and number of iterations required to solve a given problem. This section analyzes variable ordering, initial conditions, and phase heuristics to characterize the variability in the execution of the SAT attack. The presented analysis is not a methodology to increase the security of in-cone logic locking against the SAT attack, but rather to prevent security solutions that target a specific implementation of a SAT solver.

The homogeneous tree structures shown in Fig. 3 are the fundamental logical structures used in the characterization of in-cone logic locking techniques as the AND-tree was shown to provide increased resiliency against the SAT attack [15]. An analysis of variable ordering, and, therefore, the clause order provided to the SAT solver is provided in Section IV-A. The effect of the applied initial DIPs on the number of iterations required to execute the SAT attack is described in Section IV-B. The three phase heuristics of 1) all true, 2) all false, and 3) the advanced heuristic included with Lingeling are analyzed in Section IV-C to characterize the effect that the initial phase of the solver has on the SAT attack.

A. Effect of Variable Ordering on the SAT Attack

The C++ implementation of the SAT attack described in [15] topologically sorts the circuit, which results in a variable ordering of nodes at an equal logical level dependent on the structure of the netlist. Randomly rewriting the original netlist, therefore, provides a means to generate multiple variable orderings. Note that the topology of the circuit is never altered, simply the structural order of the netlist is rearranged. The structure in Fig. 3(a) with 16 inputs is implemented to analyze netlist reordering, where the tree of AND gates is linearly connected. All nodes except for the output are logic locked with an XOR, requiring a key gate at every input and internal net.

From the initial netlist, 1,000 random variable orderings of the linear AND-tree structure shown in Fig. 3(a) are generated and provided as inputs to the SAT solver. The results of executing the SAT attack on the reordered netlist are provided in Fig. 4, with the unaltered netlist order shown as a vertical line. OR, NOR, NAND, XOR, and XNOR trees are also analyzed, providing a quantified distribution of the number of iterations

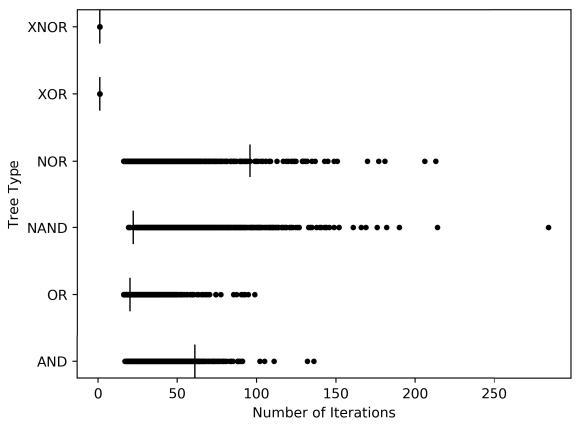

Fig. 4. Simulation of the gate tree structure from Fig. 3(a) with 1,000 random netlist orders quantifying the number of iterations required for the SAT attack to decrypt the tree. All nets within the tree are encrypted for the simulation. The vertical line represents the netlist order used in [15].

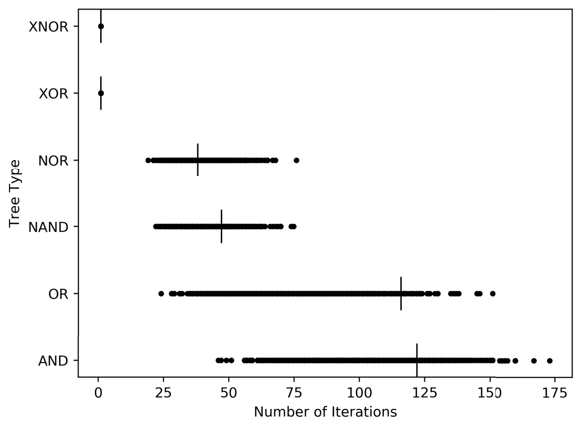

Fig. 5. Simulation of the gate tree structure from Fig. 3(b) with 1,000 random netlist orders quantifying the number of iterations required for the SAT attack to decrypt the tree. All nets within the tree are encrypted for the simulation. The vertical line represents the netlist order used in [15].

of the SAT attack required to decrypt the keys as a function of the variable ordering of the netlist.

For the linear tree structure shown in Fig. 3(a), the results shown in Fig. 4 indicate NAND and NOR trees require more iterations on average to complete the SAT attack than the AND and OR tree structures. The increase in the average number of iterations is due to 1) the distribution of output values being biased toward either logic 0 or logic 1 when applying the secured NAND and NOR tree structures and 2) the SAT attack applying initial DIPs of all logic 0 and all logic 1. The initial DIPs of all logic 0 and all logic 1 do not significantly constrain the key space for the NAND and NOR tree topologies as much as the key space is constrained for the AND and OR trees. As the linear NAND/NOR-based tree structure shown in Fig. 3(a) is vulnerable to a more limited number of significantly constraining DIPs, the average number of iterations required to execute the SAT attack increases. However, the minimum number of iterations across all gates remains relatively constant as the attack selects the more constraining DIPs in earlier iterations.

The pyramid tree structure in Fig. 3(b) is also analyzed for the same six standard gate types with 1,000 random netlist orderings of each, with results provided in Fig. 5. The AND and OR trees yield much stronger security on average than the NAND and NOR tree topologies, exactly opposite of what is indicated by the results provided in Fig. 4 for the tree topology of Fig. 3(a). The small AND-tree topology shown in Fig. 6 illustrates the cause of the increase in the number of iterations when executing the SAT attack. An output of 1 from the AND-tree forces all internal nodes of the tree to logic 1 until an XOR key gate is reached, at which point the SAT solver has an increased degree of freedom. The solver is now allowed to select either a logic 0 or 1 for each input of the XOR gate. The increase in the degrees of freedom result in a decrease in the likelihood of the SAT solver generating an input of all ones as a DIP. Any DIP that is not all ones results in a variety of key combinations that generate a zero at the output of the IC as the AND tree is significantly skewed toward a zero output. Similarly, a DIP of all logic zeros is less likely to be generated for an OR-tree, which allows for the masking of incorrect key sequences and accounts for the increase in the number

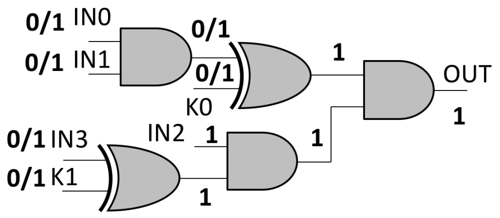

Fig. 6. SAT solver propagating a constant logic 1 at the output of the AND-tree. After a key gate is reached, the solver has an increased degree of freedom when selecting input logic values.

of iterations provided by OR-tree structures as compared to NOR, NAND, XNOR, and XOR gates, as indicated by results shown in Fig. 5.

Independent of the utilized circuit topology, the variation in the distribution of the number of SAT attack iterations due to variable ordering is an important consideration. Without accounting for the variation, the level of perceived circuit security does not necessarily match the true security provided by the implemented obfuscation technique.

B. *Effect of Initial DIPs on the SAT Attack*

Before generating any DIPs with the miter circuit, the SAT attack, as described in [15], applies the initial input constraints of all logic 0 and all logic 1. An analysis of the number of iterations of the SAT attack with and without the initial constraints of inputs of all logic 0 and all logic 1 is performed with results listed in Table I. Each gate type is analyzed with 1,000 random variable orderings of the gate structure shown in Fig. 3(b) with eight inputs to the tree. A significant difference is seen in the number of iterations to complete the SAT attack when comparing the results with and without the initial constraints of all logic 0 and all logic 1. The results listed in Table I indicate a 443% and a 636% increase in the number of SAT attack iterations on average for, respectively, the AND and OR trees when the initial constraints are removed. Similar to variable ordering, the variation in the number of SAT attack iterations due to initial constraints implies a single execution of the SAT attack provides a limited assessment of the security of an IC.

