WaLo: Security Primitive Generator for RT-Level Logic Locking and Watermarking

Jun Kuai*, Jiaji He†, Haocheng Ma*, Yiqiang Zhao*,Yumin Hou‡ and Yier Jin‡
*School of Microelectronics, Tianjin University
†Institute of Microelectronics, Tsinghua University
‡Department of Electrical and Computer Engineering, University of Florida
kuaijun@tju.edu.cn, jiaji_he@mail.tsinghua.edu.cn, hc_ma@tju.edu.cn, yq_zhao@tju.edu.cn, hou.yumin@ufl.edu, yier.jin@ece.ufl.edu

***Abstract*—Various hardware security solutions have been developed recently to help counter hardware level attacks such as hardware Trojan, integrated circuit (IC) counterfeiting and intellectual property (IP) clone/piracy. However, existing solutions often provide specific types of protections. While these solutions achieve great success in preventing even advanced hardware attacks, the compatibility of among these hardware security methods are rarely discussed. The inconsistency hampers with the development of a comprehensive solution for hardware IC and IP from various attacks. In this paper, we develop a security primitive generator to help solve the compatibility issue among different protection techniques. Specifically, we focus on two modern IC/IP protection methods, logic locking and watermarking. A combined locking and watermarking technique is developed based on enhanced finite state machines (FSMs). The security primitive generator will take user-specified constraints and automatically generate an FSM module to perform both logic locking and watermarking. The generated FSM can be integrated into any designs for protection. Our experimental results show that the generator can facilitate circuit protection and provide the flexibility for users to achieve a better tradeoff between security levels and design overheads.**

I. Introduction

The high cost of maintaining advanced foundries and the globalization of the semiconductor industry divide the front-end fabless design houses and back-end fabrication and testing companies. This trend also breeds the integrated circuit (IC) security threats such as IC counterfeiting, intellectual property (IP) theft, and IC overproduction [1]. To address these challenges, various design-for-trust (DfTr) solutions [2], [3] have been developed, e.g., logic locking, design obfuscation, watermarking, camouflaging, hardware root-of-trust, etc. While these solutions are proved effective in countering existing security concerns, new attacks emerge, almost simultaneously with the defense solutions. Taking the logic locking as an example [4], along with a series of novel locking/obfuscation techniques, there exist deobfuscation methods including satisfiability checking (SAT) based attack [5], logic cone analysis based attack [6], hill-climbing attack [7], approximate attacks [8], etc. One key issue that fuels this arms race is the single-purpose of these developed countermeasures. That is, existing hardware security solutions often focus on one specific protection scheme against one potential attack. Attackers only need to break one-level of defense in order to compromise the whole design. To overcome such limitations, the concept of defense-in-depth has been proposed recently. Instead of using one single-purpose protection, multiple solutions will be combined for more resilient security schemes. However, the compatibility among existing methods are rarely considered. The inconsistency hampers with the development of a comprehensive solution for circuit protections.

978-1-7281-8952-9/20/$31.00 ©2020 IEEE

In this paper, we are developing a security primitive generator to help solve the compatibility issue among different protection techniques. Specifically, we focus on two modern IC/IP protection methods, logic locking and watermarking. A combined locking and watermarking technique is developed based on finite state machines (FSMs). The security primitive generator will take user-specified constraints and automatically generate an FSM module to perform both logic locking and watermarking with a optimized overhead and security compared to two independent security schemes. The generated FSM can be integrated into any designs for protection.

In the proposed locking-watermark-combined security primitive generation framework, watermark's state is reused by the locking scheme, resulting in a longer unlocking path. Further, the existing FSM-based locking scheme is also enhanced by the insertion of pseudo-FSMs, unlocking chain-network, blackhole-FSMs and transitions among them. The developed locking scheme is resilient to brute-force attacks and other more advanced attacks, i.e., FSM recovery through netlist reverse engineering [9], [10]. To automate the process, we developed a generator, named WaLo: Security Primitive Generator for RT-Level Logic Locking and Watermarking. Compared to gate-level security solutions, the proposed method can fully leverage the commercial EDA toolchain.

