Comparative Analysis of Hardware Obfuscation for IP Protection

Sarah Amir
sarah.amir@ufl.edu

Bicky Shakya
bshakya@ufl.edu

Domenic Forte
dforte@ece.ufl.edu

Mark Tehranipoor
tehranipoor@ece.ufl.edu

Swarup Bhunia
swarup@ece.ufl.edu

Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611-6200

ABSTRACT

In the era of globalized Integrated Circuit (IC) design and manufacturing flow, a rising issue to the silicon industry is various attacks on hardware intellectual property (IP). As a measure to ensure security along the supply chain against IP piracy, tampering and reverse engineering, hardware obfuscation is considered a reliable defense mechanism. Sequential and combinational obfuscations are the primary classes of obfuscation, and multiple methods have been proposed in each type in recent years. This paper presents an overview of obfuscation techniques and a qualitative comparison of the two major types.

CCS Concepts

•Security and privacy → Hardware security implementation; Hardware-based security protocols;

Keywords

Hardware IP Protection; sequential obfuscation; combinational obfuscation

1. INTRODUCTION

In the era of Internet-of-Things (IoT) devices, every device around ourselves are getting connected, and the more they do so, the more vulnerable they become to cyber attacks. Hardware is the base of any system. If the hardware is compromised, no amount of software protection can secure the system. Among many hardware security features, hardware obfuscation is an important tool to ensure the integrity of the ICs. Encryption is a strong tool to ensure the security, but it can not protect the IP in all stages of its life cycle. Hardware design must be transparent to the manufacturer and testing facility, but still needs to be obscure enough for

anyone to understand or modify. Hardware obfuscation can conceal the design, protect it from attacks and allow the manufacturing and testing to be performed securely.

In this paper we explore the requirement of hardware obfuscation in present day supply chain in Section 2; explain hardware obfuscation in Section 3 and present the taxonomy in Section 4 along with examples of state-of-art methodologies of the classifications. In Section 5, we present a comparative analysis of the two major class of obfuscation. Then we mention in Section 6 the key management issues and conclude in Section 7.

2. IP THREATS IN SUPPLY CHAIN

The semiconductor industry used to follow a vertical business model. Having all the facilities in the same organization helped to secure IPs. But now with the adoption of horizontal business model, the scenario is changing. A key point in this model is that modern System-on-chip (SoC) designs rely predominantly on third party IP (3PIP) because of the benefit in reduction of cost in manufacturing and time-to-market. The system integrator purchases IP from third party vendors. These IP vendors and the manufacturing or testing facilities are distributed throughout the globe. As an inevitable by-product, trust becomes an issue in each step of the supply chain. Ensuring IP trust among entities distributed globally becomes a complex problem given that the IP protection laws are different among countries. The main categories [25] of these vulnerabilities and threats, can be analyzed as:

1. IP Piracy: This threat occurs when an entity, other than the IP owner, sells the IP to a third party, either as original IP or as renamed. This deprive the designer or the owner of their share on revenue.
2. IP Overuse: This threat relate to the scenario when an entity uses the IP in more instances than the ones they paid for or got licensed from the original owner.
3. IP Modification: There is always the risk that a malicious SoC designer or untrusted foundry may modify the IP to insert *backdoor* or Hardware Trojan, *e.g.* a kill switch. This affects the trusted operation of the IC and harms the goodwill of original IP vendor and makes them liable for the damages.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada.

© 2017 ACM. ISBN 978-1-4503-4972-7/17/05...\$15.00.

DOI: <http://dx.doi.org/10.1145/3060403.3060495>

Figure 1: IP threats in supply chain.

4. Reverse Engineering(RE): This can be used with malicious intentions to get the higher level design of an IP and expose it to other threats explained above.