TABLE I
MINIMUM AND AVERAGE NUMBER OF ITERATIONS TO COMPLETE THE SAT ATTACK WITH AND WITHOUT ALL 0/1 INITIAL CONSTRAINTS FOR THE TREE STRUCTURE OF FIG. 3(b)

Gate Type	Number of SAT Attack Iterations			
	0/1 Constraints		No Constraints	
	Min.	Average	Min.	Average
AND	12	24.26	15	131.7
OR	9	17.63	10	129.8
NAND	9	16.77	9	22.59
NOR	8	14.02	8	15.63
XOR	1	1.00	2	2.00
XNOR	1	1.00	2	2.00

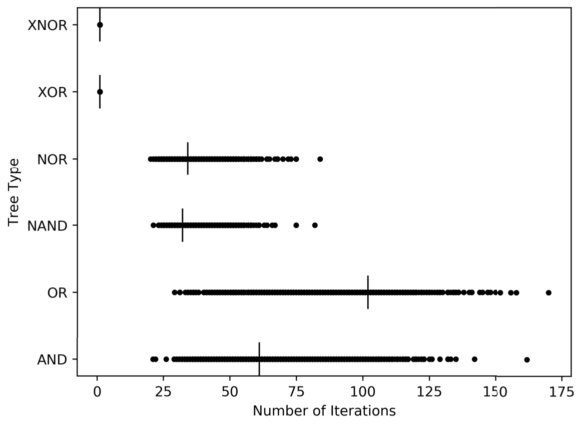

Fig. 7. Simulation of the gate tree structure of Fig. 3b with 1,000 random netlist orders. The number of iterations required for the SAT attack to decrypt the tree with a default SAT solver phase of true is quantified. The vertical line represents the netlist order used in [15]. All nets within the tree are encrypted for the simulation.

C. Effect of Phase Heuristic on the SAT Attack

SAT solvers employ varying phase heuristics to select between a true and false assignment of a given variable. To analyze the effect of the phase heuristic on the efficacy of a SAT attack, initial default phases of 1) all true, 2) all false, and 3) an advanced phase heuristic included with the Lingeling SAT solver [27] are applied to the circuit topology shown in Fig. 3(b), where the pyramid structure includes 16 inputs for each gate type. Applying a default phase of true to all inputs results in the OR-tree topology requiring more iterations on average than the AND-tree structure, as indicated by the results provided in Fig. 7. The increase in the number of iterations of the SAT attack for an OR-tree with a default phase of true is explained in Section IV-A through the example structure shown in Fig. 6. Defaulting to a phase of true, or logic 1, results in a lower probability of the SAT solver selecting a DIP of all zeros, which masks incorrect key information throughout the pyramid structure. As information regarding the circuit is masked, the number of iterations increases on average as compared to an AND-tree structure.

As the AND-tree structure of the ISCAS'85 c2670 benchmark circuit was presented as the primary factor that resulted in an increase in the security of the circuit in [15], other researchers applied an AND-tree topology to increase the security of novel obfuscation techniques [28]. However, the analysis provided in this paper demonstrates that utilizing a single tree topology leads to vulnerabilities as an adversary is able to simply adjust the phase heuristic to more efficiently attack the implemented obfuscation technique. The analysis provided in Section IV indicates that the use of a specific circuit function to increase security provides limited gains as an adversary is able to 1) transform the circuit to shift controllable inputs toward the outputs of the IC, 2) add specific DIPs to the SAT formulation, and/or 3) alter the phase heuristic of the solver to reduce the number of iterations required to determine the key.

V. EFFECT OF LOGIC STRUCTURE ON SAT ATTACK RESILIENCY

The effects of gate heterogeneity, logical reconvergence, and number of keys per node are analyzed as each impacts the efficacy of the SAT attack. The benefits of using an AND-tree structure to secure combinational logic are described in [15] when the relatively smaller ISCAS'85 c2670 benchmark circuit was one of the more challenging to decrypt. Finding similar logical structures in a netlist that exhibit increased resiliency to SAT attacks is, therefore, an important step to secure in-cone logic locking. The study of gate heterogeneity, where the logical structure of the circuit limits the exposure of key sequences to a SAT attack, is described in Section V-A. Logical reconvergence is examined in Section V-B to further characterize the effect of the circuit structure on the ability to secure the logic cone. Lastly, an analysis of the security of an IC against the SAT attack due to the number of applied key bits per node is provided in Section V-C.

A. Analysis of Gate Heterogeneity

The topology shown in Fig. 3(b) is utilized to characterize the effect gate heterogeneity has on the resiliency of a circuit against the SAT attack. The topology includes 16 inputs with a varying combination of distributions of logic gates. The gate types varied from a homogeneous structure of AND, OR, NOR, and NAND gates to a distribution where the gate type under evaluation occupied 10% of all the gates in the netlist and the remaining three gate types occupied 30% each. In order to limit the effect due to random placement of key gates, fixed locations representing 25% of the inputs of the netlist are locked. Each circuit topology is evaluated for 100 random gate types for each gate distribution. The 100 randomly generated circuits are analyzed for 1,000 random variable orderings to quantify the level of security provided by the logical structure of the circuit against the SAT attack. In order to avoid the formation of homogeneous paths, connected gates are not permitted to be of the same type unless all other gate types have already met or exceed the target selection percentage for the given netlist. The average number of iterations required to execute the SAT attack for the 100 randomly generated netlists across the 1,000 variable orderings is shown in Fig. 8.

Both the homogeneous AND and OR trees provide a higher minimum and average number of iterations than the other gate types and gate distributions as shown in Fig. 8. Using more heterogeneous gate structures decreases the number of iterations required to complete the SAT attack on average.

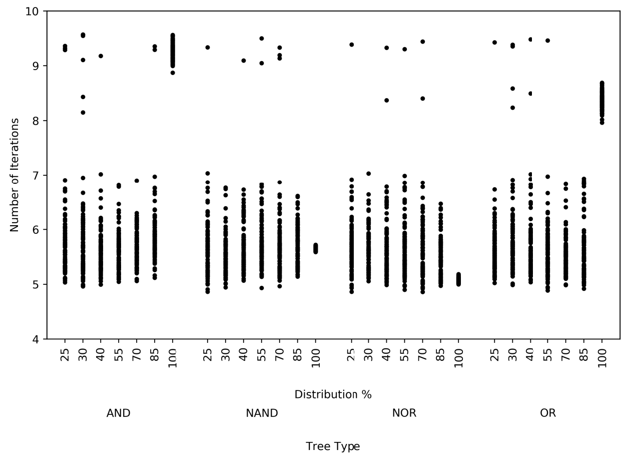

Fig. 8. Analysis of each gate type starting as a homogeneous tree (probability of dominant gate = 1) and then decreasing the probability of selecting a dominant gate to 0.1 (0.25 represents an AND, OR, NAND, and NOR selection probability of 25%). XOR-based logic locking is applied to 25% of the circuit inputs.

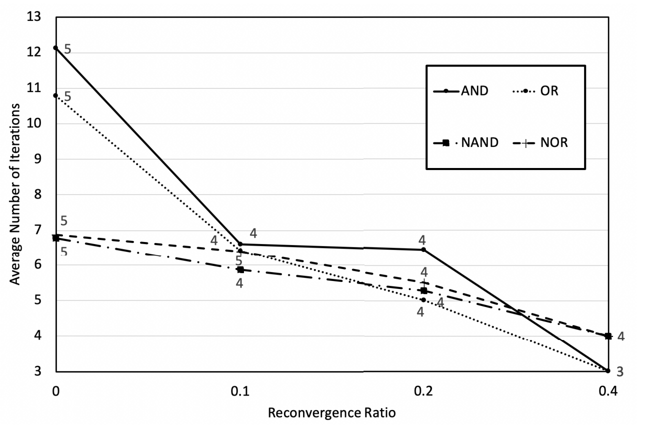

Fig. 9. Analysis of the effect of reconvergence on the number of SAT iterations required to decrypt the key for a tree structure consisting of gates of the same type. The largest minimum number of iterations across the 1,000 configurations of the benchmark circuits for each gate type and reconvergence ratio is listed above each data point.

