Blockchain-enabled Cryptographically-secure Hardware Obfuscation

Fatemeh Ganji, Shahin Tajik, Domenic Forte
Florida Institute for Cybersecurity Research
University of Florida
601 Gale Lemerand Dr.
Gainesville, USA 116550

Jean-Pierre Seifert
Security in Telecommunications
Technische Universität Berlin
Ernst-Reuter-Platz 7
Berlin, Germany 10587

ABSTRACT

Electronic chips in consumer, industrial, and military applications are targeted by untrusted entities in the design and fabrication process, aiming at reverse-engineering and Intellectual Property (IP) piracy. Hardware obfuscation techniques, namely logic locking and IC camouflaging, have been developed to mislead an adversary hoping to reverse engineer the netlist and extract the functionality. However, virtually all existing hardware obfuscation schemes developed over the last decade have been shown to be vulnerable to oracle-guided attacks, e.g., SAT and machine learning attacks. Moreover, most of these schemes rely on an ideal tamper- and read-proof memory to store the key used to unlock the circuit after manufacturing. In this work, we propose two novel cryptographically-secure hardware obfuscation schemes using garbled circuits, which are compatible with current circuit synthesis and fabrication tools. Our first construction does not require any secure hardware with tamper- and read-proof memory. In this case, the security of the obfuscation is guaranteed by Proof-of-Stack blockchain protocols and witness encryption schemes. However, for our second construction, we assume the existence of secure memory in the hardware to achieve higher performance and less overhead. Both constructions are inspired by program obfuscation and one-time program techniques enabling us to selectively encrypt and garble some IP cores during integration as well as manufacturing to prevent IP piracy. Furthermore, with the help of our constructions, we can realize one-time and pay-per-use hardware, where a user can use the electronic circuit for a limited amount of time.

KEYWORDS

IP Piracy; Logic Locking; Hardware Obfuscation; Garbled Circuits; Witness Encryption; Blockchains

1 INTRODUCTION

Over the years, electronic components and their supply chains have been considered secure and trustworthy. However, globalization of the modern integrated circuit (IC) supply chain refutes this assumption. As the fabrication of the semiconductors moves to smaller nodes, more advanced and sophisticated fabrication facilities are needed. To keep the production of ICs with the latest technology nodes profitable, most market leading IC vendors have become fabless [15], where the their products are fabricated overseas by an independent foundry. Different phases of chip manufacturing, such as design, integration, and fabrication can no longer be carried out under the same roof, see Fig. 1. Therefore, original IP owners no longer have control over the entire supply chain. Consequently, ICs become vulnerable to IP piracy, tampering, and counterfeiting. These problems continue even when the devices are delivered to the malicious end-users in the market.

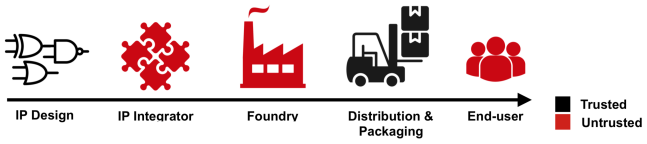

Figure 1: The supply chain in semiconductor industry. It is assumed that the designer, packaging and distribution are trusted, while the IP integrator, foundry and end-users are untrusted.

To address these issues, several IP obfuscation schemes have been proposed to prevent IP piracy of electronic chips attempted by IP integrators and untrusted foundries. In an attempt to regain control over the design, it is suggested to manufacture only the front-end-of-line (FEOL) layers at an untrusted high-end foundry, whereas a trusted low-end foundry should take over manufacturing the back-end-of-line (BEOL) layers [12]. The split manufacturing approach, although being a promising solution, has shortcomings that have been identified in the literature (e.g., [28]). As a prime example, it has been demonstrated that a malicious FEOL foundry can launch a heuristic-based attack to circumvent security measures offered by some split manufacturing techniques [20].