The overall contributions of our paper are listed as follows.

- We propose a design-agnostic solution for automatic security primitive generation. The generated security primitive supports both watermarking and logic locking.
- Different from netlist and layout level solutions, we provide RTL solutions which can be seamlessly embedded into the existing design flow.
- Security analysis shows that the developed scheme can achieve exponential security levels against watermark extraction, SAT-based deobfuscation attacks and even the latest FSM reconstruction attacks.

II. Related Work

A. Watermarking

Watermarking techniques have been widely used to protect the rights of the IP owners through authentication. The authors in [11] insert signatures in unused LUTs and create functional differences between instances to protect FPGA design. However, it is difficult to automate the watermarking process. The side-channel-based watermarking relies on side-channel information to embed the watermark. The authors in [12] add power pattern generators to circuit and use them to create power consumption differences which serve as the watermark. Similarly, the authors in [13] embed watermark by creating specific electromagnetic (EM) information.

The FSM-based watermarking approach tries to embed the watermark to FSM's transitions and then apply a special excitation sequence for verification. Approaches in [14], [15] manipulate the State Transition Graph (STG) of the original design to create a watermark. Cui et al. [16] proposed a novel FSM watermarking scheme by making the authorship information a non-redundant property of the FSM at the behavioral level. This technique makes the watermark randomly dispersed and concealed in the existing FSM transitions. The authors in [17] tried to use both existing and unused FSM transitions to insert the information encrypted by AES and MD5.

B. Logic Locking and Obfuscation

Various logic locking and obfuscation approaches have been proposed, aiming to obfuscate the high-level information of the design. These existing solutions can be roughly divided into combinational locking and FSM-based sequential locking. The authors in [18] proposed key-controlled reconfigurable logic blocks to obfuscate all paths from inputs to outputs. Then in [19], the authors proposed an approach of AND/OR gates insertion to hide signals with low controllability. Combinational logic locking schemes were initially found vulnerable to Boolean SAT attacks [20], [21]. New schemes are then proposed to resist SAT attacks by increasing the minimum number of differential input patterns (DIPs) required to find a correct key, e.g., using "point-functions" or comparator logic to decrease the effectiveness of each DIP [22], [23].

FSM-based locking techniques are more resilient to SAT-style attacks. For example, the HARPOON scheme [24] tries to augment an FSM with a series of states that form a preceding locking mode. Only with knowledge of correct excitation sequence, users can access the circuit's normal functional states. The authors in [25] proposed dynamic-state-deflection to prevent the unauthorized overwriting of the FSM state memory register. But FSM-based locking techniques suffer from FSM reverse engineering attacks and the unlocking sequence may be recovered [9], [10]. Another limitation is that all locking schemes apply in gate-level netlist. There lack RTL solutions for watermarking and logic locking.

III. Threat Model

We follow a similar threat model to previous logic locking and watermarking schemes [16], [26]. The adversary may not access the protected RTL code. Instead, the adversary have the access to the flattened gate-level netlist of design either within an untrusted foundry or through reverse engineering to a fabricated chip [27]. This netlist may not contain high level design information such as hierarchies of design, the design's synthesis constraint, nets and instances' names. Further, we assume that some adversaries may have the access to a working "oracle" chip, i.e., an unlocked and functional chip.

The adversary's goal is to identify the design's watermark and locking key. In this way, the adversary can remove or maliciously change the inserted watermarks so that an IP piracy attack becomes possible. With the unlocking key, the adversary can perform IP clone for unauthorized redistribution or overproduction of valuable IPs.

