

Blockchain-enabled Cryptographically-secure Hardware Obfuscation

Fatemeh Ganji¹ Shahin Tajik¹, Jean-Pierre Seifert², and Domenic Forte¹

¹ Florida Institute for Cybersecurity Research
University of Florida

² Security in Telecommunications
Technische Universität Berlin

Abstract. Among numerous applications, besides cryptocurrencies, the Blockchain offers inherent properties beneficial for the management of supply chains, where data is shared between trusted and untrusted parties. Electronics supply chain serves as a prime example of such chains, where one of the major players, i.e., a foundry, can be untrusted. Hardware obfuscation techniques, namely logic locking, and IC camouflaging have been developed to mislead an adversary aiming at reverse-engineering and Intellectual Property (IP) piracy. However, virtually all existing hardware obfuscation schemes developed over the last decade have been shown to be vulnerable to various attacks. The success of these attacks has been relying on either a lack of thorough, cryptographically-secure obfuscation schemes or an incorrect assumption widely made, i.e., the existence of an ideal tamper- and read-proof memory to store the key. To overcome these shortcomings, this paper proposes a novel, Blockchain-enabled, cryptographically-secure hardware obfuscation schemes being compatible with current circuit synthesis and fabrication tools. In this regard, rather than solely monitoring the supply chain via the Blockchain, the security of the obfuscation is guaranteed by Proof-of-Stack Blockchain protocols and witness encryption schemes. Furthermore, with the help of our construction, 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

Since the invention of the Blockchain technology, several real-world applications of that, beyond the cryptocurrency, have been proposed and developed. The spectrum of these applications covers various areas, where transparency, trust, and traceability are crucial, e.g., digital voting [23], smart contract [2, 8], and supply chain management [40, 36]. In the latter scenario, Blockchain-enabled

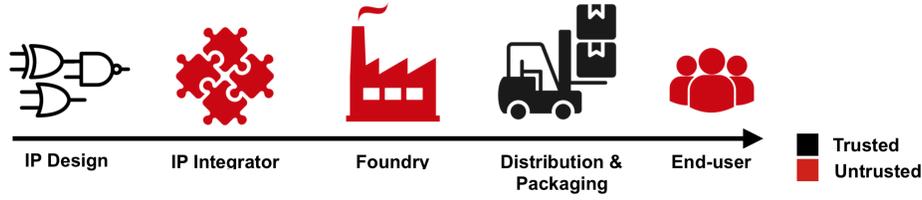


Fig. 1: The supply chain in semiconductor industry. It is assumed that the designer, packaging and distribution are trusted, while the IP integrator, foundry and end-users are untrusted.

management plays even a more important role for particular supply chains, for instance, the electronics supply chain. Over the years, electronic components and their supply chains have been considered secure and trustworthy. However, the 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 [19], where 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 Figure 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. Clearly, without ensuring the integrity of the electronics supply chain, tampered and counterfeit devices can pass through the chain and be employed in security-critical applications.

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, which are discussed briefly as follows.

A Brief Overview on Obfuscation Techniques: 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 [15]. The split manufacturing approach, although being a promising solution, has shortcomings that have been identified in the literature (e.g., [38]). 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 [28].

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 [10]. This type of approach can also

be applied at the gate-level as proposed in, e.g., [5, 4] and layout-level, see [6, 20]. 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 [29]. Logic masking [7] and logic permutation [11] 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 [27]. 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., [35]. In addition to this, modified, approximation-based versions of SAT attack, so-called APP-SAT, has been applied to de-obfuscate circuits effectively [30]. 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 [31]. 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 [25]. 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 “payer-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 [13]. Instead, we rely on inherent characteristics of the Blockchain 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 [14], the security of our scheme is related to the security of the Blockchain. This also explains the core difference between our work and [40, 16, 37], where the Blockchain 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., [9], our paper does not focus on a specific type of obfuscation approaches. However, both of our methods and one proposed by Crescenzo et al. [9] apply formal models as an enabler. While we employ the Blockchain cf. [14], indistinguishability obfuscation is considered in [9], as such models suggested for program obfuscation, the latter cannot adequately reflect the challenges confronting hardware obfuscation, as also mentioned in [9]. 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 Blockchain-enabled pay-per-use programs has been proven in the literature [14], we demonstrate how to adapt those results to achieve secure hardware obfuscation.