Further efforts to protect IPs cover a wide range of techniques developed over the past decade: compiler-level, gate-level, and layout-level hardware obfuscations. The former type of obfuscation techniques refers mainly to Finite State Machine (FSM) locking, so-called sequential logic locking, where the FSM is augmented by adding a new set of states [7]. This type of approach can also be applied at the gate-level as proposed in, e.g., [2, 3] and layout-level, see [4, 16]. In addition to this, at the gate-level, the most prominent example is (combinational) logic locking methods that include extra key gates in a design, which are controlled by a key given to it as input bits [21]. Logic masking [5] and logic permutation [8] are other techniques built upon the idea of setting a fixed output for wrong keys and permutation of interconnections by a key, respectively. Although being effective in some scenarios, no proof has supported the approaches mentioned above.

At the layout-level, camouflaging is performed to hide the functionality of a standard cell by employing a combination of real

and dummy contacts [19]. It has been assumed that an attacker requires exponential time, in the number of camouflaged gates, to de-camouflage a circuit; however, this assumption has become invalid as the de-camouflaging problem is reduced to Boolean satisfiability (SAT) problem and solved by applying off-the-shelf SAT-solvers. The application of such solvers in the hardware obfuscation area is not limited to this as they are widely adopted to compromise the security of logic locked circuits, see, e.g., [27]. In addition to this, modified, approximation-based versions of SAT attack, so-called APP-SAT, has been applied to de-obfuscate circuits effectively [22]. A recent work has gone even beyond this by showing that the existing locking techniques cannot be resilient against approximation attacks, when the adversary is given access to an oracle providing her with the outputs of an unlocked design [23]. This can further emphasize the importance of applying cryptographically-secure techniques to guarantee the security of the obfuscated circuits.

Unfortunately, virtually all of these schemes have been developed in an ad-hoc and heuristic fashion, as discussed above. Moreover, the existence of a tamper- and read-proof memory is the primary assumption made by several obfuscation techniques, e.g., logic locking. However, the most secure memory candidates, which are all based on non-volatile memory technologies, are susceptible to physical attacks, making direct readout possible [18]. As a conclusion, there is an ever-increasing need for an obfuscation technique, which is provably secure against oracle-guided attacks and relies on secure memories as little as possible.

Our Contribution: This paper suggests a novel approach aimed at not only addressing the issues with existing obfuscation methods but also enabling "pay-per-use circuits." This notion offers a higher security level since not only the first access to the circuit can be restricted, but also the total number of accesses can be pre-defined. Interestingly, our approach relies on the existence of neither a tamper-proof memory nor a self-destructing one, as required by a scheme proposed in [10]. Instead, we rely on inherent characteristics of the Blockchain (BC) technology that is, it can be regarded as a "platform" enabling us to achieve the security in the sense of cryptography. In this regard, as proved in [11], the security of our scheme is related to the security of the BC. This also explains the core difference between our work and [30], where the BC is deployed to monitor the integrity of the electronics supply chain.

Finally, we stress that our idea can be considered as a step towards the further development of a provable method, which has rarely been fully researched in the hardware obfuscation area. Compared with the most relevant study of this matter, i.e., [6], our paper does not focus on a specific type of obfuscation approaches. However, both of our methods and one proposed by Crescenzo et al. [6] apply formal models as an enabler. While we employ the BC cf. [11], indistinguishability obfuscation is considered in [6], as such models suggested for program obfuscation, the latter cannot adequately reflect the challenges confronting hardware obfuscation, as also mentioned in [6]. To address this, we provide an exhaustive discussion on how and to what extent our scheme can be implemented in real-world scenarios. In fact, this is a crucial contribution made by our paper: although the security of the BC-enabled pay-per-use circuits has been proven in the literature [11], we demonstrate how to adopt those results to achieve secure hardware obfuscation.

