JANUS-HD: Exploiting FSM Sequentiality and Synthesis Flexibility in Logic Obfuscation to Thwart SAT Attack While Offering Strong Corruption

Leon Li and Alex Orailoglu
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, 92093
xul065@ucsd.edu, alex@cs.ucsd.edu

Abstract*—Logic obfuscation has been proposed as a countermeasure towards chip counterfeiting and IP piracy by obfuscating circuit designs with a key-controlled locking mechanism. However, the extensive output corruption of early key gate based logic obfuscation techniques has exposed them to effective SAT attacks. While current SAT resilient logic obfuscation techniques succeed in undermining the attack by offering near-trivial output corruption, they do so at the expense of a drastic reduction in functional and structural protection scope. In this work, we present *JANUS-HD* based on novel insights that succeed to deliver the heretofore elusive goal of simultaneously boosting corruptibility and foiling SAT attacks. *JANUS-HD* obfuscates an FSM through diverse FF configurations for different transitions with the overall configuration setting as the obfuscation secret. A key-controlled Hamming distance comparator controls the obfuscation status at the minimized number of entrance states identified through a custom graph partitioning algorithm. Reliance on the inherent state transition patterns extends the obfuscation benefits to non-entrance states without exposing any additional key space pruning trace. We leverage the flexibility of state encoding and equivalence-based FSM transformations to generate an obfuscated netlist at low overhead using standard synthesis tools. Finally, we present a scan chain crippling mechanism that delivers unfettered scan chain access while eradicating any key trace leakage in the scan mode, thus thwarting chosen-input attacks aimed at the Hamming distance comparator. We illustrate through experiments that *JANUS-HD* delivers obfuscation scope improvements of up to 45.5x over the state-of-the-art, establishing the first cost-effective solution to offer a broad yet attack-resilient obfuscation scope against supply chain threats.

I. Introduction

Nowadays, the majority of integrated circuit (IC) design companies have transitioned to a fabless business model wherein chip fabrication is outsourced to offshore foundries. However, IC designs are suffering intellectual property (IP) piracy and overproduction threats when malicious foundries are fully cognizant of their implementation details. The financial loss due to IC counterfeiting has been estimated to exceed $100 billion each year [1], raising the imperative need to safeguard the confidentiality of chip designs.

A promising solution to supply-chain threats is logic obfuscation which embeds a key-controlled mechanism to shield the functionality and structural details of chip designs. Early logic obfuscation techniques sprinkle XOR/XNOR or multiplexer (MUX) key gates in the netlist [2], [3] so that the original functionality can be recovered only when the correct key is provided. However, these logic obfuscation attempts are vulnerable to the Boolean Satisfiability (SAT) attack which relies on oracle observations and a SAT solver to prune incorrect key values. State-of-the-art SAT-resilient logic obfuscation techniques [4]–[6] use a *point function* such as an AND tree or a Hamming distance (HD) comparator to subject any functional pruning attacks to an exponential number of attack iterations. Unfortunately, a point function inevitably results in low output corruptibility since it can only obfuscate the input patterns in its small *on-set*.

Recently, *JANUS* [7] was proposed as an obfuscation solution to breach the strict correlation between SAT-resilience and obfuscation scope in sequential designs. It applies a minimum-cut graph partitioning algorithm to the finite state machine (FSM) diagram to assign diverse FF configurations to different transitions. The configuration setting is dynamically updated by a small number of inter-partition transitions which are stored in a tamper-proof memory but otherwise held stable in their current obfuscated state for the rest of the intra-partition transitions. As a result, pruning evidence leakage for the non-key-controlled, intra-partition obfuscation scope is entirely eradicated, thus thwarting pruning attacks including SAT. However, the cost of tamper-proof memory grows in sync with the number of inter-partition transitions imposing a prohibitive cost on certain applications. A more scalable obfuscation framework is needed that welcomes the use of compact representations for overhead reduction while inheriting the expanded obfuscation scope offered by *JANUS*.