However, there are cases where the number of iterations increases significantly above the average. Analyzing the structure of the benchmark circuits that require a larger number of iterations indicates that the more resilient circuit topologies limit the degree of freedom of the SAT solver, which results in an increase in the number of iterations. Therefore, the circuits that are most secure include a high disparity in the probability of the output being either logic 0 or logic 1. The increased degrees of freedom for the SAT solver to generate DIPs then favors an input pattern that produces the probabilistically favored output (i.e., 0 for an AND gate and 1 for an OR gate) even in cases where an incorrect key is applied, decreasing the efficiency of the SAT attack. For example, the benchmark circuit requiring the highest average number of iterations has an output probability of 95.67% when producing a logic 0 and 4.33% when generating a logic 1 with a topology consisting of 55% OR gates and 15% AND, NAND, and NOR gates. The benchmark circuit with the lowest average number of iterations for a topology with 25% of each gate type has an output probability of 59.1% when producing a logic 0 and 40.9% when generating a logic 1. The analysis demonstrates that homogeneous tree structures are not the only circuit topologies providing increased protection against the SAT attack. However, the likelihood of randomly generating a heterogeneous circuit topology that sufficiently skews the output probability is low. The controllability of the gates, or the disparity between the probability of a logic 0 and 1, therefore, serves as an important quantifiable parameter to determine the optimal locations to secure in the netlist.

B. Analysis of Logical Reconvergence

Logical reconvergence, where the fanout from a gate subsequently converges at a downstream gate of the circuit, is evaluated to characterize the effect on the efficacy of the SAT attack. Reconvergence of logic is controlled probabilistically by limiting the number of available nodes that connect to the next level of logic. As n number of 2-input gates require $2 * n$ inputs at a given logic level l, limiting the number of gates providing inputs to logic level l to less than $2 * n$ requires logical reconvergence within the netlist.

The reconvergence ratio is used to represent the reduction in the number of inputs applied to logic level l. The maximum number of inputs is set to $2 * n$, which indicates a reconvergence ratio of 0. As an example, a reconvergence ratio of 0.4 implies $2 * n * (1 - 0.4)$ inputs are applied to logic level l. The results characterizing the effect on the average number of iterations of the SAT solver due to probabilistically controlling the logical reconvergence are shown in Fig. 9, where the reconvergence ratio is swept from 0.0 to 0.4. The analysis consists of generating 1,000 random implementations of logic locked circuits, each while assuming homogeneous gate types, a maximum number of inputs of 16, and a 25% overhead in area. Each of the 1,000 generated implementations is then subjected to 100 random variable orderings to further characterize the effect on the efficacy of the SAT attack, as discussed in Section IV-A.

The results shown in Fig. 9 indicate that both the average number and the largest minimum number of iterations (shown above each data point) decrease for all gate types as the reconvergence ratio increases. The reduction in the number of iterations to complete the SAT attack implies an increased challenge in masking key information and, therefore, hiding the structure of the circuit as the reconvergence ratio increases. As the level of logical fanout increases, incorrect key information spreads across a larger segment of the circuit, resulting in a lower probability of masking the applied key bit(s) at the output(s) of the IC.

C. Analysis of the Number of Keys Per Node

The analysis of gate heterogeneity and logical reconvergence demonstrates that the structure of a circuit has an impact on the resiliency against a SAT attack. In-cone logic locking techniques, for the same overhead in area, either secure more nodes of the circuit or implement additional key bits at a single node. The analysis is completed by replacing a gate(s) within the ISCAS'85 c5315 circuit netlist with a LUT, where the number of keys determines the size of the LUT and the

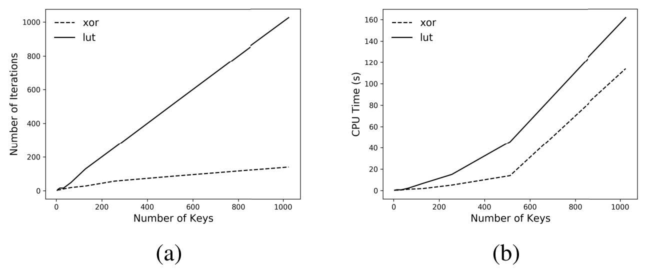

Fig. 10. Characterization of the effect the number of keys applied at a given netlist location has on (a) the number of iterations and (b) the run time required to complete the SAT attack.

number of required LUT select lines is $\log_2(\textit{number of keys})$. Once the number of select lines to the LUT increases beyond the total number of inputs to a functional block, false inputs are added to the LUT. The false inputs are generated from nearby nets within the circuit and have no effect on the output of the LUT.

LUT insertion at a single node is compared with random insertion of XOR gates at multiple nodes within an ISCAS'85 c5315 benchmark circuit, with results shown in Fig. 10. The number of keys is linearly correlated with the number of iterations required to complete the SAT attack when replacing a gate, as shown in Fig. 10(a). The number of iterations increases at a much faster rate for LUT-based logic locking, as compared to random XOR insertion. The SAT attack has a linear correlation with the number of keys at a given net as a single DIP only exposes a single key bit of the LUT. In addition to the linear increase in the number of iterations, the run time of the SAT attack increases exponentially due to the exponential increase in the number of variables provided to the SAT solver. Random XOR insertion matches more closely to LUT-based insertion with regard to CPU run time as the SAT attack spends more time searching for a satisfying assignment within the formulated miter circuit. The tradeoff between securing more nodes and increasing the number of keys per node also depends on the allocated overhead in performance, power, and area. Inserting a LUT requires the use of k 2-input MUXes, where k is the total number of key bits, which results in a large power, performance, and area overhead within a small section of the netlist.

VI. Managing the Number of Correct Keys

In addition to effecting the ability to secure against the SAT attack, the logical structure of the circuit also generates conditions that lead to undesired working keys for in-cone logic locking. To demonstrate, a set of four benchmark circuits are secured and the key space of each is exhaustively searched to determine all functional keys. The number of correct keys generated for four different gate selection algorithms is listed in Table II. For the i4 benchmark circuit, 128 functional keys are present when the logic locking algorithm described in [29] is implemented with a permitted overhead in area of 5%.

The work in [29] highlights the importance of avoiding the selection of consecutive gates for logic locking as the selected gates effectively function as a single security gate. Three new rules are, therefore, proposed to control the generation of functional keys for in-cone logic locking, and are described as 1) buffer/inverter separation, 2) insertion on XOR/XNOR gates, and 3) reconvergent fanout checks.

TABLE II
Analysis of the Number of Correct Keys for Four Gate Selection Methodologies and Benchmark Circuits. The Area Allocated for Logic Locking Is Provided in Parenthesis

Benchmark	[29]	[11] MUX	[11] XOR	[10]
c432 (10%)	2	1	1	1
c499 (10%)	4 (5%)	2	1	32
c880 (5%)	2	2	2	2
i4 (5%)	128	-	-	1

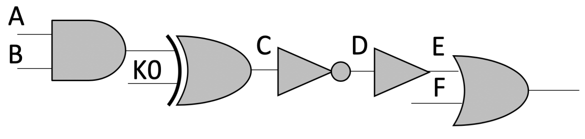

Fig. 11. Example of a buffer/inverter chain topology that must be accounted for when selecting nets to secure. Selection of nets *C*, *D*, or *E* results in an increase in the number of functional keys.

1) Buffer/Inverter Separation: The first rule accounts for the presence of buffers or inverters within the netlist. The circuit topology of concern is shown in Fig. 11, where a key gate is already placed within the circuit and the selection algorithm is determining which net to secure next. If the algorithm selects nets *C*, *D*, or *E* in Fig. 11, the number of generated functional keys increases as the resulting topology is essentially the same as a series of key gates.

To prevent the selection and insertion of secure gates on nets of a buffer/inverter chain, the netlist is traversed from the input(s) and the output(s) of a placed key gate with buffers and inverters marked until a non-buffer or non-inverter is reached. Referring to Fig. 11, the nets *C*, *D*, and *E* are marked with a penalty term, alerting the selection algorithm that inserting a key gate at these locations creates multiple functional keys.

2) Insertion on Selected XOR/XNOR Gates: The concern with inserting key gates before or after an XOR/XNOR gate is the symmetric nature of the function. Securing both inputs of a two-input XOR/XNOR results in a functional key when the two key bits are correct and when the correct key is inverted. Netlists containing higher percentages of XOR/XNOR gates are, therefore, susceptible to many working keys, which explains the results shown in Figs. 4 and 5, where both the XOR and XNOR gates require the least number of iterations to decrypt the circuit when executing a SAT attack. To limit the selection of logic locking gates to a single net connected to a single input or the output of the XOR/XNOR gate, a penalty term is applied to the remaining inputs and the output of the XOR/XNOR gate after the insertion of a key gate. For example, if input *A* of a two input XOR gate is secured, both input *B* and the output of that XOR gate are penalized to avoid the generation of undesired working keys.