2 BUILDING BLOCKS OF OUR SCHEME AND ADVERSARY MODEL

The core idea of our solution is to overcome the limitations of previous approaches by applying the notion of BC in conjunction with witness encryption and garbled circuits, as described below.

2.1 BCs and Proof-of-Stake Protocols

In our scheme, BCs can be seen as an alternative to the trusted-setup assumptions, i.e., the existence of tamper-proof hardware. This is due to the fact that BCs have been proven to offer the security-related features demanded by construction using tamper-proof hardware, e.g., one-time programs and pay-per-use programs [11]. Moreover, BCs enable us to deal with malicious parties, namely malicious foundries and users in our scenario.

Regarding the mechanism used to reach consensus, commonly referred to as "mining", BC protocols can be categorized as Proof-of-Work (POW) and Proof-of-Stake (POS) [14]. As for the former significant amount of computational power is required, in the latter case, a miner has to provide a sufficient balance. More specifically, if a party attempts to generate a block, the POS should be used as a certificate to verify the correctness. In our scenario, POS BCs provides assurance that a party (legitimate or malicious) can evaluate a circuit for only a limited time, depending on its balance [11].

2.2 Witness Encryption (WE)

The concept of (extractable) WE is similar to public-key encryption/ decryption, although the secrecy is handled in a different manner. For public key-based scheme, a secret key associated with a public one is required, whereas a message encrypted by an (extractable) WE can be decrypted if the solution to some NP-hard search problem (so-called, a witness) is known [9]. For instance, suppose that the decryption is possible if a solution to an NP-hard puzzle is known. In our scheme, such witness is the existence of users' blocks in the BC. This is possible thanks to the NP relation on the BC protocol defined based on the uniqueness of the local states of parties and their transaction over the BC [11]. Especially, for the pay-per-use application, there should be an evidence showing that a pre-specified amount of cryptocurrency is transferred from the user to the IP owner (i.e., service provider).

2.3 Garbled Circuits

The notion of garbled circuits can be thought of as randomized encoding of Boolean circuits [1]. Among several interesting applications of garbled circuits, we are interested in how they are employed to construct one-time programs, i.e., one-time circuits in our case. In this regard, it is desired to encode a circuit into one that can be executed only once, on an arbitrarily chosen input. Informally, after garbling a circuit C with n inputs, we obtain a garbled circuit, together with $2n$ wire keys. Afterward, when evaluating a garbled circuit on given key and input (see Section 4 for more details), the output of the garbled circuit is the same as the output of our circuit C (our circuit before garbling), when it is fed by the same input. The security of such a scheme is associated with the fact that during the evaluation of the garbled circuit, the information neither on circuit C nor on the input is revealed.