Figure 1 shows the threats in supply chain. The challenge to protect IP from all these threats is a diversified and complex problem. Depending on the type of an IP, the threats and challenges are different and so are the countermeasures. The IP vendors provides their IP in any of the three forms:

1. Soft IP: The IP is in behavioral or *Register Transfer Level* (RTL) format, written with Hardware Description Language (HDL). This is the most used format for 3PIP.
2. Firm IP: The IP is in structural or gate-level netlist format. The circuit is represented as list of gates and their interconnections.
3. Hard IP: The IP is in technology dependent layout format. This the physical design that can be used to manufacture the IC without further synthesis or optimization.

In a soft IP, the design is open and understandable to the receiver. It is vulnerable to IP piracy, overuse, and modification. A common practice to protect this soft IP is use of Encryption. An EDA tool provides a designer with an option to encrypt the whole or part of the IP with the tool's public key. In this way, the malicious integrator is restrained from directly observing the design in soft form. But a drawback of this is, in some cases, after the synthesis, the gate level netlist is no longer encrypted. This gate level design then becomes vulnerable to the threats nonetheless.

Unencrypted Firm IPs are vulnerable to IP piracy, overuse, modification, and it can be reverse engineered to retrieve the HDL design. But it is often necessary for the integrator to have the unencrypted netlist to do functional simulation and verification, or to insert scan chains. Because of that, solutions other than encryptions becomes necessary.

The hard IP is the design that usually is sent to offshore foundries to manufacture the IC and poses the threats of modification and reverse engineering to retrieve higher level design.

3. HARDWARE OBFUSCATION

To protect IP in all forms from retrieval and modification attacks, obfuscation is a vital tool. The software obfuscation has been there for sometimes now to protect software against piracy. The structure of this obfuscation is to have a functionally equivalent but structurally different design. As the designs are functionally equivalent, the input output association remains the same. But because the presentation gets altered, the interpretation becomes difficult. The hardware obfuscation to protect hardware IP is significantly

different than the software obfuscation. In case of hardware, along with the structural design, the functionality is needed to be hidden to protect it from malicious modification and other attacks. That renders the need for a mechanism to control functional equivalence by the IP owner. Keeping with these requirements, the hardware obfuscation has two major aspects

1. Functional locking, and
2. Structural modification of the design.

The design is modified in a way that it implements different logic functions and it is not possible to retrieve the correct logic equation by reverse engineering. A locking mechanism has to be incorporated which ensures the design becomes functionally equivalent upon correct *unlocking* process.

4. TAXONOMY OF OBFUSCATION

Based on the locking mechanism and design modification methods, hardware obfuscation techniques can be divided into two major classes:

1. Sequential Obfuscation: The design modification is performed by state machine modification. There is no additional input ports for key.
2. Combinational Obfuscation: The design modification is purely on the combinational logic and the locking key is taken from additional input ports or memory.

Figure 2: Hardware obfuscation taxonomy.

Based on the design level on which the obfuscation is done, as shown in Figure 3, the state-of-art hardware obfuscation methods can be classified into three categories:

1. Behavioral obfuscation: Some sequential obfuscation method works on this level of design. These are explained in details in Section 4.1.2.

Figure 3: Integration of hardware obfuscation in standard design flow.

2. Structural obfuscation: Most of the sequential obfuscation and almost all of the combinational obfuscation is performed on structural level designs.
3. Physical obfuscation: Multiple techniques have been proposed for physical obfuscation, which include *Mono-lithic Integration* [21], *2.5D/3D IC* split manufacturing [11], *Camouflaging* [20], and *CircuitEdit* [24].

4.1 Sequential Obfuscation

The sequential obfuscation is based on judicious modification of the finite state machine (FSM) in the design. Researchers offered numerous methods to modify the state transition graphs to control the locking key or to perform structural variations. An example of sequential obfuscation is presented in Figure 4. State-of-art methods can be put into two categories based on whether the obfuscation is done in structural design or in behavioral design.

Figure 4: Example of sequential obfuscation [4].