The objective of this article is to develop an obfuscation framework with a low-cost, point function based obfuscation control mechanism achieving simultaneously a large obfuscation scope and SAT-resilience. We leverage the flexible synthesis options at the register transfer level (RTL) through several coherent synthesis augmentations to address the constraint of the predetermined *on-set* patterns posed by the point function. We propose a state-based configuration update mechanism and exercise control over state encoding to construct a point function for defining the overall configuration settings. The configuration assignments are made through a custom graph partitioning algorithm that minimizes the number of entrance states which constitute the only *on-set* exposure points. The FSM diagram is then transformed at entrance states based on a connectivity analysis so that a configuration flip operation is sufficient to recover the original design. To top it off, a scan chain crippling mechanism is incorporated to prevent chosen-state querying attacks without affecting scan chain accessibility or degrading the structural testability of the circuit.

Our contributions can be summarized as follows:

978-3-9819263-6-1/DATE22/©2022 EDAA

- We present a logic obfuscation technique that unifies a point function for SAT-resilience and the inherent state transition history for an augmented obfuscation scope.
- We develop a scalable and complete FSM synthesis flow to generate a low overhead obfuscated netlist.
- We conduct an experimental analysis demonstrating the heightened obfuscation scope *JANUS-HD* delivers.

The rest of the paper is organized as follows. In Section II, we define the attack model and introduce related work. In Section III, we present the *JANUS-HD* obfuscation methodology including the synthesis methodology, the gate-level architecture, and techniques for graph partitioning and state encoding. In Section IV, we report on the overhead of *JANUS-HD* by obfuscating the ACM/SIGDA, HP, ITC'99, and ISCAS'89 benchmarks and demonstrate the heightened obfuscation scope it delivers. Finally, the paper is concluded in Section V.

II. Background

A. Attack Model

We assume that the attacker is an adversarial foundry with access to the obfuscated netlist. The attack aims at activating overproduced ICs and/or recovery of the functional design description. The attacker can query a functional oracle through the primary inputs and additionally the scan chain for sequential circuits. The attacker is presumed to be fully cognizant of the obfuscation algorithm including the instantiation parameters selected by the designer (except for the secret key).

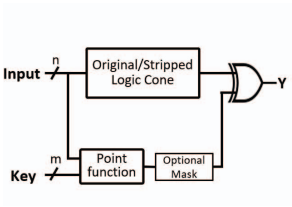
B. Definitions

The **obfuscation scope** measures the number of input patterns whose obfuscated behavior diverges from the original specification. When a function f with an n-bit input is obfuscated as f^o, the obfuscation scope under an arbitrary key k is measured as $\{\boldsymbol{L} \in \{0,1\}^n : f(\boldsymbol{L}) \neq f^o(\boldsymbol{L}, k)\}$.

The ***on-set*** of a function g with an m-bit input and a 1-bit output contains those input patterns that produce a 1 at the output, i.e., $\{\boldsymbol{L} \in \{0,1\}^m : g(\boldsymbol{L}) = 1\}$. Similarly, the ***off-set*** is defined as $\{\boldsymbol{L} \in \{0,1\}^m : g(\boldsymbol{L}) = 0\}$.

C. SAT-resilience and Obfuscation Scope Dilemma

The initial spurt of interest in logic obfuscation had focused on identifying the placement of XOR and MUX key gates to increase output corruption and preclude individual key bit recovery [3], [8]. While these early logic obfuscation techniques delivered a broad obfuscation scope, they were breached by the Boolean Satisfiability (SAT) attack which leverages a SAT solver and oracle queries to prune the key space. The tremendous success of the SAT attack has materially shifted the research focus towards SAT-resilient logic obfuscation techniques. *Anti-SAT* [4], *SARLock* [5], and *SFLL* [6] thwart the SAT attack by enforcing an exponential number of SAT iterations through the insertion of a point function. We illustrate a generalized architecture of these SAT-resilient logic obfuscation techniques in Fig. 1. The point function is responsible for either corrupting the output of the original



Input			Y	Output Y for different keys							
a	b	c	Y	Y_{k0}	Y_{k1}	Y_{k2}	Y_{k3}	Y_{k4}	Y_{k5}	Y_{k6}	Y_{k7}
0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0

Fig. 1: Generalized SAT-resilient Logic Obfuscation Netlist.

logic cone [4], [5], or recovering the output of a functionality-stripped logic cone [6]. We also illustrate in Fig. 1 a sample truth table of the obfuscated netlist under different key values for a circuit with three input bits and one output bit. It can be seen that each input observation can only prune one incorrect key value, forcing the attacker to apply all the input patterns to rule out all the incorrect keys.