3) Reconvergent Fanout Check: Another important consideration is the reconvergence of fanouts within the netlist, as inputs to a gate generated from a common source potentially reduce the total number of input combinations possible at the given gate. An example of such a circuit is shown in Fig. 12,

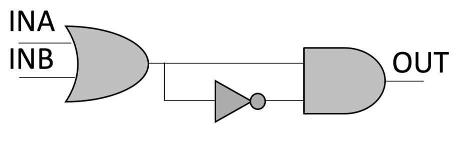

Fig. 12. AND gate topology with dependent inputs limiting the possible functional outputs, which results in multiple correct key values when both inputs are secured.

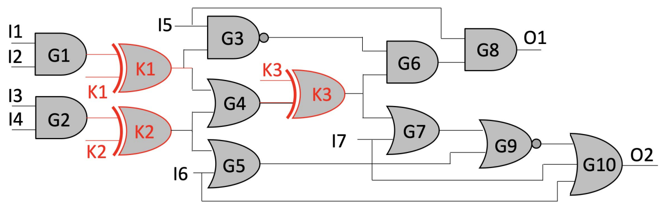

Fig. 13. Example circuit that includes pairwise secure key gates K1, K2, and K3 [29]. The pairwise security of the key bits prevents the key sensitization attack proposed in [26].

where the inputs to the AND gate never equal one another (i.e., 00 or 11). The reduction in the possible input combinations of the gate results in the generation of undesired correct keys when both inputs are secured.

The simplest solution is to only allow the locking of a single input of each gate. However, such a solution results in a large reduction in the possible nets to secure within the circuit. Tracking and simulating the location and the effect of each reconvergent fanout is computationally expensive. Therefore, a tradeoff between increasing computational complexity and not limiting the nets available when selecting gates for logic locking is required to efficiently determine the potential for an undesired increase in the number of functional keys. After a key gate is inserted, the netlist is traversed from the added secure gate to the inputs of the circuit, with each visited node marked. Each of the remaining unsecured inputs to the target gate are then traversed to the inputs of the circuit. If any of the marked nodes are encountered during the traversal, a penalty is applied to those specific inputs of the gate.

VII. Overview of MFFC Gate Selection

The circuit shown in Fig. 13 includes three pairwise secure gates to prevent the key sensitization attack [26], [29]. While the key gates cannot be muted to sensitize the key bits to individual outputs, consider the case where I1 $=$ 0, I2 $=$ 1, and I5 $=$ 1. The keyspace of the circuit shown in Fig. 13 is now significantly constrained as setting K1 $=$ 1 results in an incorrect output for the majority of the applied keys. As the DIP sequences significantly constrain the keyspace of the circuit, a novel gate selection strategy is required. The fanout of incorrect key information is a primary factor in the efficient determination of the key, as shown in Section V-B.

Increasing the resiliency of a circuit that implements in-cone logic locking in part relies on controlling the amount and location of leaked information that the SAT attack uses to generate DIPs. To ensure that the fanout of logical nets does not create unintended leakage channels for the SAT attack to utilize, the concept of MFFCs is developed. An MFFC is a cone of logic where a net within the MFFC converges to a given output net [30]. As an example, consider the ISCAS'85 benchmark c17 circuit shown in Fig. 14, where the MFFC of net N23 includes all gates contained within the dashed box. Since the fanout of G4 includes a logical path that does not converge to N23, it is not part of the MFFC. In addition, edges that converge at net N23 are potential locations to utilize for XOR/XNOR locking as all leaked information must propagate through N23. For the circuit shown in Fig. 14, the edges $N16 \rightarrow G6$ and $N11 \rightarrow G5$ are, therefore, added to the list of eligible locations for the insertion of a locking gate. Returning to the example shown in Fig. 13, if K1 is inserted only on the edge $G1 \rightarrow G4$ and K2 is inserted only on the edge $G2 \rightarrow G4$, the three gates are now pairwise secure and within an MFFC. The previous scenario, where I1 $=$ 0, I2 $=$ 1, and I5 $=$ 1, is no longer dependent on any key values, forcing the SAT attack to generate DIPs that are reliant on all three key bits. Generating constraints that depend on a greater number of key bits increases the circuit resiliency against the SAT attack as more DIPs are needed to determine the correct key sequence.

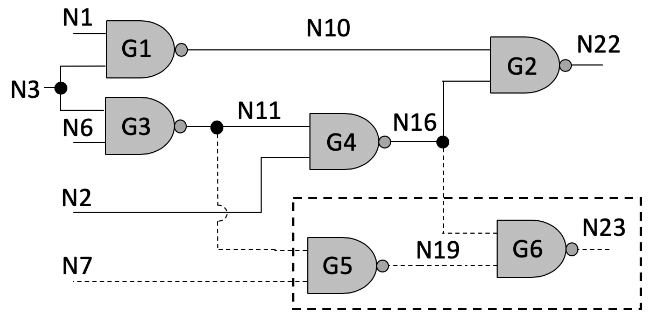

Fig. 14. Example of an MFFC for net N23. Gates contained within the MFFC converge to node N23. Dashed edges represent graph edges where potential XOR/XNOR locking is possible while still being contained within the MFFC.

A. Determination of MFFCs

To determine the MFFC, a single node is selected and Algorithm 2 is executed for the list of directly connected input nodes. When all outputs from a given node are connected to nodes within the MFFC, the given node is added to the MFFC. The algorithm is then applied recursively to input nodes that fit the MFFC criteria, termed *next_lvl_gates* in Algorithm 2. Edges that fanout to a node contained within the MFFC are added to the MFFC for the current node evaluated by Algorithm 2 ($G6$ in the example circuit shown in Fig. 14), but the starting node of the edge is not added to *next_lvl_gates* unless all of the fanout paths from the node are also contained within the MFFC. Nodes and/or edges are traversed and added to the MFFC in topological order. Although not guaranteed pairwise secure, the generated order provides inherent pairwise security. The pairwise security due to the generated MFFC protects against a sensitization attack and requires a brute force attack with complexity of approximately 2^k, where k is the total number of key bits. Therefore, the SAT attack remains the dominant attack vector. If, however,

Algorithm 2: Determination of MFFC

```
Input: Logic level gates, level_gates;
MFFC of Node N list, $M_N$
next_lvl_gates = [ ];
for gate in level_gates do
    fanout_free = True;
    tmp_edges = [ ];
    foreach gate_output in gate_outputs do
        if gate_output in mffc_nodes then
            /* Add gate $\rightarrow$ gate_output to
                tmp_edges                              */
        else
            fanout_free = False;
    end
    if fanout_free then
        /* Add node to MFFC                            */
        /* Add gate_inputs of gate to
            next_lvl_gates                             */
end
if next_lvl_gates not empty then
    /* Recursively call function with
        next_lvl_nodes                                 */
```

nonmutable edges are needed, then the algorithm in [29] can be applied to the MFFC nodes.

B. Weighting of the MFFCs

Once the MFFC of each node is calculated, the MFFCs are sorted by a weighted sum of controllability to determine the MFFC best suited for the insertion of logic locking gates. The measure of gate controllability is used due to the results described in Section V-A, which demonstrated that skewed gate probabilities result in an increase in the resiliency of the circuit against the SAT attack. The controllability is heuristically determined by estimating the probability of a logic 1 and logic 0 at each node and then taking the absolute difference between the calculated probabilities. The controllability values for the nodes in an MFFC are averaged together and multiplied by the total number of nodes in the MFFC, or multiplied by the target number of key gates to insert if the total number of gates fit within the entire MFFC.

C. XOR-Based Insertion

After the weighted MFFCs are determined, Algorithm 3 is executed to iteratively insert XOR/XNOR gates. The algorithm greedily chooses nets to insert an XOR/XNOR based on the order of the weighted MFFCs. The MFFC with the highest score is then iterated through in reverse order, as nets closest to the inputs of the circuit are added last to the MFFC. The edges and nodes within the MFFC are examined for marks, which represent a previous selection of the edge and/or a penalty applied to avoid the generation of multiple working keys. The process is repeated until the target number of key gates are inserted into the netlist.