Algorithm 1: Generation of the watermark-FSM

Input: copyright-information CI, configuration-file CF.
Output: watermark-FSM $WaFSM$.

create WaFSM as empty graph G(N,E,V)=G(Nextstate,Excitation,Output)
$aeskey \leftarrow GetKey(CF)$
$signature \leftarrow HashFunction(AES(CI, aeskey))$
$expandratio \leftarrow GetExpandRatio(CF)$
while *signature is not empty* **do**
  $tempslice \leftarrow CopyFirstSlice(signature)$
  $signature \leftarrow RemoveFirstSlice(signature)$
  $G(N, E, V) \leftarrow G(N \cup randomstate, E \cup randomexcitation, O \cup tempslice)$

IV. RTL Locking and Watermarking

A. Framework Overview

In this section, we will introduce the details of the proposed WaLo framework for RTL protection. Figure 1 shows the overall working procedure of the WaLo framework where the locking and watermarking schemes are integrated into one FSM-based security primitive. The whole process is automated and the generated security primitive can be inserted into any designs for circuit protection (with little modifications to the original design such as wire connections to the security primitive). Provided with a user-specified configuration, as shown in Figure 1, WaLo will create a watermarked FSM first which is referred to as watermark-FSM in the paper. Then it will generate pseudo-FSMs and blackhole groups for FSM-based locking. These types of FSMs are merged by a chain network, establishing the superiority of overhead and security compared to two independent security schemes. As we will demonstrate later, the generated security primitive for watermarking and logic locking is resilient to various attacks including brute-force attacks, SAT-based attacks and FSM reverse engineering attacks. The generated security primitives can also behave watermarking and logic locking independently, i.e., the watermarking verification process will not leak information about circuit unlocking, and vice versa.

B. Watermark-FSM Generation

WaLo will fist create the FSM for watermarking. The basic procedure of the FSM generation process for watermarking is shown in Algorithm 1. To obfuscate the semantic information, a cryptographic algorithm, e.g., AES cipher, is used to encrypt

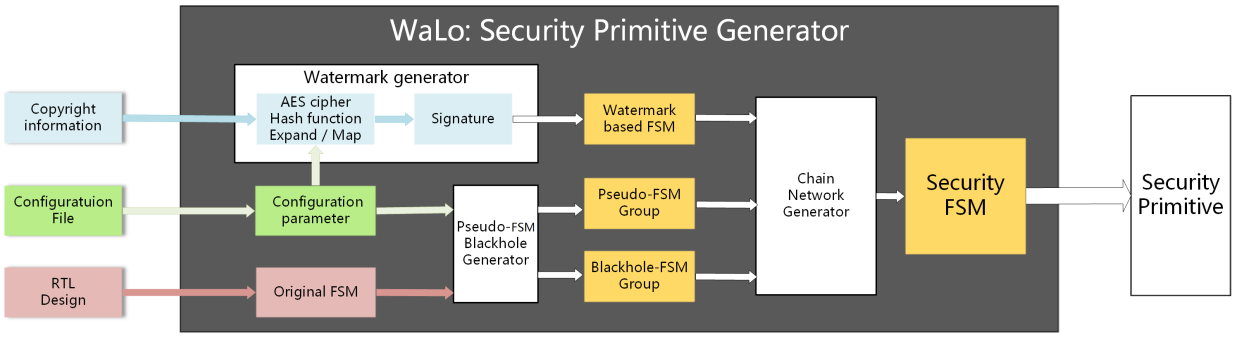

Fig. 1. The working procedure of the proposed WaLo security primitive generator.