Their strong SAT-resilience notwithstanding, these point function based logic obfuscation techniques suffer from an extremely low output corruptibility. In the example shown in Fig. 1, even if an incorrect key value were to be applied, there would only be one n-bit input pattern leading to any output corruption. As a result, the relative obfuscation scope with respect to the overall input space is merely $1/2^n$ for any incorrect key value. This extremely low output corruptibility fosters the approximate usage of obfuscated circuits without having to go through the trouble of key recovery, nullifying the most fundamental functional protection interests of logic obfuscation. Generalized point functions with larger *on-set*s [9], [10] allow higher output corruptibility but only at the cost of trading off SAT resilience. In fact, it has been argued that this dilemma is inherent to all combinational logic obfuscation techniques [9], [11]. Looking at the truth table that enumerates the inputs and key values, it can be observed that a high obfuscation scope requires columns to be heavily populated with erroneous entries, whereas SAT resilience requires rows to have extremely sparse erroneous entries. These two objectives are inherently contradictory and cannot be simultaneously satisfied in *any* obfuscated truth table.

The persistent dilemma faced by obfuscating combinational circuits has also fueled research in obfuscating sequential circuits. Scan chain disabling techniques [12], [13] have been proposed to strengthen combinational logic obfuscation techniques in sequential circuits by preventing the attacker from loading a chosen value into the state inputs or reading out the response from the scan chain. Yet, they turn out to remain vulnerable to sequential SAT attacks [14], [15]. Alternatively, *Interlocking* [16] proposes to divert transitions in an FSM diagram and recover them based on the key, yet it still suffers a minute obfuscation scope due to the usage of a combinational, key-controlled point function.

D. JANUS Obfuscation Using Inherent Transition Patterns

JANUS illustrates a solution to address the dilemma at root cause by using not only the key value but more importantly also the inherent FSM state transition patterns to determine

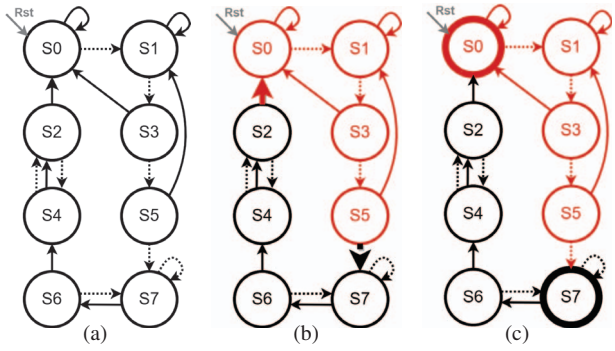


Fig. 2: (a) Original FSM. (b) *JANUS* Obfuscated FSM with bolded inter-partition transitions. (c) *JANUS-HD* Obfuscated FSM with bolded entrance states.

output corruptibility. A *JANUS* obfuscated FSM has transitions implemented with two distinct FF configurations, say the D-type and T-type for example. We show a *JANUS* obfuscated FSM in Fig. 2b where the D-type and T-type FF configurations are illustrated in black and red, respectively. Dotted edges denote transitions on a 0 input and solid edges denote transitions on a 1 input. *JANUS* stores all the transitions crossing the boundary of the two configuration groups, referred to as *inter-partition transitions*, as independent entries in a tamper-proof memory. At runtime, *JANUS* uses a register to note the active FF configuration and flips the configuration only when an inter-partition transition is executed. As a result, the obfuscation scope of *JANUS* is determined by the balance factor of partition sizes whereas the pruning attack resilience is determined solely by the frequency of inter-partition transitions. The vast majority of the obfuscation scope which is contained within partitions **reveals no information** that can be exploited for key recovery. Though *JANUS* presents a theoretical breakthrough in resolving this dilemma, its dependence on a sizable tamper-proof storage hinders its practicality. It is also impossible to specify inter-partition transitions using a point function because their predefined and near-random input conditions would not fit the orderly *on-set* patterns.