Algorithm 3: Insertion of Logic Locking Gates

```
Input: Number of Gates to Insert, num_gates;
Node MFFCs, mffcs
while keys_inserted < num_gates do
    if use_mux then
        enc_nodes = get_mux_enc_node(mffcs);
        num_inserted = insert_mux(enc_nodes);
    else if remaining_keys > 4 and use_luts then
        lut_netlist = get_lut_enc_nodes(mffcs);
        num_inserted = insert_lut(lut_netlist);
    else
        enc_node = get_xor_enc_node(mffcs);
        num_inserted = insert_xor(enc_node);
    keys_inserted += num_keys_inserted;
end
/* Determination of XOR Node                           */
/* Weighted MFFC selected and reversed
    so nodes close to inputs are
    selected                                           */
for elem in reversed_mffc do
    /* Check elem has not been consumed
        and is not marked                              */
    if elem in netlist_gates and not_marked then
        /* Select Node for locking                     */
end
```

D. Heterogeneous LUT-Based Insertion

In addition to a gate insertion strategy for XOR based logic locking, a LUT based insertion strategy is described that allows for the placement of heterogeneously sized LUTs into the circuit netlist. The LUT based algorithm provides a means to increase the number of keys per node, which was shown to increase the number of iterations and execution time of the SAT attack, as described in Section V-C. The maximum allowed LUT size is set by the *max_expansion* variable in Algorithm 4, which is limited to six inputs in this paper.

When LUT-based selection is enabled and the number of keys available is greater than four, the execution of Algorithm 3 determines the best candidate locations within the netlist to insert a LUT. The process begins by examining the MFFC with the highest weight for each node in reverse order. However, in contrast to XOR based insertion, the LUT based algorithm must examine fan-in gates as possible additions to the LUT structure.

Execution of Algorithm 4 provides a list of candidate gates to include in a functionally equivalent LUT for each gate within the highest weighted MFFC. The MFFC of a given node is also utilized in the algorithm to ensure that gates do not include fanouts that require gate replication, avoiding additional overhead when inserting the LUT. When executing Algorithm 4, the inputs to the current selected gate are analyzed. The gate is considered a candidate for inclusion in the LUT if no prior mark for selection is included or no applied penalty is present. The number of expanded nodes are then updated to ensure the number of inputs to the LUT has not exceeded six. The algorithm is recursively called for all potential candidate gates, allowing for the inclusion of additional

Algorithm 4: Determination of LUT Candidate Gates

```
Input: MFFC of Node $node\_mffc$, List of Node MFFCs $mffcs$
/* Access $node$ from $node\_mffc$ */
for $gi$ in $get\_gate\_inputs(node)$ do
    if $not\_marked$ and $gi$ in $node\_mffc$ then
        /* Add to candidate nodes */
        $num\_expansion = get\_num\_expansion(netlist, lut\_nodes)$;
        if $num\_expansion < max\_expansion$ then
            $get\_expansion\_candidates(mffcs[gi])$
end
```

gates into the logically equivalent LUT. As buffers/inverters within the MFFC of a node do not expand the number of inputs, the buffers/inverters are possible candidate gates for inclusion in the logically equivalent LUT, which allows for additional obfuscation of the original netlist without requiring an increase in the size of the LUT.

The determination of candidate nodes to include in the LUT is repeated for each of the nonmarked nodes within the weighted MFFC, one of which is selected as the insertion point of the LUT. Once the insertion node for the LUT is selected, the list of candidate gates generated from the execution of Algorithm 4 is utilized to include additional gates in the LUT until 1) all consumable gates are exhausted, 2) the *max_expasion* limit is reached, or 3) the limit on the number of key bits is reached. The netlist of the LUT is then provided to Algorithm 3, which splices the netlist of the LUT into the logic structure of the original netlist and removes the original gates that are obfuscated by the LUT. For this paper, the LUT insertion procedure is repeated until the target number of key bits (a user defined parameter) is reached or the remaining number of key bits is less than four (the minimum number of key bits needed for a 2-input LUT representation of a gate), at which point the algorithm utilizes XOR/XNOR based logic locking to reach the required number of key bits.

E. 2×1 MUX-Based Insertion

A 2×1 MUX-based interconnect logic locking algorithm is also developed, as described by Algorithm 5. A node within an MFFC is first determined that produces the maximum probability skew, as described by results provided in Section V-A. The second input to the MUX is then selected from nodes that are 1) at lower levels than the *max_node* in the topological order and 2) provide a minimum difference in the probability of the node outputting a logic 0 when subtracting the probability of logic 0 produced by *max_node*. The constraint on topological order prevents the generation of combinational logic cycles within the netlist. The difference in the probability between the two nodes is minimized to reduce the ability of the SAT attack to determine DIPs that reveal the key of the MUX. Once selected, the *max_node* is marked to prevent a repeated selection of the same node. A MUX is inserted into the netlist at the target node, with the process described by Algorithm 5 repeated until the requisite number of MUXes are added.

Algorithm 5: Determination of MUX Inputs

```
Input: List of Nodes in MFFCs $mffc\_nodes$
$max\_skew = 0$; $max\_node = None$;
/* Determine node with max skew */
for $node$ in $mffc\_nodes$ do
    if $not\_marked$ then
        $prob\_skew = get\_prob\_skew(netlist, node)$;
        if $prob\_skew > max\_skew$ then
            $max\_skew = prob\_skew$;
            $max\_node = node$;
end
/* Find node with close skew */
$max\_node\_lvl = lvl(max\_node)$;
$min\_skew = 1.0$; $obs\_node = None$;
for $node$ in $mffc\_nodes$ do
    if $not\_marked$ and $(lvl(node) < max\_lvl\_node)$ then
        $skew\_diff = get\_skew\_diff(netlist, max\_node, node)$;
        if $skew\_diff < skew\_min$ then
            $skew\_min = skew\_diff$;
            $obs\_node = node$;
end
/* mark $max\_node$ */
return $max\_node$, $obs\_node$;
```

VIII. Results of MFFC-Based Gate Selection

The gate selection algorithm is applied to the ISCAS'85 benchmark circuits and to the combinational circuits from the Microelectronics Center of North Carolina (MCNC), as described in [15]. Characterization is performed based on selection of a fraction of the nodes to obfuscate, with an analysis of locking 5%, 10%, 25%, and 50% of the nodes performed. The number of implemented key bits remains the same across all the obfuscation techniques. The developed MFFC and gate controllability based XOR, LUT, and MUX-based obfuscation techniques are compared with gate selection methodologies for in-cone logic locking that include iolts14 [31], dac12 [29], rnd [10], toc13XOR [11], and toc13mux [11].

The analysis is performed on 1,000 random variable orderings of the netlist and for the Lingeling [27], false, and true phase heuristics. All simulations were completed on a Xeon E52687W CPU running at 3 GHz with 95 GB of RAM. The execution time of the SAT attack is limited to a maximum of 24 hours (86,400 seconds).

The number of SAT iterations for each gate selection methodology when securing 5% and 25% of the gates of the circuit is provided in Figs. 15 and 16, respectively. The results indicate that the developed MFFC-based selection algorithm significantly increases *the minimum number of iterations* required to complete the SAT attack for most of the evaluated benchmark circuits. The exceptions are the c880 benchmark circuit with 5% of the gates secured in the netlist and the i9 benchmark circuit with 25% of the gates secured, where the minimum number of iterations is at most 20.8% lower than the dac12 [29] methodology. In all cases, the LUT based selection algorithm meets or further increases the minimum