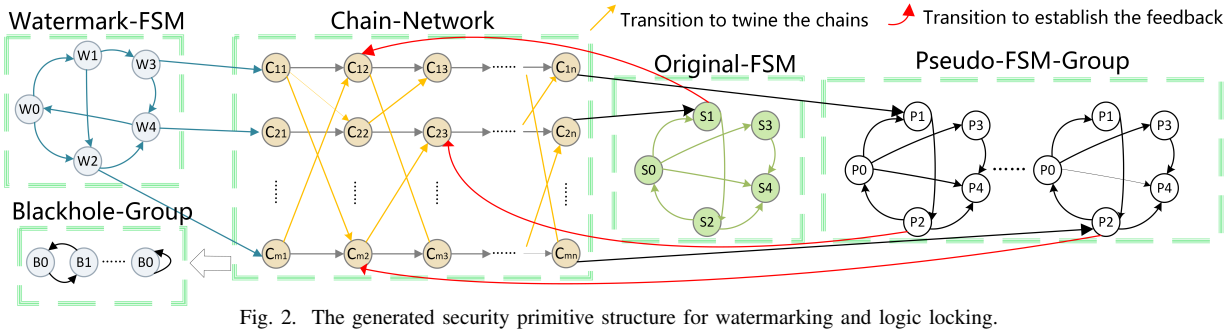

Fig. 2. The generated security primitive structure for watermarking and logic locking.

the user-defined copyright information. An optional hash function can be used to help compress the signature. The hashed digest (or the encrypted ciphertext) is used as the watermark signature for insertion.

After that, WaLo applies randomized expansion and mapping based on the provided signature, which result in information redundancy and disordering, respectively. To reduce the overhead, the WaLo will create the watermark-FSM with minimum size, $Size_{FSM}$ in Equation (1).

$$Size_{FSM} = \frac{Length_{sig}}{Width_{out} \times 2^{Width_{in}}} \tag{1}$$

where the $Width_{in}$, $Width_{out}$ and $Length_{sig}$ denote widths of FSM's input, output and the signature length, respectively.

C. Locking FSM and Chain Network Generation

The pseudo-FSM is an FSM that mimics the behavior of a normal functional FSM, which we refer to as the original-FSM in the paper. For example, there often exist bidirectional transition paths between arbitrary two states. By creating a group of pseudo-FSMs, we can increase the degree of obfuscation against FSM identification attacks. In the case that the design under protection is already available, we will extract the functional FSM to serve as a reference for pseudo-FSMs. In WaLo, we leverage the traversal-path-FSM generation approach to create pseudo-FSMs [28]. The unidirectional state chain is initialized and two intermediate states are selected at random. To ensure the path traversability, transitions loops are built by connecting the head and tail states with these two intermediate states. Another protection mechanism is called blackhole. It denotes a group of FSM states with no exits. Through creating a group of blackholes, the attack may not access the original-FSM by random input sequences.

WaLo then initializes the chain network which behaves as multiple unidirectional state chains. The watermark-FSM, pseudo-FSMs and blackhole groups are interconnected using the chain network. The constructed chain network starts with the watermark-FSM and terminates at either the original-FSM or any of the pseudo-FSM. In this way, WaLo merges watermarking and logic locking schemes.

For each state inside the watermark-FSM, WaLo calculates the length of its transition path. Those states with the longest transition length are selected as the beginning points. Note that the states of the watermark-FSM are reused as part of the locking FSM, a practice that helps reduce the overhead and increase the traverse space against brute-force attacks. Further, the watermarking verification process can be performed without disclosing the unlocking sequences. The blackhole group is then added into idle transitions of each chain state rather than watermark-FSM states. This step aims to achieve a high probability that FSM operations will be suspended if an incorrect unlocking sequence is provided. Adding blackhole states only to the chain network makes sure most transitions to blackhole are from states in the unlocking path, resulting to a lower overhead. Further, the unidirectional state chains are interconnected with each other for network structure. Through this, the attack complexity increases exponentially against accessibility analysis.

WaLo builds the feedback path starting from the endpoints, i.e., the pseudo-FSM and original-FSM, to the chain network. The feedback paths expand the path traversability, invalidating the minimum-blackhole attack (see Section IV-D).

D. Security Analysis

In this section, we will evaluate the security of the WaLo scheme. Note that we assume that there are secure storage for

Algorithm 2: Security primitive generation using the WaLo framework

Input: configuration-file CF, original-design $OriD$, Watermark-FSM $WaFsm$.
Output: WaLo framework FSM $WaLoFSM$.