III. *JANUS-HD* Obfuscation Methodology

In *JANUS-HD*, we develop a set of coherent synthesis augmentations to deliver simultaneously the cost and security benefits of a point function in the form of an HD comparator, while retaining the obfuscation scope expansion advantage offered by *JANUS*. *JANUS-HD* not only inherits the diversified FSM synthesis technique but also leverages the flexibility of state encoding and FSM transformation to address the technical challenges introduced by the compact representation. Our fundamental insight is that the configuration assignments can be judiciously exercised so that it suffices to control the overall configuration settings based **solely on the state information** at a small number of entrance states. We then exercise control over state encoding to map these critical states to the *on-set* elements of a key-controlled HD comparator. We develop a custom graph partitioning algorithm to minimize the number

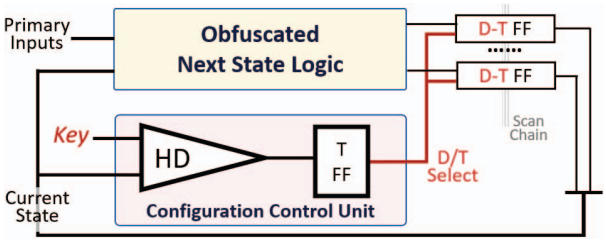


Fig. 3: Overview of *JANUS-HD* obfuscation circuit.

of entrance states while achieving a large obfuscation scope in the FSM state space. We propose a transformation technique to tackle general case scenarios wherein an entrance state receives transitions from both configuration groups; the transformed equivalent FSM ensures that entrance states receive transitions solely from the complementary configuration group enabling a toggle-based flipping of the configuration group by matching solely the state designation. Finally, the scan chain is crippled to defend against the bit-flipping attack [17] by preventing key trace leakage through scan chain access.

A. *JANUS-HD Overview*

We present an example of a *JANUS-HD* obfuscated FSM in Fig. 2c with the corresponding hardware implementation shown in Fig. 3. The FSM diagram is partitioned into two configuration groups while minimizing the number of *entrance states*, i.e. states with at least one incoming transition from the complementary configuration group. Each configuration group receives a distinct FF configuration for implementing all the outgoing transitions. The next state logic (NSL) synthesized under the configuration constraints becomes the obfuscated NSL as no single static gate-level interpretation can recover the original functionality. This obfuscated NSL is fed to reconfigurable FFs whose active configuration at runtime is determined by the configuration control unit (CCU). The CCU stores the active configuration in a T-type FF whose T-input is controlled by an HD comparator. The HD comparator compares the current state with a key value under a distance value h. When an entrance state is encountered, the HD comparator returns a 1 to flip the active configuration. Otherwise, for the rest of the internal states regardless of the configuration groups they are located in, the HD comparator returns a 0 to retain the previous configuration.

B. *Configuration Assignment through Partitioning*

To demonstrate the pruning attack resilience property of a key-controlled HD comparator, we show in Fig. 4 the number of *on-set* or *off-set* elements that are needed to recover the key value for an HD comparator with a 20-bit input and varying distance values h. It is evident that the pruning capability of *on-set* elements is drastically lower than that of *off-set* elements, especially at smaller h values. An effective logic obfuscation technique shall minimize the exposure of *on-set* elements, which in our obfuscation scheme corresponds to minimizing the number of entrance states. As no partitioning algorithm exists to the best of our knowledge with such an

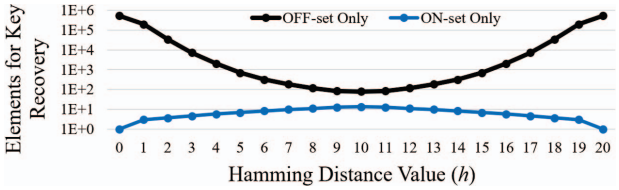


Fig. 4: The number of elements in *on-set* or *off-set* for HD key recovery for 20-bit input and various h, averaged over 1000 random querying sequences.

objective function, we develop a custom graph partitioning algorithm to identify a near-balanced partition with the fewest entrance states.

We model the FSM diagram as a directed graph $G = (V, E)$ where V represents the set of states and E represents the set of state transitions. We develop a $(2, 1+\epsilon)$-balanced graph bipartitioning algorithm that divides V into 2 blocks of vertices, V_D and V_T, each of size no more than $(1+\epsilon) \cdot \frac{|V|}{2}$. The imbalance factor $\epsilon \in [0, 1]$ defines the balance of the two partitions which determines the obfuscation scope. For instance, $\epsilon = 0$ delivers the highest obfuscation scope achievable in two partitions by ensuring that half of the patterns are obfuscated under the best static interpretation of FF configurations. In practice, the designer shall explore varying ϵ values since a small degree of imbalance often promotes a significant reduction in the number of entrance states. The algorithm minimizes the total number of entrance states $V_{Entrance} = \{v \in V_a \mid \exists u \in V_b : (u, v) \in E, a \neq b\}$ subject to the limits set by ϵ.

