Publicly Verifiable Proofs from Blockchains

Alessandra Scafuro[1], Luisa Siniscalchi[2(✉)], and Ivan Visconti[2]

[1] NCSU, Raleigh, USA
ascafur@ncsu.edu
[2] DIEM, University of Salerno, Fisciano, Italy
{lsiniscalchi,visconti}@unisa.it

Abstract. A proof system is publicly verifiable, if anyone, by looking at the transcript of the proof, can be convinced that the corresponding theorem is true. Public verifiability is important in many applications since it allows to compute a proof only once while convincing an unlimited number of verifiers.

Popular interactive proof systems (e.g., Σ-protocols) protect the witness through various properties (e.g., witness indistinguishability (WI) and zero knowledge (ZK)) but typically they are not publicly verifiable since such proofs are convincing only for those verifiers who contributed to the transcripts of the proofs. The only known proof systems that are publicly verifiable rely on a non-interactive (NI) prover, through trust assumptions (e.g., NIZK in the CRS model), heuristic assumptions (e.g., NIZK in the random oracle model), specific number-theoretic assumptions on bilinear groups or relying on obfuscation assumptions (obtaining NIWI with no setups).

In this work we construct publicly verifiable witness-indistinguishable proof systems from any Σ-protocol, based *only* on the existence of a very generic blockchain. The novelty of our approach is in enforcing a non-interactive verification (thus guaranteeing public verifiability) while allowing the prover to be interactive and talk to the blockchain (this allows us to circumvent the need of strong assumptions and setups). This opens interesting directions for the design of cryptographic protocols leveraging on blockchain technology.

1 Introduction

Blockchains are a surprising reality. Bitcoin, Ethereum, Cardano, Ripple, Zcash etc. [3,9,19,45,49] are all examples of permissionless[1] blockchains used to implement a cryptocurrency. Above all, Bitcoin [45] was the first cryptocurrency and

A. Scafuro—Work supported by NSF grant # 1012798.
L. Siniscalchi and I. Visconti—Research supported in part by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780477 (project PRIViLEDGE) and in part by "GNCS - INdAM".

[1] In the remaining of the paper we will omit the adjective "permissionless" since this work focuses on the permissionless setting only.

D. Lin and K. Sako (Eds.): PKC 2019, LNCS 11442, pp. 374–401, 2019.
https://doi.org/10.1007/978-3-030-17253-4_13

the first decentralized blockchain, and recently has celebrated 10 years of life. From a technical point of view, the robustness achieved by Bitcoin – which is a completely decentralized system developed by the voluntary effort of a large community – has motivated the cryptographic community to study the underlying consensus protocol in order to rigorously define what security properties it actually achieves and under which assumptions on the adversary [1,2,24,46]. Specifically, the works by Garay et al. [24] and Pass et al. [46] identify three properties achieved by the Bitcoin backbone protocol: *consistency*, which means that any two honest parties should share the same view of the blockchain, up to T blocks; *chain growth*, which means that the blockchain, as seen by the honest parties, will grow with a steady rate; and *chain quality*, which states that for any sequence of consecutive blocks, at least a fraction of them are contributed by honest parties. Such security properties have been adopted in all subsequent blockchain designs [2,34,47,48], enforcing the intuition that any blockchain protocol, taken as a black box, must guarantee them.

In this paper, we investigate how to leverage the sole assumption that a blockchain exists to achieve cryptographic tasks that do not seem possible without trust assumptions, heuristic security, strong computational assumptions or specific number-theoretic assumptions.

Publicly Verifiable Witness-Indistinguishable Proofs[2] *(of Knowledge).* We look at the problem of achieving "privacy-preserving" but still *publicly verifiable* proof systems. In a proof system a prover P wishes to convince a verifier V that a statement $x \in L$ is true, where L is an NP language. A proof system is privacy-preserving if the transcript of the proof somehow protects the privacy of the witness used in the proof, that is, it satisfies a witness hiding/indistinguishability (WH, WI) property [21] or zero-knowledge property [28]. A proof system is *publicly verifiable*, if any one, by looking at the transcript of the proof, can be convinced that the theorem is true. The verification procedure is therefore non-interactive. Public verifiability is useful in many settings where a prover would like to reuse the same proof with many verifiers, or in general when we want the proof to be transferable.

Public verifiability and witness hiding/indistinguishability are two important properties that are easy to achieve separately. To achieve public verifiability, ignoring any protection for the witness, one can simply publish the witness. If instead only witness hiding/indistinguishability is desired, ignoring public verifiability, there is a rich body of literature that explores many constructions, under several assumptions and for various languages. For example, WI proof systems for all NP are known from minimal assumptions [21,23], and various Σ-protocols [17] for specific languages, such as the language of DDH tuples [16,50] are WH/WI[3].