2 Building Blocks of Our Scheme and Adversary Model

Our scheme aims to offer cryptographically-secure obfuscation for circuits. In other words, for a physical circuit C_n , computing a function f on n inputs, our scheme can be employed to build a circuit C' , where an adversary with some resources (see Section 2.4) fails to extract the information required to perform reverse-engineering and IP piracy. In this regard, the core idea of our solution is to overcome the limitations of previous approaches by applying the notion of Blockchain in conjunction with witness encryption and garbled circuits, as described below.

2.1 Blockchains and Proof-of-Stake Protocols

In our scheme, Blockchains 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 Blockchains 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 [14]. Moreover, Blockchains 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”, Blockchain protocols can be categorized as Proof-of-Work (POW) and Proof-of-Stake (POS) [18]. 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 Blockchains provides assurance that a party (legitimate or malicious) can evaluate a circuit for only a limited time, depending on its balance [14].

2.2 Garbled Circuits

The notion of garbled circuits can be thought of as a 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 and

pay-per-use 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 or when paid by the user. Informally, after garbling a circuit C with n inputs, we obtain a garbled circuit, together with $2n$ wire keys. More precisely, for a family of circuits $\{C_n\}_n$, i.e., set of circuits C_n with n inputs, the garbling scheme is defined as polynomial-time algorithms **Garble** and **Eval**. As for the former one, $(G, \{w_i, b\}_{i \leq n, b \in \{0,1\}}) \leftarrow \mathbf{Garble}(1^\lambda, C \in C_n)$, i.e., given a security parameter λ and a circuit $C \in C_n$, the algorithm **Garble** outputs a garbled circuit G along with $2n$ wire keys $\{w_i, b\}_{i \leq n, b \in \{0,1\}}$. On the other hand, when these outputs are fed into the algorithm **Eval**, we obtain $y \in \{0,1\}$, cf. [3]. Put differently, when evaluating a garbled circuit on given key and input (see Section 4 for more details), given the same input, the output of the garbled circuit is the same as the output of our circuit C (our circuit before garbling). This property is associated with the “correctness” of a garbling scheme. The security of such a scheme is related to the fact that during the evaluation of the garbled circuit, the information neither on circuit C nor on the input is revealed³.

It is straightforward to observe that although the above encoding fulfills our requirements in terms of correctness and security, it does not meet the characteristics of the one-time and pay-per-use circuits. To address this, the notion of witness encryption, as defined below, can be helpful.

2.3 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 [12]. For instance, suppose that the decryption is possible if a solution to an NP-hard puzzle is known.

Formally, consider the witness relation R , that is $R : \{0,1\}^* \times \{0,1\}^* \rightarrow \{\mathbf{true}, \mathbf{false}\}$ so that $R(x) = \{w : R(x, w)\}$, where $x \in \{0,1\}^*$. Clearly, **true** and **false** can be denoted by “1” and “0”, respectively. Now, for a language L with witness relation R , i.e., an efficiently computable the witness encryption, i.e., **WE** = (**Enc**, **Dec**) over the message space $M \subseteq \{0,1\}^*$, is composed of two polynomial-time algorithms **Enc** and **Dec** with the following properties. For a given message $m \in M$ and a string $x \in \{0,1\}^*$, the algorithm $\mathbf{Enc}(1^\lambda, x, m)$ outputs a ciphertext **ct**, where λ is the security parameter. Applying the decryption algorithm $\mathbf{Dec}(\mathbf{ct}, w \in \{0,1\}^*)$ on the ciphertext and a witness string w , it delivers the message $m \in M$ or \perp , which is a special symbol denoting that the decryption fails.