Algorithm 1 shows our partitioning algorithm which builds upon the Fiduccia-Mattheyses (FM) heuristic [18] by undertaking passes of vertex moves to improve the quality of a partition solution. A move in a bipartition problem is defined as moving a vertex from its current partition to the other partition. At the beginning of the algorithm, a balanced initial partition solution is randomly generated (Line 2). The algorithm iterates in multiple passes over the vertices until a local minimum solution is reached (Lines 3-16). In each pass, the algorithm makes $|V|$ exploratory moves and the vertices are locked once they have been moved (Lines 6-14). The moves are selected greedily to deliver the highest reduction in the cost function within the balance constraint (Line 7). Once all the vertices have been moved, the partition solution with the lowest cost encountered during the pass is retained by unwinding all the moves after this optimal solution point (Lines 10-13, 15). If no improvement is observed after a full pass, the algorithm returns the best encountered partition (V_D, V_T) (Lines 16-17).

Algorithm 1 (2,1+ϵ)-Balanced Graph Partitioning

Input: A directed graph $G = (V, E)$, a balance factor ϵ
Output: Partition (V_D, V_T)

1: **procedure** GRAPHPARTITION(G, ϵ)
2: $(V_D, V_T) \leftarrow$ A random initial balanced bipartition
3: **repeat**
4: $cost_{min} \leftarrow$ = Initial number of entrance states
5: $index_{min} \leftarrow 0$; $V_{free} \leftarrow V$
6: **for** $i \leftarrow 1$ to $|V|$ **do**
7: Find the best valid single move of $v \in V_{free}$
8: Update V_D and V_T to reflect the move of v
9: Remove v from V_{free}
10: Calculate $cost_{new}$ based on current entrance states
11: **if** $cost_{new} < cost_{min}$ **then**
12: $cost_{min} = cost_{new}$; $index_{min} = i$
13: **end if**
14: **end for**
15: Unwind all moves after $index_{min}$
16: **until** $index_{min} = 0$
17: **return** (V_D, V_T)
18: **end procedure**

C. FSM Transformation

Once the partition result is obtained, all the outgoing transitions from states in the V_D (V_T) will be implemented under the D-type (T-type) configuration for synthesizing the obfuscated NSL. In order to activate the obfuscated FSM, it is necessary to flip the configuration at runtime whenever an entrance state is encountered by a transition that originates from the complementary configuration group. However, it should be noted that an entrance state may have incoming transitions from not only the complementary configuration group but also its own configuration group. If one blindly flips the configuration at entrance states, the FSM would behave incorrectly thereupon.

Our novel insight in addressing this challenge is that the same state transitional behaviors can be delivered by multiple *equivalent FSMs* with varying connectivity patterns. We propose a FSM transformation technique which selectively duplicates entrance states so that every entrance state receives incoming transitions solely from the complementary configuration group. The FSM transformation for the original partitioned FSM in Fig. 2c is shown in Fig. 5a. If an entrance state S has incoming transitions from both configuration groups, it is duplicated into two states, S_A and S_B, with identical outgoing transitions to produce an equivalent FSM. The incoming transitions to S from the complementary configuration group are stitched to S_A which serves as the actual entrance state for flipping the configuration. The incoming transitions to S from its own configuration group are reconnected to S_B which does not trigger a configuration flip. As a result, it suffices to

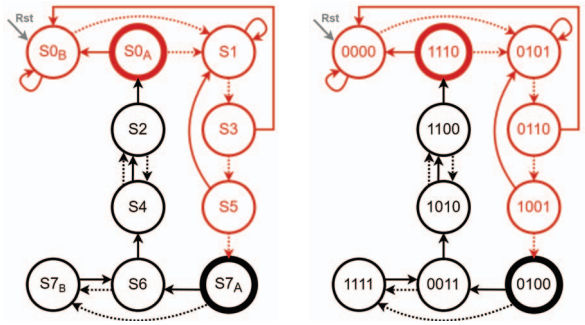


Fig. 5: (a) An equivalent FSM with duplicated entrance states. (b) An HD-driven state encoding with $key = 1100$ and $h = 1$.

always flip the FF configuration at transformed entrance states to deliver the correct FF configuration for every state when the FSM is sequentially traversed.