4.1.1 Sequential Obfuscation on structural design:

A modified FSM was proposed in [1] by Alkabani and Koushanfar and termed as boosted finite state machine (BFSM). The existing FSM is modified to have additional states, the number of which are significantly higher than the number of original states. This ensures the probability of power-up state to be in original states be negligible. This additional states, along with original states, are called as Boosted FSM. In addition to the BFSM, a random unique block (RUB), which utilized unique identification circuitry, is used to put the BFSM into additional states at power up. The security is based on the assumption that only the designer would know the design of the BFSM, and can provide the sequence to put the circuit into working mode. The sequence is the unique unlocking key for each IC. Also, the authors suggested to design ‘blackhole’ states to remotely disable the IC upon any evidence of tempering.

In [3], Chakraborty and Bhunia proposed *ObfusFlow*, in which a simple FSM is integrated in the sequential circuit for controlling the locking-unlocking mechanism. The output of

the FSM is XORed with few internal nodes. This technique can be considered as *sequentially controlled combinational obfuscation*.

The authors improved the method in their follow-up paper [4], initiating the idea of obfuscation in gate level files to protect it from supply chain threats. They defined a methodology, *HARPOON*, for ensuring security in SoC designing. The method offers both obfuscation and authentication in designing and manufacturing steps by modifying both the state transition function and the internal logic structures of the design. In state transition modification part, they implemented an additional FSM for initialization. Upon successful input sequence this FSM drives the circuit to working mode. For structural obfuscation, modification cells are proposed, which takes input from the state elements and performs Boolean operation on selected circuit nodes. This cell XORs the original node in some specific scenario which are controlled by both the sequential and combinational logic. The nodes to insert these cells or to take input from are selected based on fan-out and fan-in size of the nodes. Along with the design process, the paper gives a detailed analysis of securing design flow with the help of obfuscation. The authors, on a separate paper [6], extended the applicability of the obfuscation to soft IPs by decompiling the obfuscated firm IP.

Along with IP protection, the role of sequential obfuscation in protecting against hardware Trojan attack was studied in [5]. In this paper, authors proposed the process to modify *State Transition Graph* (STG) to integrate obfuscation in the structural design.

Li and Zhou offered an obfuscation based on ‘stuttering’ [13]. The obfuscated circuit is not functionally different from the original one, but slower. The slow-down is expected to be polynomial-time to degrade the design. The state machine controls two modes of the IC, *normal* and *slow mode*. Until the IC reaches the end of supply chain, state elements are loaded with wrong sequence for the IC to be in *slow mode*. It is hypothesized that the throughput difference is large enough for the obfuscation to have any effect and small enough for any attacker to realize that.

4.1.2 Sequential Obfuscation on behavioral design:

Though soft IP or RTL level design obfuscation was mentioned before [6], but it was still performed on gate-level netlist, and then a de-compilation stage regenerated the RTL design. An actual RTL level obfuscation was introduced in [7]. In this paper, the authors extract *Control Flow Graph* (CFG) from the high level description and modifies the graph to have extra decision nodes. The modified or obfuscated CFG is then converted to obfuscated RTL design. The additional control decisions are based on boolean operation on state elements of the obfuscation FSM.

Another technique that performs the obfuscation on RTL design is proposed in [9] where the authors considered multiple inputs together as *code-words* for driving the obfuscated FSM. The key aspect of the design is that, even with wrong *code-word*, the state of FSM goes from initial *entry mode* to *functional mode*, but in that case, the behavior of the *functional mode* is different. The authors additionally claimed that the *entry mode* FSM is inseparable from the *functional mode* FSM.

4.2 Combinational Obfuscation

The combinational obfuscation is the hardware obfuscation where the design modification and locking mechanism is purely combinational. Usually this obfuscation is performed on structural level design and the locking key is taken from outside as primary input or read from tamper-proof non volatile memories on chip. If the key is taken as inputs, the number of modified nodes or inserted locks are proportional to the increment in primary inputs. In most cases, the key exchange is performed through external secure encryption protocol. Figure 5 shows an example of combinational obfuscation technique. The state-of-art combinational obfuscation methods can be analyzed into different group based on the similarity in concepts.