³ To prove this, it is inevitable to follow a simulation-based formulation of security. In this sense, informally, a simulator in an alternative, secure-by-definition world is constructed, which generates a view of the adversary in the real world. This view should be computationally indistinguishable from the real one [21].

The above WE scheme is correct if for all λ , $(x,w) \in R$, $m \in M$, we have $m = \text{Dec}(\text{ct}, w)$ with $\text{ct} \leftarrow \text{Enc}(1^\lambda, x, m)$. The security for a WE is defined with regard to the notion of “extractable security” stating that only if an adversary knows a witness for the instance used by the encryption algorithm, she can learn some non-trivial information about the encrypted message. Definition 1 formulates this precisely.

Definition 1 *The WE scheme WE is extractable secure, if for all security parameter λ , PPT adversary A , and a polynomial $p(\cdot)$, a PPT extractor E with a polynomial $q(\cdot)$ can be found that satisfies the following. For every pair of messages $m_0, m_1 \in M$ and a string $x \in \{0,1\}^*$ we have*

$$\Pr[A(1^\lambda, \text{ct}) = b \mid b \leftarrow \{0,1\}; \text{ct} \leftarrow \text{Enc}(1^\lambda, x, m_b)] \geq \frac{1}{2} + \frac{1}{p(\lambda)},$$

and then,

$$\Pr[(x,w) \in R \mid w \leftarrow E(1^\lambda, x, m_0, m_1)] \geq \frac{1}{2} + \frac{1}{q(\lambda)}.$$

In our scheme, such witness is the existence of users’ blocks in the Blockchain. This is possible thanks to the witness relation on the Blockchain protocol defined based on the uniqueness of the local states of parties and their transaction over the Blockchain [14]. Especially for the pay-per-use application, there should be evidence showing that a pre-specified amount of cryptocurrency is transferred from the user to the IP owner (i.e., service provider). We further elaborate on this in Section 3.2.

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. [9, 31]. 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 adversary⁴ and the Blockchain, we assume that the adversary has complete access to the Blockchain and can possibly have a malicious influence on the protocol execution by mining blocks or

⁴ Needless to say that by the term adversary, we refer to the above PPT algorithm controlling all the corrupt parties.