D. HD-Driven State Encoding

The aforementioned partitioning and transformation techniques allow the overall configuration to be controlled at a small number of entrance states so as to minimize key trace leakage. In order to enable a point function to monitor such entrance states, we assign state encodings so that the HD comparator outputs a 1 only upon encountering an entrance state. The encoding process starts by randomly selecting the correct key of the same length as the number of state elements. We select the smallest distance value h offering sufficient *on-set* elements to cover all entrance states, so as to further minimize the pruning capability of the majority of states that receive *off-set* encodings. The entrance states receive state encodings in the *on-set* of the HD comparator, whereas all the internal states including the S_B copy of the original entrance states receive state encodings in the *off-set* of the HD comparator. As an example, we illustrate a valid state encoding scheme for the obfuscated FSM in Fig. 5b when $key = 1100$ and $h = 1$.

E. Scan Chain Crippling

When a key-controlled HD comparator is presented in a combinational circuit, an attacker can break the key value by identifying a *single on-set* element and pivoting on it to query the oracle with bit-flipped input patterns [17]. The success of this attack hinges on the ability of an attacker to load a chosen state into the oracle and retrieve the output of the HD comparator. If the scan chain is disabled, the attacker would not enjoy such a freedom and the well-known intractability of sequential automatic test pattern generation (ATPG) [19] would pose a deterrent against any attempts to identify an input sequence to a chosen state. This disablement ensures that (1) the key value remains secure even if a few *on-set* elements in the form of entrance states have been identified through state traversal and (2) no attack strategy can improve on the general pruning attack using sequentially visited state patterns.

However, an outright disabling of the scan chain, say with fuse technology [20], hinders defect diagnosis for malfunctioning chips returned by customers. We propose a lightweight scan chain crippling technique to maintain full accessibility to the scan chain for testing purposes yet bypassing the HD comparator in the scan mode. In a *JANUS-HD* obfuscated netlist, the enable line of the T-FF in the CCU is determined by a tiny controller that monitors whether the Scan Enable (SE) signal has been turned on since power-up. Once the scan mode has been entered, the CCU is effectively disconnected from the next state logic by overwriting the D/T select signal with a constant *dummy* configuration. As a result, querying the oracle with the scan chain would only reveal the non-secret behavior of the obfuscated NSL and the dummy configuration, with no information leakage as to the key value. Furthermore, the bypassing technique has no impact on the coverage and diagnosability of structural tests because the scan chain remains fully accessible and any faulty observations can be interpreted based on the dummy, static FF configuration.

	Circuits	FFs	Inputs	States	Transitions
ACM/SIGDA	bbara_bbtas	7	4	128	852
	kirkman	4	12	16	370
	sand	5	11	32	184
	scf	7	27	121	165
	s298	8	3	218	1096
	s420	5	19	18	137
	s820	5	18	25	131
	s1488	6	8	48	134
	tbk	5	6	32	1569
HP	alex1	6	5	42	130
	intel_edge.dummy	5	3	28	64
	isend	6	7	40	93
	vbe4a	6	6	58	182
	vmebus.master.m	5	11	32	437
	pe-rcv-ifc.fc	6	8	46	108
	pe-send-ifc.fc	7	8	70	184
RE	b06	9	4	256	960
	s444	21	3	8864	70912

TABLE I: Specifications of benchmarks.

IV. Experimental Results

In this section, we demonstrate the effectiveness of *JANUS-HD* by obfuscating the 16 largest FSM circuits from the ACM/SIGDA (MCNC) [21] and HP benchmarks [22]. Due to the effective shortage in larger RTL benchmarks, we additionally reverse engineered two large FSMs from ITC'99 [23] and ISCAS'89 [24] using a custom static FSM extraction tool. We show the specifications of the circuits we tested in Table I.