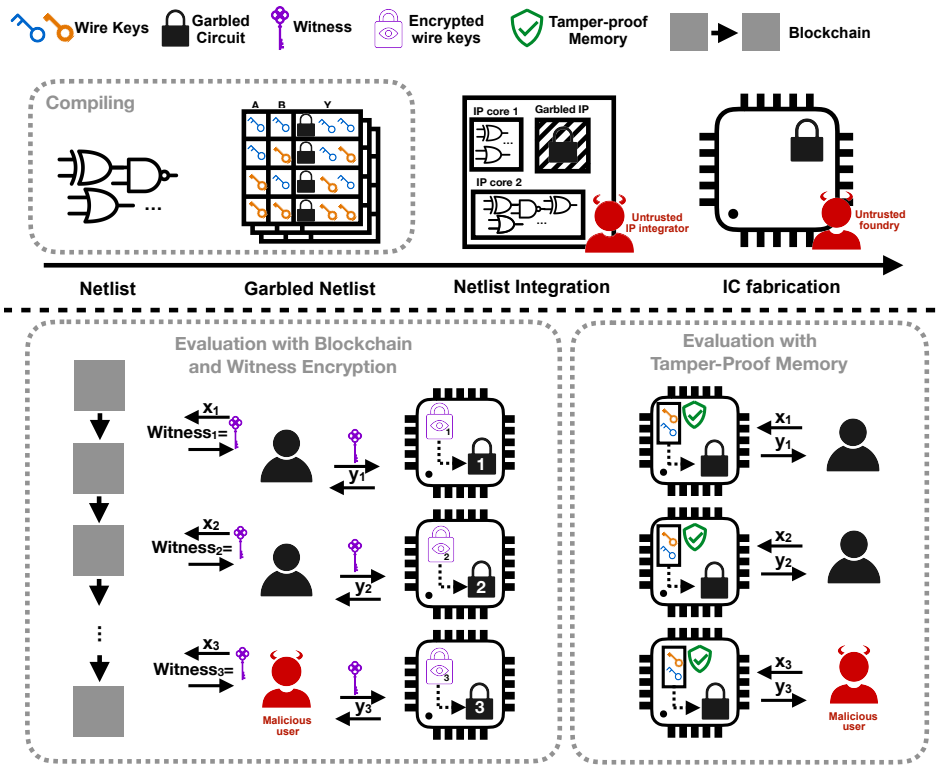

Figure 2: Generating garbled IPs and their deployment in the field. The wire keys can decrypt the garbled IP for evaluation. Construction 1 (Evaluation with Blockchain and and Witness Encryption) stores encrypted wire keys on the chip and uses Blockchain to get a witness along with the current state of the user to decrypt them. Construction 2 (Evaluation with Tamper-proof Memory) stores the wire keys in plaintext in a tamper-proof memory.

After establishing the foundation of our framework, we can now define the adversary model considered in our work.

2.4 Adversary Model

Similar to the most relevant studies on circuit obfuscation, we consider adversaries attempting to run polynomial-time (in a security parameter λ) algorithm to deobfuscate the netlist, cf. [6, 23]. In our attack model, the adversary is given access to the black-box hardware component. More precisely, there exists a probabilistic polynomial time (PPT) adversarial algorithm, when being given the above access, whose output is indistinguishable from the output of a simulator with *restricted* oracle access to the circuit. As can be understood, the crucial difference between our model and the existing adversary model in the circuit obfuscation-related studies is that we *ensure* limited oracle access given to the adversary.

Regarding the interaction between the adversary[1] and the BC, we assume that the adversary has complete access to the BC and can possibly have a malicious influence on the protocol execution by mining blocks or deviating from the protocol, cf. [11]. Finally, with respect to the notion of WE, we say that the adversary can extract some non-trivial information about the encrypted message only if she can come up with a witness for the instance used during encryption. In Section 3, we explain how such a witness can be crafted for honest parties and why the adversary cannot know any witness.

[1]Needless to say that by the term adversary, we refer to the above PPT algorithm controlling all the corrupt parties.

3 CRYPTOGRAPHICALLY-SECURE HARDWARE OBFUSCATION

Our proposal covers two constructions: Construction 1 ensures a high degree of security that is, no tamper-proof hardware is required. This can be achieved at the price of implementing POS BCs equipped with WE. Nevertheless, if a tamper-proof hardware is used, another construction (Construction 2) can be built exhibiting provably-secure obfuscation. In both constructions, the Boolean circuit of one or more IPs are garbled, and the wire keys associated with them are generated locally, see Fig. 2. The garbled truth tables or lookup tables are then sent to the IP integrator, and eventually to the foundry. Consequently, the garbled lookup tables are integrated and fabricated along with other IP cores on the chip. Note that for both of the constructions explained here, the proofs of the security has been given in [11], although we adapt those to the context of hardware obfuscation.