[2] In the introduction, informally we will generically use the word "proof" to refer also to *computationally sound* proofs [44].

[3] Every perfect special honest-verifier zero-knowledge (SHVZK) is WI [16]. If a Σ-protocol is computational SHVZK, then it could not enjoy the WI property [11],

Instead, achieving public verifiability *and* witness indistinguishability at the same time, is very non-trivial. In particular, any *interactive* witness indistinguishable proof, is intrinsically not publicly verifiable, since no one, besides the verifier who chooses the messages in the protocol, can be guaranteed that the prover did not know the messages in advance and thus believe in the validity of the proof. If the WI proof system is public coin, one could use the Fiat-Shamir transform [22] and replace the messages of the verifier with the output of the random oracle. However, this is an heuristic assumption that we wish to avoid towards providing publicly verifiable WI proof systems. We also would like to avoid trusted setups that have been widely used to get NIZK [6,13,21,31,38–41]. A relaxed trusted setup was used by [30] where there are multiple common reference strings and a majority of them is required to be honest. While such assumption is more realistic, we notice that the construction of [30] is based on a setup that does not reflect what is available in the real world. Our goal is to end up with publicly verifiable proofs that can be run exploiting a generic blockchain as setup.

The above discussion seemingly suggests that for a WI proof to be publicly verifiable, it must be either non-interactive. The only known non-interactive witness-indistinguishable proof systems without trusted setups and heuristic assumptions are due to:

- Groth, Ostrovsky and Sahai [31], and is based on specific number-theoretic hardness assumptions in bilinear groups. Such a scheme is not a proof of knowledge (for some languages, membership of an instance is trivially checkable by inspection, as for the case of knowledge of one out of two discrete logarithms, and what really matters is to make sure that a succeeding prover always knows a witness proving the truthfulness of the theorem).
- Bitansky and Paneth [5], and is based on indistinguishable obfuscation [26] and one-way permutations. In particular, their construction leverages the existence of witness encryption schemes [27] and the existence of ZAPs [18]. As in [31], the proposed approach does not provide the proof of knowledge property.

This is somewhat unsatisfactory. Given that we have a rich portfolio of interactive WI proof systems (and a large part of it consists of Σ-protocols), under various (weaker) complexity assumptions, optimized for different languages and that provide also proof-size optimizations, we would like to use these systems also when public verifiability is required. In this paper we ask the following question:

Can we construct a publicly-verifiable WI proof system given any Σ-protocol, by leveraging only the existence of a blockchain, and without any additional assumption?

however [25] shows that the OR-composition of computational SHVZK Σ-protocols is WI when all involved instances are true.

Goyal and Goyal in [29] proposed to use specific blockchains to construct a non-interactive zero-knowledge proof of knowledge. They do so by assuming the existence of a non-interactive WI proof system in the standard model (therefore inheriting all the limitations discussed above) that they use in conjunction with the assumption that the underlying blockchain is based on a proof-of-stake (PoS) consensus protocol. Their construction crucially leverages the PoS setting, and in addition also imposes other specific requirements on the cryptographic primitives used in the underlying consensus protocol (i.e., they require that the blockchain protocol uses a signature scheme which keys can be used also in a CPA-secure encryption scheme).

Our Contribution. As main contribution of this work we show how to construct a publicly verifiable WI proof systems from any Σ-protocol essentially assuming only the chain quality property of any blockchain[4].

We relax the connection between non-interactiveness and public verifiability by proposing a novel approach which consists in having an execution of a Σ-protocol where the prover is interactive, but the verifier's message is somehow played by the blockchain, making the verification process completely non-interactive for anyone who has access to the blockchain. If the underlying Σ-protocol additionally satisfies the delayed-input property (that is, a prover can compute the first round without knowing the theorem that she will prove, then our proof system allows preprocessing of the first message, and the actual proof can be computed in one-shot, therefore is completely non-interactive (modulo just one round of offline preprocessing done by the prover without knowing which instance will be proven and when). We also discuss the case of on-chain and off-chain verifiability depending on the blocksize supported by the blockchain and the communication efficiency of the underlying Σ-protocol.

Additionally, we observe that our publicly verifiable WI could be used in the construction of [29] to obtain a publicly verifiable ZK proof of knowledge with improved complexity assumptions relying on PoS blockchains. While the above observation might look an interesting improvement over the state-of-the art, more interestingly as additional contribution in this work we discuss some issues in the approach of [29] that can seemingly be addressed only by relying on additional assumptions (possibly implicit in [29] but that we believe it is worthy to make explicit).