We run the partitioning algorithm with $\epsilon = 0.2$ to deliver a near-balanced partition solution and show the results in Table II. We assume *SFLL-hd* and an HD-based implementation of *Interlocking* as the baseline attack resilient obfuscation solutions which can obfuscate as many states as the *on-set* size. For the same number of *on-set* elements (and thus the same pruning attack resilience), we illustrate the improvement on the obfuscation scope as the *expansion ratio* which measures the ratio between the smaller partition and the number of entrance states. Compared to the baseline solutions which use a solely key-controlled corruption mechanism, *JANUS-HD* delivers **a maximum of 45.5x and an average of 8.9x expansion of the obfuscation scope** at the same pruning attack resilience. Furthermore, compared to an unprotected scan chain, the integrated scan chain crippling technique allows a higher number of *on-set* elements to be revealed to an attacker without endangering the key value, with the benefits ranging from one ($k = 20$) to two orders ($k = 256$) of magnitude.

We demonstrate the implementation overhead of *JANUS* in Table II using the Synopsys Design Compiler with the TSMC 65nm Standard Cell Library. On average, the area, delay, and power overheads for the obfuscated NSL and reconfigurable FF are **5.7%, 9.8%, and 7.0%**, respectively. The overhead is generally minimal because the original NSL and the obfuscated NSL both specify "random" logic with similar implementation complexity. The HD comparator incurs a fixed

Circuits	Partitioning Result	**Exp. Ratio**	Area OH	Delay OH	Power OH
bbara.	{77, 51}:9	**5.7**	4.5%	13.2%	5.2%
kirkman	{8, 8 }:2	**4.0**	12.6%	22.6%	13.9%
sand	{19, 13}:5	**2.6**	-2.1%	-5.2%	2.6%
scf	{64, 57}:2	**28.5**	5.6%	1.8%	5.6%
s298	{128, 98}:12	**8.2**	1.7%	17.1%	5.7%
s420	{9,9}:3	**3.0**	19.6%	6.5%	22.3%
s820	{13,12}:5	**2.4**	12.5%	12.0%	13.6%
s1488	{24,24}:5	**4.8**	6.1%	25.1%	6.9%
tbk	{18, 14}:16	**0.9**	-0.3%	-3.1%	0.0%
alex1	{24, 18}:5	**3.6**	5.9%	7.5%	6.3%
intel.	{14, 14}:4	**3.5**	8.1%	13.8%	12.4%
isend	{20, 20}:2	**10.0**	7.0%	12.2%	6.9%
vbe4a	{29, 29}:9	**3.2**	3.7%	19.2%	5.9%
vmebus.	{16, 16}:3	**5.3**	3.1%	-2.7%	2.1%
pe-rcv.	{23, 23}:3	**7.7**	6.5%	15.7%	8.4%
pe-send.	{36, 34}:3	**11.3**	3.3%	11.0%	4.1%
b06	{128, 128}:13	**9.8**	4.3%	7.5%	-2.2%
s444	{5318, 3547}:78	**45.5**	0.4%	1.4%	6.7%

TABLE II: Partitioning results under $\epsilon = 0.2$ and overheads. A partition is represented as $\{|V_D|, |V_T|\} : |V_{Entrance}|$. The expansion ratio is defined as $Min(|V_D|, |V_T|)/|V_{Entrance}|$.

area overhead which wanes to less than 1% of that of the obfuscated NSL for s444 with thousands of states and thus reduces to insignificance for any realistic application scenario. The integrated scan chain crippling technique delivers the same security benefit as other scan chain disabling techniques to prevent the bit-flipping attack yet only at negligible costs.

V. Conclusion

In this article, we develop an FSM obfuscation methodology through a set of coherent synthesis augmentations to achieve simultaneously high output corruptibility and pruning attack resilience using a low-overhead key-controlled HD comparator. We exploit the inherent state transition patterns of an FSM as the main lever for controlling corruptibility through a novel synthesis methodology that implements state transitions under diverse FF configurations. The configuration assignments are made through a custom graph partitioning algorithm which effectively minimizes the *on-set* exposure of the point function. We generously leverage the flexibility inherent to RTL specification and synthesis to deliver a point function for controlling the overall configuration settings with minimum key trace leakage, through an HD-driven state encoding technique as well as an FSM transformation technique. Finally, the sequential access to the oracle is enforced through a lightweight scan chain protection technique to thwart general chosen-input pruning attacks. We confirm through experiments that *JANUS-HD* delivers an obfuscation scope orders of magnitude higher than existing attack resilient work at a marginal area overhead and performance impact.