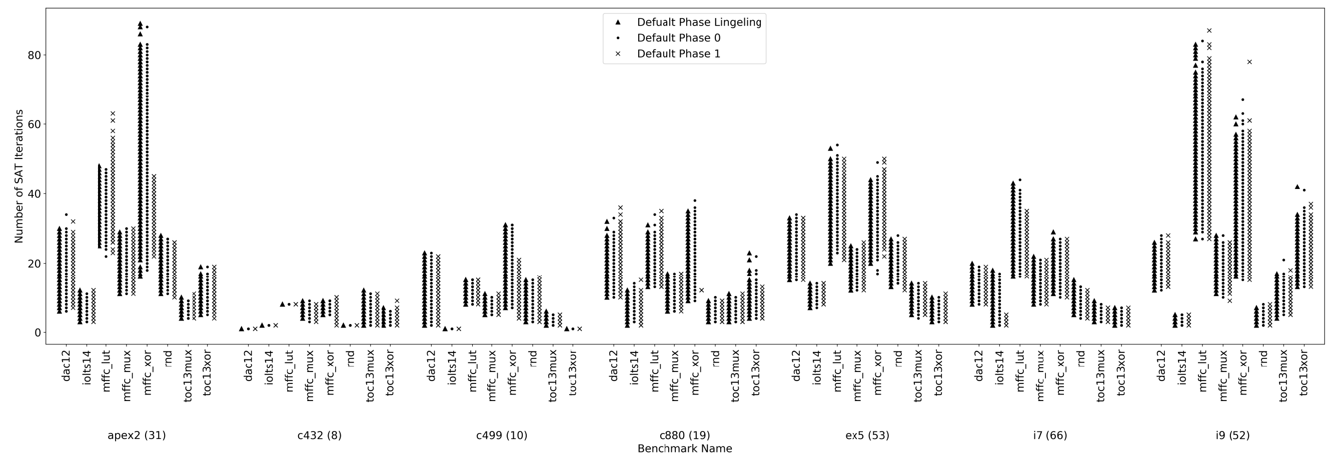

Fig. 15. Number of required iterations of the SAT attack for in-cone logic locking gate selection methodologies when securing 5% of the gates in the netlist. Number of keys shown in parenthesis next to benchmark name. Evaluations are completed with 1,000 random variable orders of the netlist and the default phase of all logic 0, all logic 1, and the Lingeling phase heuristic [27].

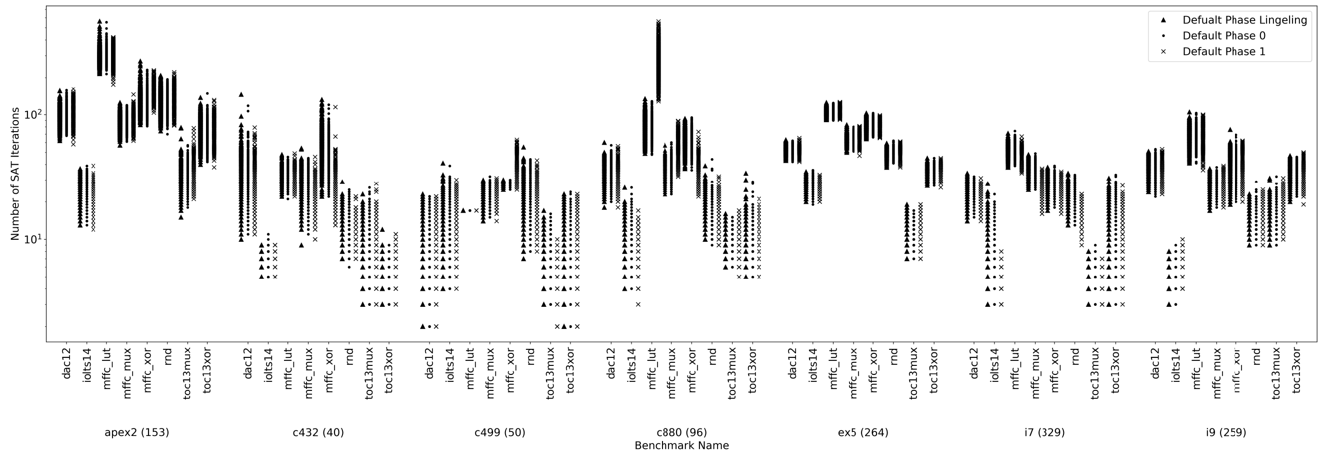

Fig. 16. Number of required iterations of the SAT attack for in-cone logic locking gate selection methodologies when securing 25% of the gates in the netlist. Number of keys shown in parenthesis next to benchmark name. Evaluations are completed with 1,000 random variable orders of the netlist and the default phase of all logic 0, all logic 1, and the Lingeling phase heuristic [27].

number of iterations required to execute the SAT attack. Overall, the XOR MFFC gate selection strategy increases the minimum number of iterations to complete the SAT attack by 61.8% and increases, on average, the mean number of iterations by 80.1% across all benchmark circuits. The XOR based selection strategy results in a 196.2% increase in the average CPU run time, with a 119.6% increase in the minimum run time. The LUT MFFC based selection methodology increases the minimum number of iterations to complete the SAT attack by 123.6% and increases, on average, the mean number of iterations by 82.8% across all benchmark circuits. The LUT based selection strategy results in a 88.2% increase in the average CPU run time, with a 72.4% increase in the minimum run time. The 2 $\times$ 1 MUX MFFC based selection methodology increases the minimum number of iterations to complete the SAT attack by 38.2% and increases, on average, the mean number of iterations by 23.1% across all benchmark circuits. The MUX based selection strategy results in a 26.6% increase in the average CPU run time, with a 17.4% increase in the minimum run time. The power, performance, and area overheads of the apex2, c432, c499, c880, ex5, i7, and i9 benchmark circuits are provided in Table III for the XOR, LUT, and MUX MFFC based gate selection algorithms and in Table IV for the dac12, iolts14, rnd, toc13XOR, and toc13mux methodologies. All techniques were synthesized using Synopsys DC Compiler in a 180 nm technology, with the toggle rate of the key inputs set to 0%. Overall, the overheads for all of the techniques are large, with a few benchmark circuits for each technique resulting in single digit overheads in area, power, and/or timing. The one exception is in the power overhead after implementing the iolts14 selection methodology, which inserts AND and OR gates into the netlist to increase the switching probability of nets specifically to enhance the detection of Trojans for a given key. However, as the correct key is unknown when performing the power analysis, the switching activity of the circuit is modified, which results in a low or reduced power consumption for iolts14. While the overheads for the techniques are significant, the primary optimization criteria is security. A reduction in the overhead is possible

TABLE III
AREA, POWER, AND PERFORMANCE OVERHEADS OF THE XOR, LUT, AND MUX OBFUSCATION TECHNIQUES IMPLEMENTING THE MFFC AND GATE CONTROLLABILITY BASED GATE SELECTION METHODOLOGIES FOR THE BENCHMARK CIRCUITS APEX2, C432, C499, C880, EX5, I7, AND I9

Benchmark	mffc_xor			mffc_lut			mffc_mux		
	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)
apex2	30.9	15.5	43.1	22.7	14.0	5.2	33.5	18.5	77.7
c432	23.1	12.8	9.3	21.4	5.4	18.7	14.6	15.3	20.4
c499	6.7	9.2	35.9	4.5	0.6	16.5	7.5	14.8	17.7
c880	19.3	10.1	8.4	16.5	13.6	4.1	17.7	14.6	13.0
ex5	35.0	4.4	51.6	22.5	5.0	37.1	35.7	19.6	116.7
i7	39.0	13.0	65.4	26.1	0.7	52.3	40.5	14.9	28.5
i9	25.9	75.4	108.7	18.9	38.5	52.8	28.7	26.6	112.7

TABLE IV
AREA, POWER, AND PERFORMANCE OVERHEADS OF THE DAC12, IOLTS14, RND, TOC13XOR, AND TOC13MUX LOGIC LOCKING GATE SELECTION METHODOLOGIES FOR THE APEX2, C432, C499, C880, EX5, I7, AND I9 BENCHMARK CIRCUITS

Benchmark	dac12			iolts14			rnd			toc13xor			toc13mux		
	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)
apex2	29.7	14.4	3.7	8.6	5.2	8.6	41.3	40.7	23.4	31.2	19.0	42.0	25.7	4.9	54.6
c432	18.1	8.6	21.0	23.1	3.8	22.4	21.2	8.8	35.1	31.4	25.2	43.3	37.3	34.9	51.3
c499	6.1	6.6	21.0	1.7	-3.7	3.6	9.0	1.7	39.5	8.4	18.6	33.1	7.6	6.1	19.0
c880	18.5	6.1	2.3	5.5	4.2	4.8	15.6	6.9	9.1	16.9	13.5	22.5	21.4	19.2	38.6
ex5	47.4	27.7	67.7	10.4	1.2	5.9	64.3	52.0	41.4	73.6	77.1	87.6	75.6	41.5	303.8
i7	60.0	11.8	103.1	22.4	1.8	1.5	60.1	32.1	79.2	62.5	32.4	116.2	55.2	51.4	99.2
i9	19.4	64.4	55.0	4.8	-63.2	27.5	31.7	67.9	44.1	23.6	174.4	101.7	27.1	14.2	203.5