OriFSM ← ExtractFSM(origin-design)
(pseudo-num,black-num) ← GetGroupSet(CF)
(PseFSM,BlaFSM) ← GenerateFsmGroup(pseudo-num,black-num)
repeat
  create chain topology G(preState,postState,Excitation,Output)
  State1 ← one of farthest states to initial state in WaFSM
  State2 ← GetState(PseFSM or OriFSM)
  G ← G∪(State1,GetStartState(G),-,-)
  G ← G∪(GetEndState(G),State2,-,-)
  ChainFSM∪G(preS,postS,E,O)
until *times reach pseudo-num+1*;
repeat
  ChainFSM ← ChainFSM∪ (GetState(ChainFSM),GetState(BlaFSM),-,-)
until *Probability to Blackhole satisfaction*;
repeat
  ChainFSM ← ChainFSM∪ (GetState(ChainFSM),GetState(ChainFSM),-,-)
until *Complexity satisfaction*;
WaLoFSM ← OriFSM∪ChainFSM∪PseFSM∪BlaFSM∪WaFsm
Function $GenerateFsmGroup$ (*pse-num,bla-num*) **:**
  repeat
    (statenum,transitionnum) ← GetPseudofsmSet(CF)
    create chain topology G(preS,postS,E,O)
    G ← G∪(GetState(G),GetStartState(G),-,-)
    G ← G∪(GetEndState(G),GetState(G),-,-)
    repeat
      G ← G∪(GetState(G),GetState(G),-,-)
    until *Similarity satisfaction*;
    PseFSM ← PseFSM∪$G(preS, postS, E, O)$
  until *times reach pse-num*;
  repeat
    create ring topology G(preS,postS,E,O) with 2 states
    BlaFSM ← BlaFSM∪G(preS,postS,E,O)
  until *times reach bla-num*;
  return pseudofsmgroup $PseFSM$, blackholegroup $BlaFSM$

unlocking keys so we will not discuss the key storage security.

According to our threat model, we consider three adversaries with different resources and capabilities. The **Type I Adversary** treats a gate-level netlist as a blackbox and tries to find the watermark or to unlock the design by applying random input sequences. The **Type II Adversary** has the ability to rebuild the entire FSM of watermarked and locked design using advanced netlist reverse engineering tools [9]. By observing FSM's states and transitions, the adversary tries to identify original-FSM and extract the watermark. Note that the previous two types of adversaries do not have access to an "oracle" chip. The **Type III Adversary** will also use FSM construction tools to rebuild the FSM. Further, the adversary has an "oracle" chip which is unlocked and functional.

Brute-Force Attack (Type I Adversary). In this attack, the adversary would apply random input sequences to explore

TABLE I
WALO-GENERATED SECURITY PRIMITIVE AREA OVERHEADS.

input /output	chain length	pseudo FSMs	prob to blackhole	#state	#trans	#reg	#gate
4 / 4	4	3	0.2	65	420	15	2155
6 / 6	4	3	0.2	40	601	17	1717
8 / 8	4	3	0.2	31	1022	19	2970
10 / 10	4	3	0.2	32	3466	21	9896
4 / 4	6	3	0.2	74	469	15	2347
4 / 4	8	3	0.2	82	517	15	2528
4 / 4	10	3	0.2	92	567	15	2802
4 / 4	12	3	0.2	98	613	15	3043
4 / 4	4	3	0.3	66	433	15	2260
4 / 4	4	3	0.4	65	456	15	2317
4 / 4	4	3	0.5	67	470	15	2149

FSM states for watermark extraction or design unlocking. Supported by the WaLo framework, the length of locking key depends on the watermark and chain-network's structures and the width of FSM's input. With the input width *I*, the number of watermark-FSM's state *Ws*, the number of transitions to the furthest state in the watermark-FSM *Wt*, and the number of every chain's state *Cs*, the length of the unlocking path is *(Wt+Cs)* for *I*-bit sequences. Note that our scheme support multiple unlocking paths. With the total number of unlocking paths being *u*, the probability to randomly achieve one correct unlocking sequence is calculated as follows.

$$Pr = \frac{u}{2^{I\cdot(Wt+Cs)}} \tag{2}$$

Further, due to the inserted blackhole states, the adversary need to reset the design whenever the FSM is stuck at blackholes. Since the adversary may not know the length of unlocking sequence, the attack complexity is further increased when all possible lengths are tested.