References

[1] IHS iSuppli. Top 5 most counterfeited parts represent a $169 billion potential challenge for global semiconductor market. 2012.

[2] Jarrod A Roy, Farinaz Koushanfar, and Igor L Markov. EPIC: Ending piracy of integrated circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1069–1074. IEEE, 2008.

[3] Yu-Wei Lee and Nur A Touba. Improving logic obfuscation via logic cone analysis. In *16th Latin-American Test Symposium (LATS)*, pages 1–6. IEEE, 2015.

[4] Yang Xie and Ankur Srivastava. Mitigating SAT attack on logic locking. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 127–146. Springer, 2016.

[5] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan Rajendran, and Ozgur Sinanoglu. SARLock: SAT attack resistant logic locking. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 236–241. IEEE, 2016.

[6] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan Rajendran, and Ozgur Sinanoglu. Provably-secure logic locking: From theory to practice. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1601–1618, 2017.

[7] Leon Li and Alex Orailoglu. JANUS: Boosting logic obfuscation scope through reconfigurable FSM synthesis. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2021.

[8] Jeyavijayan Rajendran, Huan Zhang, Chi Zhang, Garrett S Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Fault analysis-based logic encryption. *IEEE Transactions on Computers*, 64(2):410–424, 2013.

[9] Kaveh Shamsi, Travis Meade, Meng Li, David Z Pan, and Yier Jin. On the approximation resiliency of logic locking and ic camouflaging schemes. *IEEE Transactions on Information Forensics and Security*, 14(2):347–359, 2018.

[10] Jingbo Zhou and Xinmiao Zhang. Generalized SAT-attack-resistant logic locking. *IEEE Transactions on Information Forensics and Security*, 16:2581–2592, 2021.

[11] Hai Zhou. A humble theory and application for logic encryption. *IACR Cryptol. ePrint Arch.*, 2017:696, 2017.

[12] Rajit Karmakar, Santanu Chatopadhyay, and Rohit Kapur. Encrypt flip-flop: A novel logic encryption technique for sequential circuits. *arXiv preprint arXiv:1801.04961*, 2018.

[13] Ujjwal Guin, Ziqi Zhou, and Adit Singh. Robust design-for-security architecture for enabling trust in IC manufacturing and test. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(5):818–830, 2018.

[14] Mohamed El Massad, Siddharth Garg, and Mahesh Tripunitara. Reverse engineering camouflaged sequential circuits without scan access. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 33–40. IEEE, 2017.

[15] Kaveh Shamsi, Meng Li, David Z Pan, and Yier Jin. KC2: Key-condition crunching for fast sequential circuit deobfuscation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 534–539. IEEE, 2019.

[16] Avinash R Desai, Michael S Hsiao, Chao Wang, Leyla Nazhandali, and Simin Hall. Interlocking obfuscation for anti-tamper hardware. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, pages 1–4, 2013.

[17] Fangfei Yang, Ming Tang, and Ozgur Sinanoglu. Stripped functionality logic locking with Hamming distance-based restore unit (SFLL-hd)–unlocked. *IEEE Transactions on Information Forensics and Security*, 14(10):2778–2786, 2019.

[18] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference (DAC)*, pages 175–181. IEEE, 1982.

[19] Kwang-Ting Cheng. Gate-level test generation for sequential circuits. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(4):405–442, 1996.

[20] Min Shi, Jin He, Lining Zhang, Chenyue Ma, Xingye Zhou, Haijun Lou, Hao Zhuang, Ruonan Wang, Yongliang Li, Yong Ma, Wen Wu, Wenping Wang, and Mansun Chan. Zero-mask contact fuse for one-time-programmable memory in standard cmos processes. *IEEE Electron Device Letters*, 32(7):955–957, 2011.

[21] Saeyang Yang. *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.

[22] Timothy Kam, Tiziano Villa, Robert Brayton, and Alberto Sangiovanni-Vincentelli. A fully implicit algorithm for exact state minimization. In *31st Design Automation Conference*, pages 684–690. IEEE, 1994.

[23] Fulvio Corno, Matteo Sonza Reorda, and Giovanni Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of computers*, 17(3):44–53, 2000.

[24] Christoph Albrecht. IWLS 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*, 2005.