1.1 Our Techniques

In order to construct publicly verifiable WI proofs we start with any Σ-protocol [17]. A Σ-protocol is a 3-round public-coin proof system that satisfies special soundness and special honest-verifier zero-knowledge (HVZK) properties, where the first and third round are computed by the prover and the second round

[4] The actual assumption is a bit different but is essentially captured by the chain quality property and some natural requirements that are seemingly satisfied by known blockchains.

is a random string sent by the verifier. A transcript (a, c, z) of a Σ-protocol is not publicly verifiable, indeed, due to the special HVZK property, anyone could come up with an accepting transcript by choosing c on its own. Our goal is to compute c in a verifiable manner, *without relying on the random oracle*, but simply leveraging the properties of *any* blockchain.

Challenge 1: Extracting Random Bits from Any Blockchain. Several works [4,7] have investigate the possibility of implementing a publicly verifiable beacon from the bitcoin blockchain. In particular, Bentov, Gabizon and Zuckerman in [4] show that, under stronger assumptions on the adversary (i.e., assuming that the adversary would not too often discard blocks that she computed), it is possible to extract unbiased and publicly verifiable bits from the bitcoin blockchain and thus realize a publicly verifiable beacon. Their result is somehow unsatisfactory for our goals since (1) it is tailored to bitcoin, (2) it makes additional assumptions on the adversary beyond the generic properties of a blockchain.

Our observation is that for our purposes we do not need the strong guarantees required by a publicly verifiable beacon. In particular, we don't need to precisely identify which string is random, we only need to ensure that, within a long string of bits, there exists a subsequence of λ bits that is sufficiently unpredictable to the adversary. In other words, in our setting, we can relax the requirement that the challenge c is a string of λ random bits, and instead consider c to be a much longer string composed by τ substrings $c_1, \ldots, c_\tau$, and the guarantee is that some of the substrings have sufficient min-entropy and are independently generated.

This relaxation allows us to extract enough random bits by essentially only assuming that the blockchain satisfies a property that is very similar to the η-chain quality property defined in [46]. Recall that η-chain quality states that for any K consecutive blocks in any chain held by some honest party, the fraction of blocks that were contributed by honest parties is at least η, with overwhelming probability in K. Our assumption is similar in the sense that we additionally observe that a block generated by an honest party must contain strings with high min-entropy, at the very least for the cryptographic material required to generate a block that must be unpredictable to an adversary (e.g., a wallet identifier used by the miner to cash the reward). More specifically, for each block created by a honest party, we identify a field that contains high min-entropy material (and discard the data concerned the transactions since they could have 0 entropy). We only need to assume that a constant number of blocks in a long enough sequence of blocks are computed by distinct honest parties (or even the same party as long as the special field is computed with independent randomness), then these chunks of the blocks can legitimately be considered as independent sources of randomness. Putting the above things together, we can think of K consecutive blocks as K potential sources of randomness, out of which a constant η (notice that we don't need a constant fraction, but just a constant) are guaranteed to be independent and have high min-entropy.

Our idea is therefore to leverage the above observations along with a multiple-source randomness extractor. The 3-source randomness extractor of Lin [36], given in input 3 high min-entropy independent sources, outputs a λ-bit truly

random string. By using the η-chain quality property and the observation we made about honest blocks, our goal is to retrieve 3 honest blocks on which to apply the extractor. Since we don't know which blocks are honest, we will just consider all possible $\binom{K}{3}$ triples of distinct blocks over last K blocks of the blockchain. The η-chain quality guarantees that at least 3 of them are honest. We are guaranteed that there exists a triple of honest block chunks that are independent high min-entropy sources (we stress again, that we will consider only certain strings of the blocks that contain high min-entropy information, we also note that such strings have to be sufficiently long and have sufficient min-entropy for the output of the randomness extractor to be statistically close to uniform). By running the 3-source extractor on input such triple we will obtain a random string.

Are We Done by Just Using ZAPs? Since our approach consists in extracting random bits from the blockchain, one potential shortcut to obtain a publicly verifiable WI proof could consist of using the extracted bits as the first round of a ZAP, therefore requiring that the prover just computes the second round. This solution however comes with several shortcomings that we want to avoid.

First of all, the second round of the ZAP requires computational assumptions that are not necessarily used by the blockchain. For instance, the ZAP of [18] requires doubly-enhanced trapdoor permutations. In our case, we aim at relying on collision-resistant hash functions only.