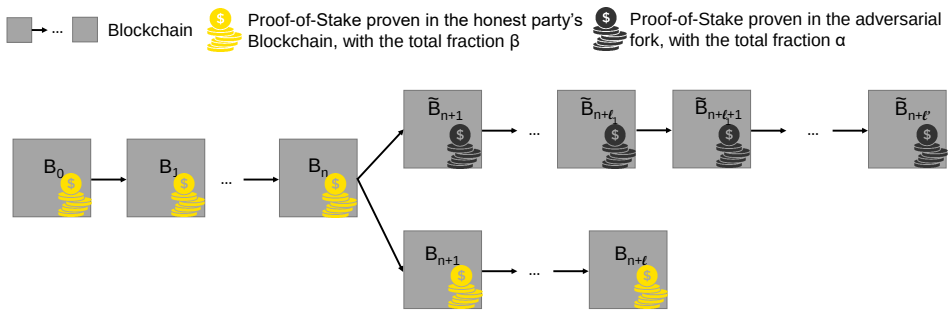

Figure 3: A Blockchain with forking distinguishability property. In our setting, the number of honest parties is n. The length of the maliciously generated fork is ℓ', and its minimum length is $\ell_1 + \ell_2$: $\ell' \geq \ell_1 + \ell_2$ (the first part of the fork is denoted by ℓ_1). The length of the honest parties' local blockchain is ℓ. The fraction of the amount of stake that is proven in the honest parties' blockchain is at least β, whereas in the adversarial fork, it is at most α. The above parameters (α, β, ℓ_1, and ℓ_2) reflect the hardness of our Blockchain-based scheme [11].

3.1 Construction 1

The process of committing a circuit over the BC **B** begins with garbling the circuit resulting in a garbled circuit and wire keys, as described above. Clearly, these wire keys must be stored encrypted on the chip so that the circuit cannot be evaluated freely. Although one can encrypt the wire keys by employing a public key system, we stick to WEs since they allow us to decrypt wire keys only conditionally as required by the one-time circuit and pay-per-use approaches [11]. Furthermore, WEs meet the one-time secrecy condition, i.e., during the evaluation, only the wire keys corresponding to the input given to the circuit is revealed. Therefore, the IP owner (i.e., the service provider) encodes the wire keys independently by using a WE system, see Fig. 2. Moreover, the IP owner defines a unique identity (id) for each IC. Note that the security of our scheme *does not* depend on the security of this id and it can be public.

When the IC is registered, the design - including the garbled circuit and its corresponding wire keys- along with the id and the initial balance associated with that id is committed over a blockchain **B**. It is evident that although the design is the same for a family of chips, the garbled circuits are instance-specific, ID-related; otherwise, the attacker can pay to use a circuit, but use her credential to use other chips. To evaluate the compiled circuit, as a *witness* the user has to generate a blockchain **B'** that (1) contains a block with the input, on which the circuit should be evaluated, and (2) that block should be followed by at least a pre-defined, minimum number of blocks, e.g., n. These n blocks contain a minimum amount of combined POS α to stop adversaries attempting to generate those n blocks by themselves (i.e., malicious extension). It is worth noting here that even if a malicious blockchain $\tilde{\mathbf{B}}$ is extended, the adversary still has to deal with the garbled circuit. Moreover, to ensure that the user commits only one input over the BC, our scheme naturally involves a mechanism to check the witness **B'**. Finally, to draw a conclusion of this section, we stress that the one-time secrecy and security against a malicious extension of the BC is reduced to the security of the BC (e.g., chain consistency, etc.) equipped with the WE and security of garbling scheme, respectively (see [11] for the proofs). This point, as well as how existing approaches may achieve the security-related requirements, have been summarized in Table 1.

How to set parameters related to POS BC to achieve the secure scheme: First, we again put emphasis on the role of the BC. In our framework, the purpose of the BC is to offer a platform, upon which we construct a provably-secure scheme. To base such a platform on the POS BC, we must define a setting, in which the security of our construction can be formalized and proved. To this end, we begin with the requirement that is, any *fork* crafted by the adversary on her own (i.e., in an off-line manner) must be clearly distinguished from the real BC, cf. [11]. More precisely, with high probability, we can distinguish an individual, invalid chain of blocks generated by an adversary from the honest parties' BC, see Fig. 3.