For watermark extraction, the watermark-FSM has a two-way interconnected network topology. Therefore, the space of FSM's output is exponential to *Ws* and the number of total transitions in the watermark-FSM. The original copyright information is encrypted and hashed, so the adversary cannot tell whether an extracted watermark is corrected or not.

Accessibility-Analysis Attack (Type II Adversary). A more experienced adversary may use latest reverse engineering tools to extract the FSM structure, e.g., rebuilding the design's FSM using structural analysis methods [9]. The adversary will then try to find the original-FSM from the reconstructed FSM (see Figure 2). In fact, one key limitation of previous FSM-based locking solution, e.g., [24], is that the probability to transit within locking states is low but the probability of state transitions in the original-FSM is much higher. The adversary can rely on this observation to differentiate locking states from functional states and then recover the unlocking sequence [10].

The proposed WaLo framework tries to overcome this limitation. Besides the chain network and blackholes, WaLo also creates pseudo-FSMs with high internal accessibility, a property similar to the original-FSM. Without knowing the correct outputs, the adversary cannot differentiate the original-FSM from pseudo-FSMs through the accessibility analysis. Further, the use of a chain-network, instead of simple chains,

will further complicate the structural analysis. Assuming that the number of chains is *m*, every chain has *Cs* states, and the average number of transition to every state in chain is *l*, the total number of chains, denoted as *N*, in the chain-network can be calculated as follows.

$$N = m \cdot (l)^{Cs} \tag{3}$$

Note that the large amount of chains only slightly impacts the resistance to brute-force attacks because $Pr \times N$ is still a very small number.

Minimum-Blackhole Attack (Type II Adversary). As we discussed earlier, the original-FSM often has the property of high path traversability and does not contain blackhole states. In previous FSM-based locking schemes, the original-FSM itself is a blackhole group, i.e., the FSM cannot switch back to the locked mode after being unlocked. This design consideration prevents the circuit being locked accidentally but also make the original-FSM be identified easier from the recovered FSM. The minimum-blackhole attack is to check all blackhole-style FSMs and identify the original-FSM. The attack complexity is linear to the total amount of the blackhole-style FSMs (see Figure 3, bottom).

To counter this attack, WaLo breaks the assumption that the original-FSM only has inbound transitions. By using undefined transitions in the original-FSM, WaLo inserts transition paths from the original-FSM states back to the chain network. We argue that any normal computations with legal inputs will not cause circuit locking accidentally.

Oracle-Guided Attack (Type III Adversary). In the case that an adversary can reconstruct the whole FSM accurately and have the access to an oracle circuit, an oracle-guided attack can be performed. This will decrease the attack complexity of Accessibility-Analysis and Minimum-Blackhole attacks from exponential to linear. The adversary only needs to compare the outputs of each candidate state with the oracle outputs to differentiate original-FSM from pseudo-FSMs. Note that this attack only applies to locking scheme but does not impact the security of the watermarking scheme.

To counter this attack, a simple solution is to increase the number of pseudo-FSMs and add more transitions from pseudo-FSMs and orignal-FSM to the chain network. Since WaLo works at RTL, it is easy to achieve the tradeoff between security and overhead. With a large number of registers, it becomes difficult to analyze and rebuild the whole FSM. Another method is to design pseudo-FSMs which has close (but not the same) behaviors to the original-FSM. Note that the solution requires the understanding of the original-FSM and will be investigated in our future work.