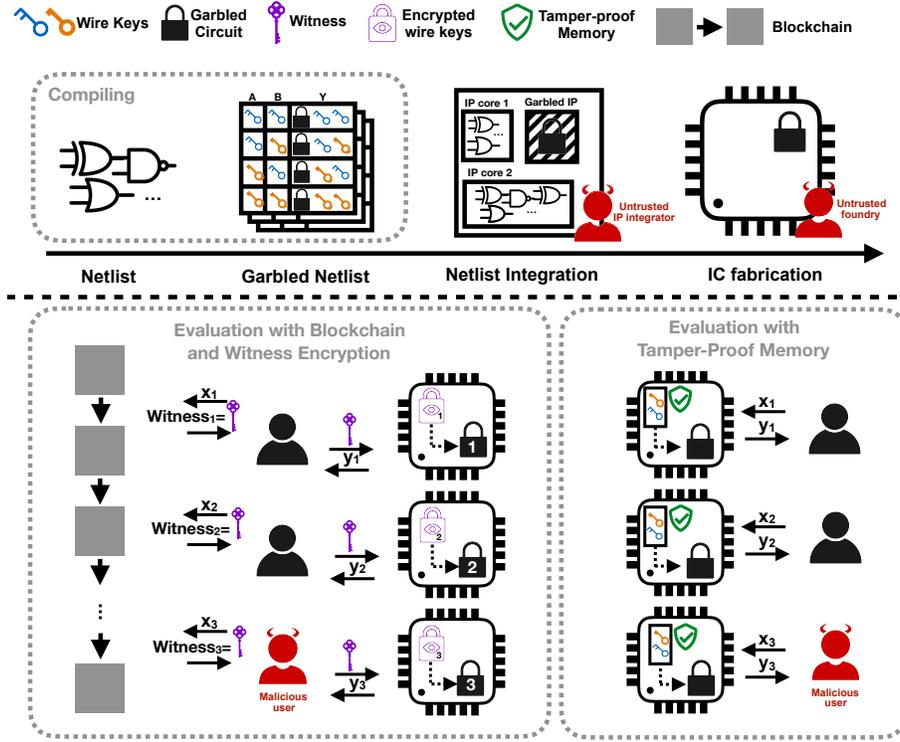


Fig. 2: Generating garbled IPs and their deployment in the field. The wire keys can decrypt the garbled IP for evaluation. Construction 1 (Evaluation with Blockchain and WE) stores encrypted wire keys on the chip and uses Blockchain to get a witness along with the current state of the user to decrypt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

deviating from the protocol, cf. [14]. 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.

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 Blockchains equipped with WE. Nevertheless, if tamper-proof hardware is used, another construction (Construction 2) can be built exhibiting provably-secure obfuscation. We further compare these two

Algorithm 1 Function `Compile` included in the protocol underlying the Construction 1.

Require: Access to a Blockchain protocol Γ with validity predicate V and distinguishable forking property-related parameters $(\beta, \ell_1, \ell_2, n)$ and a WE scheme $\text{WE} = (\text{Enc}, \text{Dec})$, security parameter λ , a public identity id , the amount of stake $q \in \mathbb{Q}$ belonging to id , and the circuit $C \in C_n$.

Ensure: A compiled circuit CC

```

1:  $(G, \{w_{i,b}\}_{i \leq n, b \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, C \in C_n)$ .
2: while  $i \leq n$  do
3:   for  $b \in \{0, 1\}$  do
4:      $x_{i,b} = (1^\lambda, \text{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \text{id}, q)$ . ▷ st: the local state.
5:      $\text{ct}_{i,b} \leftarrow \text{Enc}(1^\lambda, x_{i,b}, w_{i,b})$ .
6:   end for
7: end while
8:  $x_0 = (1^\lambda, \text{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \text{id}, q)$ .
9:  $\text{ct}_0 \leftarrow \text{Enc}(1^\lambda, x_0, G)$ .
10:  $CC = (1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \text{id}, q, \text{ct}_0, \{\text{ct}_{i,b}\}_{i \leq n, b \in \{0,1\}})$ .
11: return  $CC$ 

```

constructions and demonstrate why the Blockchain-enabled one outperforms the obfuscation scheme, where tamper-proof hardware is integrated into the scheme. In both constructions, the Boolean circuit of one or more IPs are garbled, and the wire keys associated with them are generated locally, see Figure 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 [14], although we adopt those to the context of hardware obfuscation.

3.1 Construction 1

As explained above, this construction is built around Blockchain as an enabler. To elaborate on how Construction 1 works, we begin with the definition of our Blockchain protocol as the main building block. The Blockchain protocol Γ is composed of the following polynomial-time algorithms: (1) `UpdateState` (1^λ) that maintains a local state st ⁵, given the security parameter λ , (2) a sequence of valid blocks contained in st is delivered by the algorithm `GetRecords` $(1^\lambda, \text{st})$, where each of them includes a sequence of records/ messages m , and (3) to send the message m to all nodes executing the protocol Γ , `Broadcast` $(1^\lambda, m)$ is used. With this protocol, we also associate the validity predicate V associated with the application of the Blockchain protocol, which indicates if a sequence of blocks \mathbf{B} is valid by outputting 1, and vice verse [24].

Inspired by the design of per-pay-use programs [14], our construction encompasses two main algorithms `Compile` and `Eval`, whose main steps are depicted

⁵ As for the local state, the entire Blockchain should be considered [14].

Algorithm 2 Function `Eval` included in the protocol underlying the Construction 1.

Require: Access to a Blockchain protocol Γ with validity predicate V and distinguishable forking property-related parameters $(\beta, \ell_1, \ell_2, n)$ and a WE scheme $\text{WE} = (\text{Enc}, \text{Dec})$, security parameter λ , a public identity id , the amount of stake $q \in \mathbb{Q}$ belonging to id , the compiled circuit CC , and the input to be evaluated $y \in \{0,1\}^n$.