For this purpose, we define a threshold for the amount of POS belonging to an adversary as well as a minimum value for the POS owned by the honest parties. More specifically, the fraction of the amount of stake proven in the honest parties' blockchain is at least β, whereas, in the adversarial fork, it is at most α. It has been proven that the above parameters (α, β, and the length of adversary's fork, all polynomial in λ) reflect the hardness of our Blochchain-based scheme [11]. Note that this requirement is in line with the consistency and quality properties of appropriate stake sharing between the honest parties, as satisfied by POS based blockchain protocols, e.g., [14].

3.2 Construction 2

For the second construction, we assume the existence of a tamper- and read-proof memory on the chip (see Fig. 2 and Table 1). In this case, after the generation of the garbled IP, the wire keys are not needed to be stored encrypted on the chip and can be merely stored in the secure memory, see Fig 2. During the evaluation phase, based on the user's input, the corresponding wire keys are read from the memory and fed to the garbled IP. As a result, the garbled IP is decrypted and evaluated. While the assumption of having a tamper- and read-proof memory on the chip makes this construction similar to the conventional logic locking schemes, the security of the obfuscated IP is the primary difference of this construction. In other words, in contrast to the heuristic locking techniques, this

Property	Camouflaging	Logic Locking	Construction 1	Construction 2
Cryptographically-secure	✗	✗	✓	✓
One-time Circuit	✗	(✓)*	✓	(✓)*
Pay-per-use Circuit	✗	(✓)*	✓	(✓)*
Protection against SAT Attacks	✗	✗	✓	✓
Protection against Physical Attacks	✗	✗	✓	✗
Requiring No Tamper-proof Memory	✓	✗	✓	✗

* When equipped with self-destructing memory

Table 1: Security requirements and how they are met by previous approaches and ours

construction still deploys garbled circuits to obfuscate the netlist, which is cryptographically secure.

4 PRACTICAL CONSIDERATION

Last but not least, to support our theoretical, abstract constructions, here we highlight how the practical challenges facing our scheme can be addressed. First and foremost, the question would arise whether a WE scheme can be integrated into a BC system. This has been already discussed and proposed in the literature [17]. Moreover, we should come up with a WE scheme that offers extractability and efficiency. For this purpose, a promising candidate can be extractable hash proof systems [29]. Secondly, the implementation of POS BC systems should be considered. In this regard, we can rely on already existing POS protocols, for instance, Ouroboros [14], whose security-related features (e.g., consistency, chain quality, etc.) have been proven. More crucially, although it may seem that a user has to have access to the BC system to evaluate the circuit, we stress that two possibilities are available: (1) for a security-critical IP, where the user is not trusted, the BC access is essential, (2) if the user is trusted, or the protection of intellectual property is less vital, our one-time circuit scheme can be used to make the design (e.g., the bitstream for FPGAs) unlocked once and forever and disconnect the chip from the BC.

The second question would be whether the proposed schemes can be synthesized and integrated into the current, real-world chips, e.g., application-specific integrated circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). It has been shown that any circuit can be compiled in a very optimized way with current synthesis tools [24–26]. In addition, regarding the realization of garbled circuits, since a garbled IP does not need any specific logic rather than lookup tables to store the encrypted truth tables, it can be integrated into any ICs. In this regard, in the case of ASICs, a memory array as well as cryptoprocessors, capable of running symmetric and asymmetric ciphers, should be built along with other IP cores. The memory arrays can be programmed with encrypted truth tables during fabrication or later by the designer. Note that the truth tables are different for each chip instance, and hence, different encrypted values have to be stored on the memory arrays of each chip. On the other hand, in the case of FPGAs, this can be done in a simple manner as FPGAs are configured by a bitstream, which can contain arbitrarily garbled configurations, cf. [13]. In addition to the above discussion, in the second construction, a mechanism for storing the wire keys in the secure memory should be considered. To this end, a tamper-proof memory can be realized by non-volatile memories, such as flash or eFuses. Additionally, these memories can be configured to support one-time usage, by erasing a secure flash memory or burning an eFuse to make a rewrite operation into them almost impossible. However, an adversary with access to the advanced failure analysis equipment might still be able to reverse this process. Last but not least, note that both ASICs and FPGAs can also be configured to be connected to a network to transact on blockchain protocols.