V. Experimental Results

To evaluate the overheads of the generated security primitives, we use a standard ASIC design flow to synthesize WaLo generated primitives under different configurations. The overhead is measured by the number of combinational gates and flip-flops, #gate and #reg in Table I, respectively. For security analysis, we apply the *REFSM* tool described in [9] to reverse engineer the whole FSM from the synthesized netlist. We then analyze the resilience against the first three attacks mentioned in Section IV-D.

A. Overhead Analysis

Based on different user configurations, WaLo generated a series of security primitives with different complexities. All security primitives provide watermarking and logic locking protections to the target design. These configurations and overheads are listed in Table I.

The primitive overhead is linearly proportional to WaLo's parameters except for the input and output widths since possible transitions among states are exponentially linked to input and output widths. To maximize the security with low overhead, the designer may increase the length of the locking chains and insert a few pseudo-FSMs with a similar size to the original-FSM. In general, the designer should keep the width of input and output relatively small. However, a width less than 3 may impact the watermarking scheme security.

B. Security Evaluation

For the security analysis purpose, we choose a security primitive generated with the following parameters, 4-bit input and output, 2 pseudo-FSMs each with 5 states, locking chains with 3 states, 2 blackholes, and 20% chance to transit into blackhole state for every transition in the chain-network. The generated FSM have around 50 states and 350 transitions (see FSM-A in Figure 3).

Brute-Force Attack and Accessibility-Analysis Attack. For the brute-force attack, the shortest path to unlock the design has 9 transitions, i.e., there are $2^{4\times9}$ possible input sequences and 96 unlocking paths. As a result, the maximum *Pr*, the probability to randomly achieve one correct unlocking sequence, is 1.4×10^{-9}. For the accessibility-analysis attack, the average number of transitions to every state in chain network is 2. So *N*, the total number of unlocking chains, is 24.

Note that by adding more states to the chain network, *Pr* will decrease exponentially and *N* will increase exponentially. For example, by adding 10 states to each of 3 chains and increasing transitions to every state in chain network from 2 to 4, the *Pr* will decrease from 10^{-9} to 10^{-21}, and *N* will increase from 24 to 2×10^8.

Minimum-Blackhole Attack. To evaluate the resistance to the Minimum-Blackhole attack, we generate another security primitive (see FSM-B in Figure 3) which shares a similar structure but without transitions from the original-FSM and pseudo-FSMs to the chain network. As shown in Figure 3, the Minimum-Blackhole attack may find 3 minimum-blackholes in FSM-B and one of them is the original-FSM. The adversary can explore all 3 FSMs quickly to identify the original-FSM. The adversary may only find a big minimum-blackhole which includes the original-FSM, pseudo-FSMs, blackhole states and the chain network. And due to the exponential complexity of chain network, the time cost to analyze this FSM is much higher than analyzing the FSM-B. Transitions from the original-FSM and pseudo-FSMs to the chain-network will improve the resilience against attacks. The overall overhead is