Ensure: $\text{Eval}(G, \{w_i\}_{i \leq n})$.

```

1:  $m = (\widetilde{\text{id}}, \text{id}, q, \text{aux} = y)$ .
2: Broadcast( $1^\lambda, m$ ).
3: UpdateState ▷ Wait for the chain to be extended by  $\ell_1 + \ell_2$  blocks
4:  $G = \text{Dec}(\text{ct}_0, \text{st})$ .
5: while  $i \leq n$  do
6:    $w_i = \text{Dec}(\text{ct}_{i,y_i}, \text{st})$ .
7:   if Fail then
8:     return  $\perp$ 
9:   else
10:    return  $\text{Eval}(G, \{w_i\}_{i \leq n})$ 
11:   end if
12: end while

```

in Algorithms 1-2. To perform the protocol, first and foremost, the IP owner defines a unique identity ($\text{id} \in \{0,1\}^*$) for each IC. Note that the security of our scheme *does not* depend on the security of this id and it can be public. The process of committing a circuit over the Blockchain \mathbf{B} , i.e., running the `Compile` algorithm, begins with garbling the circuit resulting in a garbled circuit and wire keys, as described in Section 2.2. 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 [14]. 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 Figure 2.

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 \mathbf{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 (see Algorithm 2), the user first broadcast a message m to other nodes executing the protocol. This message demonstrates that the user with the public identity id transfers q coins to the IP owner $\widetilde{\text{id}}$ in the hope of evaluating the circuit on the input y . After posting this message

and waiting till the Blockchain is extended by $\ell_1 + \ell_2$ blocks, the local state of the user \mathbf{st} can be used to decrypt the garbled circuit and wire keys. To this end, as a *witness* the user has to generate a Blockchain \mathbf{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 the above description as well as Algorithms 1- 2 corresponds to the pay-per-use applications. To achieve one-time and pay-per-use circuits, these algorithms can be adopted and modified slightly: instead of transferring the coins along with the desired input, the user sends the garbled circuit G and the input to the IP owner to evaluate it.

3.2 Security and Correctness

When it comes to assessing the security of the protocol mentioned above, it can be thought that the adversary may extend the Blockchain to impair the effectiveness of the protocol. Note that even if a malicious Blockchain $\tilde{\mathbf{B}}$ is extended, the adversary still has to deal with the garbled circuit. Besides, the parameters of the Blockchain can be defined so that such malicious extension can be distinguished. We elaborate on this point in the next subsection; however, before that, we expand on a principle providing the basis for the security and correctness proofs. These proofs rely heavily on the notion of extractable secure WE⁶. An example of how the WE can be helpful is related to avoiding multiple inputs simultaneously committed to the Blockchain. To this end, to ensure that the user commits only one input over the Blockchain, our scheme naturally involves a mechanism to check the witness \mathbf{B}' .

The importance of defining a witness relation has been already stressed in Section 2.3. Here, we provide more detail on how such a relation can be defined based on the properties of a Blockchain protocol. Consider the relation R_Γ associated with the Blockchain protocol Γ , where $x = (1^\lambda, \mathbf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathbf{id}, q)$ and $w = \tilde{\mathbf{st}}$. Clearly, $\mathbf{B} = \text{GetRecords}(1^\lambda, \mathbf{st})$ and $\tilde{\mathbf{B}} = \text{GetRecords}(1^\lambda, \tilde{\mathbf{st}})$ can be defined. For w , to provide witness to x in the sense of R_Γ , the following should hold. Both of the Blockchains \mathbf{B} and $\tilde{\mathbf{B}}$ should be valid and consistent. Additionally, for a given unique message $m = (\mathbf{id}, \mathbf{id}, q, \mathbf{aux} = y)$, there is no block with a record of m unless it is in $\tilde{\mathbf{B}}$ (assuming that \mathbf{B} is a prefix of $\tilde{\mathbf{B}}$). And, after this unique block in $\tilde{\mathbf{B}}$, the Blockchain is extended by $\ell' \geq \ell_1 + \ell_2$ blocks with sufficient fraction of stake, greater than β .