Finally, while our first construction consumes more die area and its evaluation might take longer than the maximum time that some specific applications can tolerate, the second construction causes less overhead (i.e., die area and latency), but of course, offers only limited security. Construction 2 can be further optimized if the potential adversaries are foundries or IP integrators, but the user is trusted. For instance, if an IC is used in a satellite with no physical access, the garbled IP can be unlocked once and forever to further decrease the latency and power consumption of the circuit.

5 CONCLUSION

This paper forms an idea of how cryptographically-secure hardware obfuscation can be achieved by relying on the notion of Blockchain. We explain why neither self-destructing nor tamper-proof memory is required, in contrast to several celebrated existing methods. Furthermore, when such memories are available, we demonstrate how our scheme can be adapted to further offer security guarantees. These guarantees are based on concepts that are widely accepted in cryptography, namely, witness encryption and garbled circuits. Last but not least, we discuss the feasibility of our theoretical, abstract constructions in practice. We believe that the latter can largely contribute to the development of the knowledge and methodologies within the domain of hardware obfuscation.

REFERENCES

[1] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. 2006. Cryptography in NC^0. *SIAM J. on Computing* 36, 4 (2006), 845–888.

[2] Rajat Subhra Chakraborty and Swarup Bhunia. 2009. HARPOON: An Obfuscation-based SoC Design Methodology for Hardware Protection. *Trans. on Computer-Aided Design of Integrated Circuits and Systems* 28, 10 (2009), 1493–1502.

[3] Rajat Subhra Chakraborty and Swarup Bhunia. 2009. Security through Obscurity: An Approach for Protecting Register Transfer Level Hardware IP. In *Intrl. Workshop on Hardware-Oriented Security and Trust*. IEEE, 96–99.

[4] Rajat Subhra Chakraborty and Swarup Bhunia. 2010. RTL Hardware IP Protection using Key-based Control and Data Flow Obfuscation. In *Intrl. Conference on VLSI Design*. IEEE, 405–410.

[5] Brice Colombier, Lilian Bossuet, and David Hély. 2015. Reversible Denial-of-Service by Locking Gates Insertion for IP Cores Design Protection. In *Computer Society Annual Symp. on VLSI*. IEEE, 210–215.

[6] Giovanni Di Crescenzo, Jeyavijayan Rajendran, Ramesh Karri, and Nasir Memon. 2017. Boolean Circuit Camouflage: Cryptographic Models, Limitations, Provable

Results and a Random Oracle Realization. In *Proc. of the 2017 Workshop on Attacks and Solutions in Hardware Security*. ACM, 7–16.

[7] Sophie Dupuis and Marie-Lise Flottes. 2019. Logic Locking: A Survey of Proposed Methods and Evaluation Metrics. *Journal of Electronic Testing* (2019), 1–19.

[8] Domenic Forte, Swarup Bhunia, and Mark M Tehranipoor. 2017. *Hardware Protection through Obfuscation*. Springer.

[9] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. 2013. Witness Encryption and Its Applications. In *Proc. of the forty-fifth annual ACM Symp. on Theory of computing*. ACM, 467–476.

[10] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. 2008. One-time Programs. In *Annual Intrl. Cryptology Conf.* Springer, 39–56.