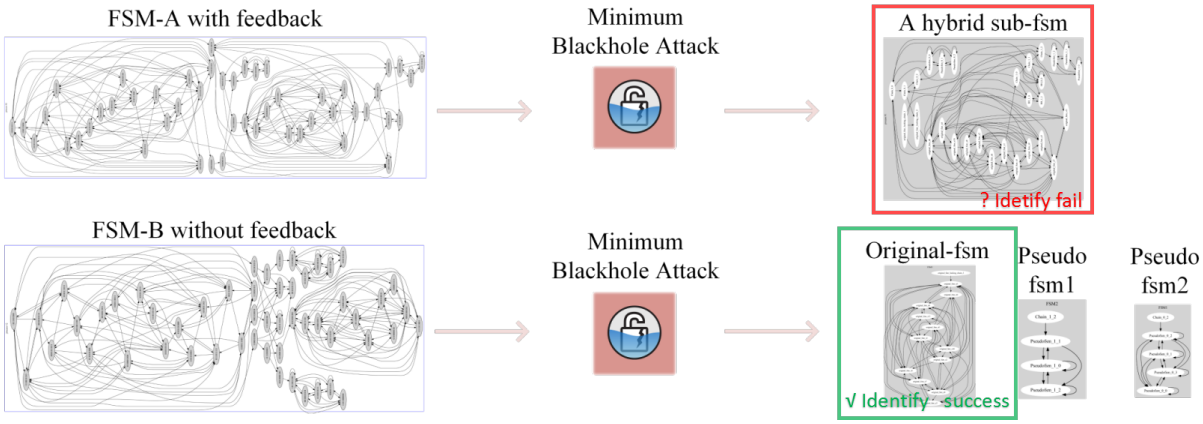

Fig. 3. Security analysis of minimum-blackhole attack.

linearly proportional to WaLo's settings such as input/output width, the count of pseudo-FSMs, etc. while the security level increases exponentially.

VI. Conclusions

In this paper, we proposed and evaluated WaLo, an RTL security primitive generator combining Locking and watermarking schemes. The automatically generated FSM-based security primitive can be integrated into any designs for protection, with high resilience to the brute-force attack and other more powerful attacks relying on netlist reverse engineering.

Acknowledgements

This work is partially supported by the National Science Foundation (NSF-1812071).

References

[1] Y. Shen, A. Rezaei, and H. Zhou, "Sat-based bit-flipping attack on logic encryptions," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 629–632.

[2] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in ic manufacturing and test," *IEEE Transactions on Very Large Scale Integration Systems*, vol. PP, no. 99, pp. 1–13, 2018.

[3] R. Aitken, "Panel: Is design-for-security the new dft?" in *VLSI Test Symposium*, 2015.

[4] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.

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

[6] W. L. Yu and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *16th Latin-American Test Symposium (LATS)*, 2015.

[7] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in ic piracy with test-aware logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[8] K. Shamsi, D. Z. Pan, and Y. Jin, "On the impossibility of approximation-resilient circuit locking," in *Proceedings of the IEEE Symposium on Hardware Oriented Security and Trust (HOST)*. McLean, VA, USA: IEEE, 2019, pp. 161–170.

[9] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. Macau, China: IEEE, 2016, pp. 655–660.

[10] M. Fyrbiak, S. Wallat, J. Déchelotte, N. Albartus, S. Böcker, R. Tessier, and C. Paar, "On the difficulty of fsm-based hardware obfuscation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 293–330, 2018.

[11] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fpga fingerprinting techniques for protecting intellectual property," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1998, pp. 299–302.

[12] G. T. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-channel based watermarks for integrated circuits," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 30–35.

[13] T. Kean, D. McLaren, and C. Marsh, "Verifying the authenticity of chip designs with the designtag system," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 59–64.

[14] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101–1117, 2001.

[15] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 434–440, 2000.

[16] A. Cui, C. Chang, S. Tahar, and A. T. Abdel-Hamid, "A robust fsm watermarking scheme for ip protection of sequential circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 678–690, 2011.

[17] M. Meenakumari and G. Athisha, "A new approach to protect fpga based sequential ip cores," in *2014 International Conference on Electronics and Communication Systems (ICECS)*. IEEE, 2014, pp. 1–5.

[18] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[19] S. Dupuis and G. D. Natale, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *On-Line Testing Symposium*, 2014, pp. 49–54.

[20] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2015, pp. 137–143.

[21] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proceedings of the IEEE Symposium on Hardware Oriented Security and Trust (HOST)*. McLean, VA, USA: IEEE, 2017, pp. 46–51.

[22] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[23] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2018.

[24] R. S. Chakraborty and S. Bhunia, *HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection*, 2009.

[25] J. Dofe, Y. Zhang, and Q. Yu, "Dsd: A dynamic state-deflection method for gate-level netlist obfuscation," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 565–570.

[26] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "Ip protection and supply chain security through logic obfuscation: A systematic overview," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 6, pp. 65:1–65:36, 2019.

[27] Tech insights. [Online]. Available: https://www.techinsights.com

[28] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security." in *USENIX security symposium*, 2007, pp. 291–306.