Figure 5: Example of combinational obfuscation.

4.2.1 Datapath obfuscation

Roy et. al. suggested the idea to scramble the bus connecting the components in the IC [23]. The scrambling on the output and de-scrambling on the input of each component is done with key-based Benes network. Each network has butterfly and reverse-butterfly models coupled, which offers scrambling flexibility, but at a higher cost. Authors also suggested Diffie-Hellman key exchange protocol for unlocking the chip. No analysis on overhead has been done, but structural modification with Benes network in this volume hints to large overhead.

A deeper insight in industrial designs gave rise to attacks based on scan chain [29, 16]. In these papers, respective authors used the vulnerability of *scan-enable* mode of IC to leak secrets. For protection against attacks on scan chain, methods have been developed to break scan chain into parts and the reconnection is done through a key-based scrambler block. [8, 17, 18]. In manufacturing and assembly testing, upon application of wrong key, the scan chains are scrambled but the designer knows about how the scrambling is done. As a result, though the scan chain based testing result would be different than un-obfuscated IC, the designer can still evaluate whether the IC is working as expected or not. This concept can be viewed as *Combinational obfuscation on sequential circuits*.

4.2.2 Logic obfuscation

The first proposition of combinational obfuscation was made by Roy et. al [22]. They introduced purely combinational locking mechanism of placing key-controlled XOR-XNOR gates randomly in the design. Along with the algorithm of obfuscation, they provided the model to implement chip activation and secure communication based on public key cryptography and true random number generator.

Rajendran et. al. published an improvement of the randomly insertion of locking gates in circuit by offering a heuristically selected insertion based on fault analysis [19]. They introduced a *fault impact* metric which indicates how much impact a locking gate would have in a particular position. In key gate type, the authors offered an alternate to XOR based design by proposing MUX gates to scramble the internal wire connections upon wrong key input. In order to decrease masking of locking effect, it is advised to minimize the correlation between those by avoiding placing new locks in the logic cone of previously inserted one. But later, the authors reverted this idea in follow-up paper and proposed a new heuristic to place locks with higher correlation, renamed later as *Secure Logic Locking(SLL)* [32]. The design is based on the correlation type between locking gates. Though the SLL shows resiliency against *key-sensitizing attack* [32], the algorithm presented has polynomial complexity and not scalable for industrial size designs. The interconnection of these locking networks was improved by connecting them with MUX gates [12].

Dupuis et. al. explained an obfuscation technique that can also be used against hardware trojan attacks [10]. They designed a way to improve the controllability of signals with lowest controllabilities, termed as *Rare nodes*, which can be used as triggering signal of trojans. The heuristic to insert obfuscation block selected the *Rare node* signals that are not in the critical delay paths. The limitation of this method is, the number of *Rare nodes* among the total nodes in a practical IC is very high, and the small amount of modification with practical overhead makes the improvement negligible.

4.2.3 Reconfigurable logic based obfuscation

A proposal of using reconfigurable logic as a combinational obfuscation tool was presented by Baumgarten et. al. [2]. They explained about a hypothetical *slice* in the IC design through which all the data paths from input to output crosses and that is almost equidistant from the inputs and outputs in terms of observability and controllability. The authors suggested to replace the logic gates on the hypothetical *slice* with reconfigurable logic blocks, like look-up-table(LUT)s. A limitation of this model is, the size of the *slice* can be exponentially large for industrial design, hence insertion of large number of lock would have monstrous overhead. The protection lies in the concept that- as long as the LUTs are not loaded with correct logic function, the design is essentially obfuscated and not possible to reverse engineer. For the same security concept, reconfigurable logic based obfuscation was suggested to obfuscate instruction decoder unit (IDU) to secure it from code injection Trojan [14].