According to the above definition, it is evident that the security of the entire protocol depends on how the parameters of the Blockchain are set, as discussed below.

⁶ As the proofs of security and correctness of our protocol are similar to pay-per-use programs presented in [14], we refer the reader to that for more details.

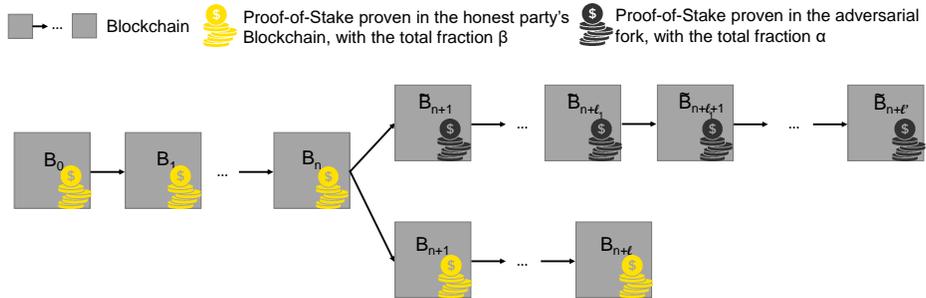


Fig. 3: A Blockchain with forking distinguishability property. In our setting, the number of honest parties is n . The length of the maliciously generated fork is ℓ' , and its minimum length is $\ell_1 + \ell_2$: $\ell' \geq \ell_1 + \ell_2$ (the first part of the fork is denoted by ℓ_1). The length 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 [14].

How to Set Parameters Related to POS Blockchain to Achieve the Secure Scheme? First, we again put emphasis on the role of the Blockchain. In our framework, the purpose of the Blockchain is to offer a platform, upon which we construct a provably-secure scheme. To base such a platform on the POS Blockchain, 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 Blockchain, cf. [14]. More precisely, with high probability, we can distinguish an individual, invalid chain of blocks generated by an adversary from the honest parties' Blockchain, see Figure 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 Blockchain-based scheme [14]. 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., [18].

Finally, to draw a conclusion of this section, we stress that the one-time secrecy and security against a malicious extension of the Blockchain is reduced to the security of the Blockchain (e.g., chain consistency, etc.) equipped with the WE and security of garbling scheme, respectively (see [14] for the proofs). This point, as well as how existing approaches may achieve the security-related requirements, have been summarized in Table 1.

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

* When equipped with self-destructing memory

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

3.3 Construction 2

The purpose of this construction is to provide a comparison between our Blockchain-enabled obfuscation scheme and conventional ones. In this regard, for the second construction, we assume the existence of a tamper- and read-proof memory on the chip (see Figure 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 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 common 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 construction still deploys garbled circuits to obfuscate the netlist, which is cryptographically secure. Nevertheless, when compared to our Blockchain-enabled construction, Construction 2 cannot achieve the same security level. More specifically, although the existence of a tamper- and read-proof memory makes the design of a scheme more straightforward, such a scheme is vulnerable to physical attacks, similar to conventional logic locking schemes [26].

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 Blockchain system. This has been already discussed and proposed in the literature [22]. 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 [39]. Secondly, the implementation of POS

Blockchain systems should be considered. In this regard, we can rely on already existing POS protocols, for instance, Ouroboros [18], 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 Blockchain 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 Blockchain 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 Blockchain.

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 [32, 34, 33]. 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 crypto-processors, 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. [17]. 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 decrease the latency and power consumption of the circuit further.