TABLE V
MINIMUM PERCENTAGE AND AVERAGE PERCENTAGE IMPROVEMENT IN THE MINIMUM NUMBER OF CUBES, MINIMUM NUMBER OF ITERATIONS, AND MINIMUM CPU TIME OVER 100 VARIABLE ORDERINGS TO COMPLETE THE SAT ATTACK IS PROVIDED FOR THE XOR, LUT, AND MUX BASED OBFUSCATION TECHNIQUES IMPLEMENTING THE MFFC AND GATE CONTROLLABILITY BASED LOGIC LOCKING GATE SELECTION METHODOLOGY FOR A 128 BIT KEY AS COMPARED TO THE DAC12, IOLTS14, RND, TOC13XOR, AND TOX13MUX GATE SELECTION METHODOLOGIES

Benchmark	Num. Gates	mffc_xor			mffc_lut			mffc_mux		
		Num. Cubes	Num. Iterations	CPU Time (s)	Num. Cubes	Num. Iterations	CPU Time (s)	Num. Cubes	Num. Iterations	CPU Time (s)
c7552	3512	8.91E+05 **Min:** 54.04% **Avg:** 423.7%	46 **Min:** 39.29% **Avg:** 447.6%	8.67 **Min:** 8.44% **Avg:** 598.3%	8.87E+05 **Min:** 53.35% **Avg:** 421.4%	47 **Min:** 42.42% **Avg:** 459.4%	4.65 **Min:** -41.90% **Avg:** 269.2%	7.38E+05 **Min:** 54.04% **Avg:** 334.0%	38 **Min:** 15.15% **Avg:** 352.3%	2.53 **Min:** -68.30% **Avg:** 101.3%
c5315	2307	4.89E+05 **Min:** 93.48% **Avg:** 263.2%	35 **Min:** 59.09% **Avg:** 235.3%	7.04 **Min:** 5.28% **Avg:** 648.7%	5.91E+05 **Min:** 133.60% **Avg:** 338.6%	44 **Min:** 100.0% **Avg:** 321.5%	5.01 **Min:** -25.0% **Avg:** 433.0%	5.57E+06 **Min:** 93.48% **Avg:** 313.6%	40 **Min:** 81.81% **Avg:** 283.2%	3.13 **Min:** -53.30% **Avg:** 232.2%
ex1010	5066	2.39E+06 **Min:** 67.78% **Avg:** 679.5%	93 **Min:** 69.09% **Avg:** 874.8%	10.03 **Min:** 71.70% **Avg:** 547.0%	1.74E+06 **Min:** 22.17% **Avg:** 467.5%	69 **Min:** 25.45% **Avg:** 623.3%	6.08 **Min:** 186.0% **Avg:** 292.2%	1.48E+06 **Min:** 67.78% **Avg:** 381.1%	57 **Min:** 3.63% **Avg:** 497.5%	4.91 **Min:** 10.01% **Avg:** 216.5%
des	6473	1.59E+06 **Min:** 127.1% **Avg:** 697.1%	49 **Min:** 133.3% **Avg:** 926.6%	5.6 **Min:** 137.60% **Avg:** 362.1%	1.34E+06 **Min:** 92.04% **Avg:** 547.1%	42 **Min:** 100.0% **Avg:** 780.0%	3.97 **Min:** 68.60% **Avg:** 227.8%	1.43E+06 **Min:** 127.1% **Avg:** 616.7%	44 **Min:** 109.52% **Avg:** 821.9%	4.9 **Min:** 107.80% **Avg:** 304.2%

by implementing a gate selection methodology that applies a multiobjective optimization algorithm.

In addition to the benchmark circuits characterized by results included in Fig. 15, an analysis of the largest benchmark circuits from ISCAS'85 and MCNC is performed, with the results listed in Table V for a key size of 128 bits. The minimum number of cubes, number of iterations, and CPU time for the XOR, LUT, and MUX MFFC based gate selection methodologies across 100 variable orderings are listed. The minimum percentage and average percentage increase for the XOR, LUT, and MUX based techniques when compared with dac12, toc13XOR, tox13mux, rnd, and iolts14 is also provided. Shorter CPU run times are observed for the LUT and MUX based techniques as each iteration is generally shorter than XOR based methods. The increased iteration time for the XOR techniques may be an artifact of using a nonincremental SAT attack [32]. Another consideration is that the iteration time increases when the oracle model is not provided to the SAT attack, forcing an adversary to query the IC and wait for the corresponding response, which often increases the iteration time.

A. Comparison With SFLL

The corruption of the primary output signals of the obfuscated benchmark circuits is analyzed, with results listed in Table VI for 1,000 random input patterns and 100 random key patterns applied to each random input. The toc13XOR and toc13mux techniques provide the highest percentage of corruption as the techniques were developed to produce a 50% Hamming distance when compared with the original outputs of the circuit. In addition, implementation of the MFFC based techniques results in a lower percentage of corruption as gate controllability is used as an insertion metric, which favors less observable gates within the netlist.

The implemented SFLL technique that includes Hamming distance error correction circuitry is $k - \log_2 \binom{k}{h}$ secure against

TABLE VI
AVERAGE PERCENTAGE OF CORRUPTED OUTPUTS OF THE APEX2, C432, C499, C880, EX5, I7, AND I9 BENCHMARK CIRCUITS FOR 1,000 RANDOM INPUT PATTERNS. ONE HUNDRED RANDOM KEY SEQUENCES ARE APPLIED TO EACH OF THE 1,000 RANDOM INPUT PATTERNS

Security Technique	apex2	c432	c499	c880	ex5	i7	i9
dac12	12.68	17.85	0.78	3.32	6.31	47.30	40.74
iolts14	2.61	30.18	18.70	13.52	18.33	11.96	38.57
mffc_xor	0.09	13.34	3.17	1.97	0.39	2.23	20.36
mffc_lut	0.13	11.22	3.08	1.82	0.29	1.91	28.07
mffc_mux	0.03	6.00	1.04	1.54	0.69	2.87	23.00
rnd	7.05	21.09	6.03	14.70	12.67	27.58	43.31
toc13mux	68.08	37.35	14.99	15.88	16.86	37.51	44.37
toc13xor	50.14	45.24	27.96	19.08	19.08	48.70	43.90

TABLE VII
ANALYSIS OF THE AVERAGE NUMBER OF ITERATIONS AND OUTPUT CORRUPTION PERCENTAGE FOR $h = k/4$, $k/3$, AND $k/2$ WHEN IMPLEMENTING SFLL

Benchmark	# Keys	Average Iterations			Corruption		
		k/4	k/3	k/2	k/4	k/3	k/2
apex2	30	263.71	17.87	3.46	0.38	5.44	24.72
c432	8	4.57	4.57	1.83	19.48	19.48	39.73
c499	10	11.38	4.27	2.03	8.40	20.69	37.11
c880	19	67.63	9.66	2.84	1.47	9.81	29.03
ex5	53	5352.55	139.39	4.63	0.02	0.71	19.28
i7	66	43129.06	202.51	5.11	2E-3	0.49	17.65
i9	52	3546.07	102.61	4.54	0.03	0.97	19.60

the SAT attack, where k is the total number of key bits and h is the target Hamming distance of the primary outputs [18]. The probability of a successful attack is, therefore, $q/2^{k-\log_2 \binom{k}{h}}$, with q number of queries made to the circuit [18]. The number of altered minterms is $\binom{k}{h}$, while the number of non-altered minterms is $(2^n - \binom{k}{h})$, where n is the total number of inputs. The corruption of the outputs of the circuit for a non-altered minterm is $\binom{k}{h}/2^k$, while an altered minterm results in $(2^k - \binom{k}{h})/2^k$ corrupted outputs. The number of iterations for a success rate of 50% and the corresponding percentage of corrupted outputs are listed in Table VII for $h = k/4$, $k/3$, and $k/2$. The data indicates that for lower k values and increased levels of output corruption, in-cone techniques are capable of outperforming out-of-cone techniques. The analysis is completed by applying the SAT attack, and not the more recently developed dominant attack vector for SFLL described in [23], where properties of the Hamming distance checking circuitry are exploited.