Wendt and Potkonjak proposed methods of cascading Physically Unclonable Function (PUF) and reconfigurable logic to replace a block of combinational logic to render the entire circuit design to be hidden [27]. They also offered a method of datapath obfuscation with the reconfigurable logic.

Table 1: Comparison between sequential and combinational obfuscation techniques.

Comparison basis	Sequential obfuscation	Combinational obfuscation
Implementation method	The modification and addition are sequential logic based. In most scenarios, state machines are modified and utilized to lock the design .	The modification and addition are based on purely combinational logic. The logic can either be simple logic gates, look-up-tables (LUTs) or block of combinational logic.
Applicability	Applicable to designs that has sequential elements, i.e. registers. Most practical circuits have sequential components and state machines.	The authors of combinational obfuscation methods experimented their methods only on purely combinational benchmarks. In case of industrial IP, purely combinational designs are rare.
Scalability	Unless the graphs in [5, 7] of industrial design becomes too large to work on, scalability is not an issue. Also, cascading separate FSM [1, 4] would not be an issue for scalability.	Some of the methods are easily incorporable to large scale designing [22], but there are few methods that are impractical for real world circuitry [32].
Overhead	Some methods have low overheads [1, 3, 5], but others have moderate [9, 13, 7].	Techniques of [14, 27] have low overheads, but [19, 2, 23] have fairly high overheads.
Design Flow	Most of the methods work on behavioral or structural design.	Almost all the models are based on structural design modification.
Security against RE attack	The security is built on the concept that it is not possible to reverse engineer the state machines efficiently.	Reverse engineering can easily detect the gates connected to key inputs. Whether it will then be removable depends on specific method.
Security against SAT attack [26]	No obfuscated sequential circuit was broken with SAT attack. The tools authors used to attack can only work with a file type that cannot handle memory component.	SAT attack is shown to find the key inputs unless a dedicated resiliency block [28, 30] is accompanied.
Security against KSA [32]	The attack takes the advantage of the direct dependency of output bits to key values. Whether memory delays or sequential key input successfully thwarts the attack is yet to be proven.	Keys can be deduced if those are not securely correlated with other key gates. If the original circuit itself has high correlation in gates, like a multiplier, the attack efficiency is lessened.

5. COMPARISON OF SEQUENTIAL AND COMBINATIONAL OBFUSCATION

While the decision whether the obfuscation of any particular IP should be sequential or combinational or both depends on the characteristic of the IP and the choice of the designer, a comparative analysis can be drawn between these two types on various properties. Table 1 presents such a qualitative comparison.

6. LOCKING AND KEY MANAGEMENT

The locking mechanism of hardware obfuscation can both be key-less and keyed. In keyless obfuscation, there is no necessity of a secret key. The physical obfuscations- *Mono-lithic Integration* [21], *2.5D/3D IC* split manufacturing [11], *Camouflaging* [20], and *CircuitEdit* [24] are examples of key-less obfuscation.

In case of keyed obfuscation, one or more secret key is required to ‘lock’ and ‘unlock’ the functionality of the circuit [22, 19, 32, 1, 4]. In a combinational obfuscated circuit, all the key bits are applied at once, while in sequential obfuscated IC, the key bits are applied sequentially. These keys need to be stored within the IC in a tamper-proof memory, or be generated by a PUF. Also, to provide safe communication of the key between the designer and the IC, often there arise the requirement to have strong encryption hardware like AES or RSA inside the chip.

There had been a number of attacks proposed to retrieve the key from the IC. Among those, two prominent attack

are key-sensitizing attack (KSA) and Boolean satisfiability (SAT) attack. In KSA [32], the keys are propagated to primary output by specific input pattern utilizing automatic test pattern generation tool. In SAT attack, the Boolean satisfiability algorithm is utilized to rule out wrong keys for distinguishing input patterns (DIPs) [26] using conventional SAT solvers. This attack can efficiently break obfuscation and extract key in negligible amount of iterations.