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. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0 . *SIAM J. on Computing* **36**(4), 845–888 (2006)
2. Bartoletti, M., Pompianu, L.: An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns. In: *Intr. Conf. on Financial Cryptography and Data Security*. pp. 494–509. Springer (2017)
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of Garbled Circuits. In: *Proc. of the 2012 ACM Conf. on Computer and communications security*. pp. 784–796. ACM (2012)
4. Chakraborty, R.S., Bhunia, S.: HARPOON: An Obfuscation-based SoC Design Methodology for Hardware Protection. *Trans. on Computer-Aided Design of Integrated Circuits and Systems* **28**(10), 1493–1502 (2009)
5. Chakraborty, R.S., Bhunia, S.: Security through Obscurity: An Approach for Protecting Register Transfer Level Hardware IP. In: *Intr. Workshop on Hardware-Oriented Security and Trust*. pp. 96–99. IEEE (2009)
6. Chakraborty, R.S., Bhunia, S.: RTL Hardware IP Protection using Key-based Control and Data Flow Obfuscation. In: *Intr. Conference on VLSI Design*. pp. 405–410. IEEE (2010)
7. Colombier, B., Bossuet, L., Hély, D.: Reversible Denial-of-Service by Locking Gates Insertion for IP Cores Design Protection. In: *Computer Society Annual Symp. on VLSI*. pp. 210–215. IEEE (2015)
8. Delmolino, K., Arnett, M., Kosba, A., Miller, A., Shi, E.: Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. In: *Intr. Conf. on Financial Cryptography and Data Security*. pp. 79–94. Springer (2016)
9. Di Crescenzo, G., Rajendran, J., Karri, R., Memon, N.: 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*. pp. 7–16. ACM (2017)
10. Dupuis, S., Flottes, M.L.: Logic Locking: A Survey of Proposed Methods and Evaluation Metrics. *Journal of Electronic Testing* pp. 1–19 (2019)
11. Forte, D., Bhunia, S., Tehranipoor, M.M.: *Hardware Protection through Obfuscation*. Springer (2017)

12. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness Encryption and Its Applications. In: Proc. of the forty-fifth annual ACM Symp. on Theory of computing. pp. 467–476. ACM (2013)
13. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time Programs. In: Annual Intrl. Cryptology Conf. pp. 39–56. Springer (2008)
14. Goyal, R., Goyal, V.: Overcoming Cryptographic Impossibility Results Using Blockchains. In: Theory of Cryptography Conf. pp. 529–561. Springer (2017)
15. Hill, B., Karmazin, R., Otero, C.T.O., Tse, J., Manohar, R.: A Split-foundry Asynchronous FPGA. In: Proc. of the Custom Integrated Circuits Conference. pp. 1–4. IEEE (2013)
16. Islam, M.N., Patii, V.C., Kundu, S.: On IC Traceability via Blockchain. In: Intrl. Symp. on VLSI Design, Automation and Test. pp. 1–4. IEEE (2018)
17. Järvinen, K., Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Garbled Circuits for Leakage-resilience: Hardware Implementation and Evaluation of One-time Programs. In: Intrl. Workshop on Cryptographic Hardware and Embedded Systems. pp. 383–397. Springer (2010)
18. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In: Annual Intrl. Cryptology Conf. pp. 357–388. Springer (2017)
19. Koushanfar, F.: Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management. *Trans. on Information Forensics and Security* **7**(1), 51–63 (2011)
20. Koushanfar, F.: Hardware Metering: A Survey. In: Introduction to Hardware Security and Trust, pp. 103–122. Springer (2012)
21. Lindell, Y.: How to Simulate It—A Tutorial on the Simulation Proof Technique. In: Tutorials on the Foundations of Cryptography, pp. 277–346. Springer (2017)
22. Liu, J., Jager, T., Kakvi, S.A., Warinschi, B.: How to Build Time-lock Encryption. *Designs, Codes and Cryptography* **86**(11), 2549–2586 (2018)
23. McCorry, P., Shahandashti, S.F., Hao, F.: A Smart Contract for Boardroom Voting with Maximum Voter Privacy. In: Intrl. Conf. on Financial Cryptography and Data Security. pp. 357–375. Springer (2017)
24. Pass, R., Seeman, L., Shelat, A.: Analysis of the Blockchain Protocol in Asynchronous Networks. In: Annual Intrl. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 643–673. Springer (2017)
25. Rahman, M.T., Shi, Q., Tajik, S., Shen, H., Woodard, D.L., Tehranipoor, M., Asadizanjani, N.: Physical Inspection & Attacks: New Frontier in Hardware Security. In: 3rd Intrl. Verification and Security Workshop (IVSW). pp. 93–102. IEEE (2018)
26. Rahman, M.T., Tajik, S., Rahman, M.S., Tehranipoor, M., Asadizanjani, N.: The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes. *Cryptology ePrint Archive, Report 2019/719* (2019), <https://eprint.iacr.org/2019/719> [Accessed on Sept. 23, 2019]
27. Rajendran, J., Sam, M., Sinanoglu, O., Karri, R.: Security Analysis of Integrated Circuit Camouflaging. In: Proc. of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 709–720. ACM (2013)
28. Rajendran, J.J., Sinanoglu, O., Karri, R.: Is Split Manufacturing Secure? In: Proc. of the Conference on Design, Automation and Test in Europe. pp. 1259–1264. EDA Consortium (2013)
29. Roy, J.A., Koushanfar, F., Markov, I.L.: Ending Piracy of Integrated Circuits. *Computer* **43**(10), 30–38 (2010)