IX. DISCUSSION OF TRENDS IN LOGIC LOCKING

While out-of-cone techniques such as SFLL provide provable security against the SAT attack when the number of key bits is large, the data listed in Table VII indicates that the security is not adequate when the number of key bits is limited and the desired level of corruption of the primary outputs is high. In-cone techniques, such as the MFFC based methods developed in this paper, outperform SFLL for a low number of key bits and for circuit conditions that require a higher level of output corruption. However, even though the resiliency against the SAT attack increases, execution of the SAT attack still determines the correct key within tens of seconds. The concern is that even for large circuits, scan chain access partitions the IC into smaller blocks that provide an easy means to attack obfuscated circuits. Scan chain obfuscation, such as the work proposed in [33]–[40], is consequently vital to the effective implementation of logic locking techniques. However, many of the methods that obfuscate the scan chain do not secure the logic between the registers, which allows for the reverse engineering of the IC. Modifications to the logic cones between registers is, therefore, needed, while accounting for the analysis and results presented in this paper since attacks without scan chain access exist [41]. Any novel technique must ensure obfuscation of both the scan chain and logic cone, with selection of the specific logic locking implementation dependent on the number of cone inputs, desired level of output corruption, and the cone structure, as indicated by the results in this paper.

X. CONCLUSION

The effect logical structure has on the security provided by in-cone logic locking techniques against the SAT attack is analyzed for gate heterogeneity, logical reconvergence, and the number of key bits per node. Based on the analysis, three novel algorithms utilizing MFFCs and gate controllability are developed to increase the resiliency of in-cone logic locking against the SAT attack. The developed XOR MFFC based algorithm results in an average increase in the minimum number of iterations to complete the SAT attack by 61.8% and an increase in the average number of iterations by 80.1% when securing 5% of the gates within the netlist, while the LUT MFFC based algorithm increases the minimum number of iterations by 123.6% and the average number of iterations by 82.8%. The 2×1 MUX MFFC based selection methodology increases the minimum number of iterations to complete the SAT attack by 38.2% and increases, on average, the mean number of iterations by 23.1% across all benchmark circuits. The developed techniques are analyzed for output corruption, and are compared with the SFLL out-of-cone technique presented in [18], where the developed techniques outperform SFLL when the size of the key space is limited. The analysis of logic locking techniques described in this paper provides guidance to increase the resiliency of in-cone techniques to SAT attacks.

REFERENCES

[1] DigiTimes. (Mar. 2012). *Trends in the Global IC Design Service Market*. [Online]. Available: http://www.digitimes.com/news/a20120313RS400.html?chid=2
[2] *Trusted Integrated Chips (TIC) Program*, IARPA, Riverdale Park, MD, USA, Oct. 2011, pp. 4–5.
[3] *Defense Industrial Base Assessment: Counterfeit Electronics*, U.S. Dept. Commerce, Washington, DC, USA, 2010.
[4] Committee on Armed Services, *Inquiry Into Counterfeit Electronics Parts in the Department of Defense Supply Chain*. Washington, DC, USA: United States Senate, May 2012.
[5] S. Skorobogatov and C. Woods, “Breakthrough silicon scanning discovers backdoor in military chip,” in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, Leuven, Belgium, Sep. 2012, pp. 23–40.

[6] R. W. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931, 2004.

[7] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Camouflaging a standard cell based integrated circuit," U.S. Patent 8 151 235, 2012.

[8] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang, "Building block for secure CMOS logic cell library," U.S. Patent 8 111 089, 2012.

[9] J. P. Baukus, L. W. Chow, J. Clark, and G. J. Harbison, "Conductive channel pseudo block process and circuit to inhibit reverse engineering," U.S. Patent 8 258 583, 2012.

[10] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, Oct. 2010.

[11] J. Rajendran *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015.

[12] K. Juretus and I. Savidis, "Reduced overhead gate level logic encryption," in *Proc. IEEE/ACM Great Lakes Symp. VLSI*, May 2016, pp. 15–20.

[13] K. Juretus and I. Savidis, "Reducing logic encryption overhead through gate level key insertion," in *Proc. IEEE Int. Conf. Circuits Syst.*, May 2016, pp. 1714–1717.

[14] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Des. Test. Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.

[15] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust*, May 2015, pp. 137–143.

[16] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, Jun. 2016, pp. 127–146.

[17] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust*, May 2016, pp. 236–241.

[18] 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,

[19] 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.

[20] 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. Oriented Security Trust*, May 2017, pp. 95–100.

[21] 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.

[22] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[23] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *Proc. IEEE Design Autom. Test Europe Conf.*, Mar. 2019, pp. 936–939.

[24] 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 Europe Conf.*, Mar. 2019, pp. 940–943.

[25] 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.

[26] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2012, pp. 83–89.

[27] A. Niemetz, M. Preiner, and A. Biere, "Boolector at the SMT competition 2015," Inst. Formal Models Verification, Johannes Kepler Univ., Linz, Austria, Rep. 15/1, Jun. 2015.

[28] K. Shamsi, W. Wen, and Y. Jin, "Hardware security challenges beyond CMOS: Attacks and remedies," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2016, pp. 200–205.

[29] M. Yasin, J. J. V. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.

[30] J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 2, pp. 137–148, Jun. 1994.

[31] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *Proc. IEEE Int. On-Line Test. Symp.*, Jul. 2014, pp. 49–54,

[32] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb, "Incremental SAT-based reverse engineering of camouflaged logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1647–1659, Oct. 2017.

[33] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, "Secured flipped scan-chain model for crypto-architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 11, pp. 2080–2084, Nov. 2007.

[34] Y. Atobe, Y. Shi, M. Yanagisawa, and N. Togawa, "Dynamically changeable secure scan architecture against scan-based side channel attack," in *Proc. IEEE Int. SoC Design Conf.*, Nov. 2012, pp. 155–158.

[35] R. Karmakar, S. Chattopadhyay, and R. Kapur, "Encrypt flip-flop: A novel logic encryption technique for sequential circuits," *Comput. Res. Repository*, vol. abs/1801.04961, pp. 1–14, Aug. 2018.

[36] J. Lee, M. Tehranipoor, and J. Plusquellic, "A low-cost solution for protecting IPs against scan-based side-channel attacks," in *Proc. IEEE VLSI Test Symp.*, Apr. 2006, pp. 1–6.

[37] M. A. Razzaq, V. Singh, and A. Singh, "SSTKR: Secure and testable scan design through test key randomization," in *Proc. IEEE Asian Test Symp.*, Nov. 2011, pp. 60–65.

[38] S. Paul, R. S. Chakraborty, and S. Bhunia, "VIm-Scan: A low overhead scan design approach for protection of secret key in scan-based secure chips," in *Proc. IEEE VLSI Test Symp.*, May 2007, pp. 455–460.

[39] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure scan and test using obfuscation throughout supply chain," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 9, pp. 1867–1880, Sep. 2018.

[40] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 5, pp. 818–830, May 2018.

[41] M. E. Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2017, pp. 33–40.

Kyle Juretus (S'11) received the Bachelor of Science degree in computer and electrical engineering and the Master 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 the Integrated Circuits and Electronics (ICE) Laboratory.

His current 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.

Mr. Juretus is currently a National Defense Science and Engineering Fellow.

Ioannis Savidis (S'03–M'13–SM'18) 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.

In 2013, he joined the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA, 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 electric and thermal modeling and characterization, signal and power integrity, and power and clock delivery for heterogeneous 2-D and 3-D circuits.

Dr. Savidis is a recipient of the 2018 National Science Foundation Early Faculty (CAREER) Award. 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 International Verification and Security Workshop. He is a member of the Association of Computing Machinery, the IEEE Circuits and Systems Society, the IEEE Communications Society, and the IEEE Electron Devices Society. He also serves on the editorial boards of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS (VLSI), the *Microelectronics Journal*, and the *Journal of Circuits, Systems and Computers*.