It is worth mentioning here that, both these attacks have been proposed on purely combinational circuits. How efficient these will be against sequential obfuscation is still an open question.

Methods to resist these attacks has been published. In [30, 28], respective authors proposed comparing key inputs and only selectively obfuscating the circuit for certain key patterns, to increase the number of DIPs in boolean satisfiability attack. But recently, it was pointed out that protecting from SAT attack, the SAT resilient circuit blocks makes the obfuscation vulnerable to emerging attacks like *signal probability skew (SPS)* attack [31]. That renders the combinational logic locking with major challenges.

7. CONCLUSIONS

Hardware obfuscation is an emerging field in hardware security. New attacks are surfacing and countermeasures that are feasible, scalable, and practical are desired. Emerging concepts like using polymorphic gates in logic level obfuscation [15] and using focused ion beam (FIB) to modify chip post-fabrication [24] are promising. Further, various

sequential obfuscation techniques appear to be very powerful for protecting all forms of IP. As we are entering an all-connected device era, we have to be prepared with a strong and secure hardware platform to accommodate high level of security. And for that, hardware obfuscation, along with other hardware security features, are becoming increasingly significant.

8. ACKNOWLEDGMENTS

This work is supported in part by Cisco Systems and National Science Foundation.

9. REFERENCES

- [1] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, 2007*, 2007.
- [2] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers*, 27(1):66–75, 2010.
- [3] R. S. Chakraborty and S. Bhunia. Hardware protection and authentication through netlist level obfuscation. In *ICCAD 2008, San Jose, CA, USA, 2008*, pages 674–677, 2008.
- [4] R. S. Chakraborty and S. Bhunia. HARPOON: an obfuscation-based soc design methodology for hardware protection. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.
- [5] R. S. Chakraborty and S. Bhunia. Security against hardware trojan through a novel application of design obfuscation. In *ICCAD 2009, San Jose, CA, USA, 2009*, pages 113–116, 2009.
- [6] R. S. Chakraborty and S. Bhunia. Security through obscurity: An approach for protecting register transfer level hardware IP. In *IEEE Intl. Workshop on HOST 2009, San Francisco, CA, USA, 2009. Proceedings*, pages 96–99, 2009.
- [7] R. S. Chakraborty and S. Bhunia. RTL hardware IP protection using key-based control and data flow obfuscation. In *VLSI Design 2010: 23rd Intl. Conf. on VLSI Design, 9th ICES, Bangalore, India, 2010*, pages 405–410, 2010.
- [8] G. K. Contreras, M. T. Rahman, and M. Tehranipoor. Secure split-test for preventing IC piracy by untrusted foundry and assembly. In *IEEE Intl. Symposium on DFTS 2013, New York City, NY, USA, 2013*, pages 196–203, 2013.
- [9] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall. Interlocking obfuscation for anti-tamper hardware. In *CSIRW '13, Oak Ridge, TN, USA, 2013*, page 8, 2013.
- [10] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *IEEE 20th IOLTS 2014, Platja d'Aro, Girona, Spain, 2014*, pages 49–54, 2014.
- [11] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara. Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, 2013*, pages 495–510, 2013.
- [12] Y. Lee and N. A. Toubia. Improving logic obfuscation via logic cone analysis. In *16th LATS 2015, Puerto Vallarta, Mexico, 2015*, pages 1–6, 2015.
- [13] L. Li and H. Zhou. Structural transformation for best-possible obfuscation of sequential circuits. In *IEEE Intl. Symposium on HOST 2013, Austin, TX, USA, 2013*, pages 55–60, 2013.