30. Shamsi, K., Li, M., Meade, T., Zhao, Z., Pan, D.Z., Jin, Y.: AppSAT: Approximately Deobfuscating Integrated Circuits. In: *Intr. Symp. on Hardware Oriented Security and Trust (HOST)*. pp. 95–100. IEEE (2017)
31. Shamsi, K., Pan, D.Z., Jin, Y.: On the Impossibility of Approximation-Resilient Circuit Locking. In: *Intr. Symp. on Hardware Oriented Security and Trust (HOST)*. pp. 161–170. IEEE (2019)
32. Songhori, E.M., Hussain, S.U., Sadeghi, A.R., Schneider, T., Koushanfar, F.: Tinygarble: Highly Compressed and Scalable Sequential Garbled Circuits. In: *Symp. on Security and Privacy*. pp. 411–428. IEEE (2015)
33. Songhori, E.M., Riazi, M.S., Hussain, S.U., Sadeghi, A.R., Koushanfar, F.: ARM2GC: Succinct Garbled Processor for Secure Computation. *arXiv preprint arXiv:1902.02908* (2019)
34. Songhori, E.M., Schneider, T., Zeitouni, S., Sadeghi, A.R., Dessouky, G., Koushanfar, F.: Garbledcpu: a mips processor for secure computation in hardware. In: *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*. pp. 1–6 (2016)
35. Subramanyan, P., Ray, S., Malik, S.: Evaluating the Security of Logic Encryption Algorithms. In: *Intr. Symp. on Hardware Oriented Security and Trust (HOST)*. pp. 137–143. IEEE (2015)
36. Tian, F.: An Agri-food Supply Chain Traceability System for China Based on RFID & Blockchain Technology. In: *13th Intr. Conf. on Service Systems and Service Management*. pp. 1–6. IEEE (2016)
37. Toyoda, K., Mathiopoulos, P.T., Sasase, I., Ohtsuki, T.: A Novel Blockchain-based Product Ownership Management System (POMS) for Anti-counterfeits in the Post Supply Chain. *IEEE Access* **5**, 17465–17477 (2017)
38. Wang, Y., Chen, P., Hu, J., Li, G., Rajendran, J.: The Cat and Mouse in Split Manufacturing. *Trans. on Very Large Scale Integration (VLSI) Systems* **26**(5), 805–817 (2018)
39. Wee, H.: Efficient Chosen-ciphertext Security via Extractable Hash Proofs. In: *Annual Cryptology Conf.* pp. 314–332. Springer (2010)
40. Xu, X., Rahman, F., Shakya, B., Vassilev, A., Forte, D., Tehranipoor, M.: Electronics Supply Chain Integrity Enabled by Blockchain. *ACM Trans. on Design Automation of Electronic Systems (TODAES)* **24**(3), 31 (2019)