[11] Rishab Goyal and Vipul Goyal. 2017. Overcoming Cryptographic Impossibility Results Using Blockchains. In *Theory of Cryptography Conf.* Springer, 529–561.

[12] Benjamin Hill, Robert Karmazin, Carlos Tadeo Ortega Otero, Jonathan Tse, and Rajit Manohar. 2013. A Split-foundry Asynchronous FPGA. In *Proc. of the Custom Integrated Circuits Conference*. IEEE, 1–4.

[13] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. 2010. Garbled Circuits for Leakage-resilience: Hardware Implementation and Evaluation of One-time Programs. In *Intrl. Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 383–397.

[14] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Annual Intrl. Cryptology Conf.* Springer, 357–388.

[15] Farinaz Koushanfar. 2011. Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management. *Trans. on Information Forensics and Security* 7, 1 (2011), 51–63.

[16] Farinaz Koushanfar. 2012. Hardware Metering: A Survey. In *Introduction to Hardware Security and Trust*. Springer, 103–122.

[17] Jia Liu, Tibor Jager, Saqib A Kakvi, and Bogdan Warinschi. 2018. How to Build Time-lock Encryption. *Designs, Codes and Cryptography* 86, 11 (2018), 2549–2586.

[18] M Tanjidur Rahman, Qihang Shi, Shahin Tajik, Haoting Shen, Damon L Woodard, Mark Tehranipoor, and Navid Asadizanjani. 2018. Physical Inspection & Attacks: New Frontier in Hardware Security. In *3rd Intrl. Verification and Security Workshop (IVSW)*. IEEE, 93–102.

[19] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security Analysis of Integrated Circuit Camouflaging. In *Proc. of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 709–720.

[20] Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. 2013. Is Split Manufacturing Secure?. In *Proc. of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 1259–1264.

[21] Jarrod A Roy, Farinaz Koushanfar, and Igor L Markov. 2010. Ending Piracy of Integrated Circuits. *Computer* 43, 10 (2010), 30–38.

[22] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z Pan, and Yier Jin. 2017. AppSAT: Approximately Deobfuscating Integrated Circuits. In *Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*. IEEE, 95–100.

[23] Kaveh Shamsi, David Z Pan, and Yier Jin. 2019. On the Impossibility of Approximation-Resilient Circuit Locking. In *Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*. IEEE, 161–170.

[24] Ebrahim M Songhori, Siam U Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. 2015. Tinygarble: Highly Compressed and Scalable Sequential Garbled Circuits. In *Symp. on Security and Privacy*. IEEE, 411–428.

[25] Ebrahim M Songhori, M Sadegh Riazi, Siam U Hussain, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. 2019. ARM2GC: Succinct Garbled Processor for Secure Computation. *arXiv preprint arXiv:1902.02908* (2019).

[26] Ebrahim M Songhori, Thomas Schneider, Shaza Zeitouni, Ahmad-Reza Sadeghi, Ghada Dessouky, and Farinaz Koushanfar. 2016. Garbledcpu: a mips processor for secure computation in hardware. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[27] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the Security of Logic Encryption Algorithms. In *Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*. IEEE, 137–143.

[28] Yujie Wang, Pu Chen, Jiang Hu, Guofeng Li, and Jeyavijayan Rajendran. 2018. The Cat and Mouse in Split Manufacturing. *Trans. on Very Large Scale Integration (VLSI) Systems* 26, 5 (2018), 805–817.

[29] Hoeteck Wee. 2010. Efficient Chosen-ciphertext Security via Extractable Hash Proofs. In *Annual Cryptology Conf.* Springer, 314–332.

[30] Xiaolin Xu, Fahim Rahman, Bicky Shakya, Apostol Vassilev, Domenic Forte, and Mark Tehranipoor. 2019. Electronics Supply Chain Integrity Enabled by Blockchain. *ACM Trans. on Design Automation of Electronic Systems (TODAES)* 24, 3 (2019), 31.