- [14] B. Liu and B. Wang. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *DATE 2014, Dresden, Germany, 2014*, pages 1–6, 2014.
- [15] J. T. McDonald, Y. C. Kim, T. R. Andel, M. A. Forbes, and J. McVicar. Functional polymorphism for intellectual property protection. In *IEEE Intl. Symposium on HOST 2016, McLean, VA, USA, 2016*, pages 61–66, 2016.
- [16] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor. AVFSM: a framework for identifying and mitigating vulnerabilities in fsm. In *Proceedings of the 53rd Annual DAC 2016, Austin, TX, USA, 2016*, pages 89:1–89:6, 2016.
- [17] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras, and M. Tehranipoor. CSST: an efficient secure split-test for preventing IC piracy. In *IEEE 23rd NATW 2014, Johnson City, NY, USA, 2014*, pages 43–47, 2014.
- [18] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras, and M. M. Tehranipoor. CSST: preventing distribution of unlicensed and rejected ics by untrusted foundry and assembly. In *IEEE Intl. Symposium on DFT 2014, Amsterdam, The Netherlands, 2014*, pages 46–51, 2014.
- [19] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Logic encryption: A fault analysis perspective. In *DATE 2012, Dresden, Germany, 2012*, pages 953–958, 2012.
- [20] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. Security analysis of integrated circuit camouflaging. In *ACM SIGSAC Conference on CCS'13, Berlin, Germany, 2013*, pages 709–720, 2013.
- [21] J. Rajendran, O. Sinanoglu, and R. Karri. Is split manufacturing secure? In *DATE 13, Grenoble, France, 2013*, pages 1259–1264, 2013.
- [22] J. A. Roy, F. Koushanfar, and I. L. Markov. EPIC: ending piracy of integrated circuits. In *DATE 2008, Munich, Germany, 2008*, pages 1069–1074, 2008.
- [23] J. A. Roy, F. Koushanfar, and I. L. Markov. Protecting bus-based hardware IP by secret sharing. In *Proceedings of the 45th DAC 2008, Anaheim, CA, USA, 2008*, pages 846–851, 2008.
- [24] B. Shakya, N. Asadizanjani, D. Forte, and M. M. Tehranipoor. Chip editor: leveraging circuit edit for logic obfuscation and trusted fabrication. In *Proceedings of the 35th ICCAD 2016, Austin, TX, USA, 2016*, page 30, 2016.
- [25] B. Shakya, M. Tehranipoor, S. Bhunia, and D. Forte. *Introduction to Hardware Obfuscation: Motivation, Methods and Evaluation. Hardware Protection through Obfuscation*, chapter 1, pages 3–32. Springer, 2017.
- [26] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE Intl. Symposium on HOST 2015, Washington, DC, USA, 2015*, pages 137–143, 2015.
- [27] J. B. Wendt and M. Potkonjak. Hardware obfuscation using puf-based logic. In *The IEEE/ACM ICCAD 2014, San Jose, CA, USA, 2014*, pages 270–277, 2014.
- [28] Y. Xie and A. Srivastava. Mitigating SAT attack on logic locking. In *Proceedings of 18th Intl. Conf. on CHES 2016, Santa Barbara, CA, USA, 2016*, pages 127–146, 2016.
- [29] B. Yang, K. Wu, and R. Karri. Scan based side channel attack on dedicated hardware implementations of data encryption standard. In *Proceedings (ITC 2004), 2004, Charlotte, NC, USA*, pages 339–344, 2004.
- [30] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu. Sarlock: SAT attack resistant logic locking. In *IEEE Intl. Symposium on HOST 2016, McLean, VA, USA, 2016*, pages 236–241, 2016.
- [31] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. Security analysis of anti-sat. In *22nd ASP-DAC 2017, Chiba, Japan, 2017*, pages 342–347, 2017.
- [32] M. Yasin, J. J. V. Rajendran, O. Sinanoglu, and R. Karri. On improving the security of logic locking. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 35(9):1411–1424, 2016.