Second, a ZAP is not a proof of knowledge and therefore is not useful when knowledge of the witness is what really matters. Indeed we will construct a proof of knowledge.

Third, as we observed above, our guarantee is only that 1 out of $\binom{K}{3}$ retrieved strings is random, but we don't know which one. The ZAP of [18] relies on an extremely long random string sent by the verifier. Obtaining such a huge random string (even assuming that we can extract many huge strings so that at least one of them is random), through extraction of random bits from the blocks of current blockchains is not realistic. Because of the above shortcomings, we devised a more elaborated construction that is in spirit much close to available blockchains, avoiding strong additional assumptions.

Challenge 2: Size of the Transcript and Off-Chain Public Verifiability. Given that we are able to extract random[5] bits from the blockchain, our proof system starting from a Σ-protocol and ending with a publicly verifiable WI proof follows naturally the Fiat-Shamir transform, and it works as follows.

The prover computes τ first rounds $a_1, \ldots, a_\tau$ of the Σ-protocol and publishes them on the blockchain $\mathbf{B}$, where $\tau = \binom{K}{3}$. Then, P waits for K new blocks (where K depends on the chain-consistency and chain-quality properties) added to the blockchain after the first message was posted. Let us denote such blocks as $B_1, \ldots, B_K$. The prover obtains challenges $c_1, \ldots, c_\tau$ by evaluating the

[5] We stress that we obtain a random string that is an unknown position in a vector of $\binom{K}{3}$ strings.

randomness extractor[6] on τ triples of distinct blocks in the set $\{B_1, \ldots, B_K\}$. Finally, P publishes the third rounds $z_1, \ldots, z_\tau$.

The above approach works only when each message a_i fits in the space allowed for a transaction in a block of the blockchain. This might not be necessarily true for any Σ-protocol. Some Σ-protocols might require a first round that is cubic or more in the security parameter and in the length of the statement, and once concretely instantiated, it might easily lead to a first round of few megabytes (and perhaps gigabytes if the instance is really large, also considering potential NP reductions), which can obviously be beyond what is allowed by a blockchain.

To overcome this problem, we propose to upload the hash of the first message, i.e., $H(\mathsf{sid}|a_1||..||a_\tau)$ for an arbitrarily specified handle sid, on the blockchain. Then each c_i is computed as above, and then the third round would consist in revealing the a_i's and the answer z_i's to the verifier only. We call this the extended transcript. This approach allows public verifiability *off-chain*. That is, the entire proof cannot be downloaded from the blockchain, but must be obtained from another source (either the prover itself or another repository), and this is the standard way non-interactive proofs have always been propagated to be verified. We stress that the verifier here would not contribute in the computation of the transcript, she only needs to see the extended version of the transcript. Thus the proof is still reusable many times and is verifiable non-interactively (i.e., it is publicly verifiable). To choose a collision-resistant hash function H, we leverage the blockchain again. We observe that in all existent blockchains, the blocks are chained using a public collision-resistant hash function, that we can use in our proof system.

If instead a transaction of the blockchain can accommodate an a_i and a z_i of the underlying Σ-protocol, then we even get a better property (i.e., on-chain public verifiability) since the NIWI proof will appear completely in the blockchain and therefore there is no need to think about propagating a proof to reach several verifiers.

Additional Properties. Our publicly verifiable WI proof system achieves also additional properties such as:

- Pre-processing: if the underlying Σ-protocol is delayed input, that is, it allows the prover to compute the first round without knowing the theorem to be proven, then a prover can pre-process the first round, and then simply compute z when necessary. To leverage this property the underlying Σ-protocol must be an adaptive-input WI proof of knowledge. This additional requirement however does not significantly restrict the class of suitable Σ-protocols since there are known transformations that add such properties [10,12].
- Proof of knowledge: if the underlying WI proof system is special sound, then we show that our NIWI is also a proof of knowledge. The idea is that in the reduction, our knowledge extraction can simulate the blockchain to the malicious prover and change the relevant subsequence of the honest blocks

[6] More specifically, only some specific parts of the blocks are given as input to the randomness extractor.

(i.e., the cryptographic material bringing high min-entropy) in order to change the output of the randomness extractor and obtain a new challenge.
– Statistical WI PoK: when instantiating our compiler with the LS Σ-protocol [35] and its underlying commitments with a statistically hiding commitment scheme from collision-resistant hash functions, we obtain a statistical WI PoK system. Statistical WI allows to protect the privacy of the secret for ever, even w.r.t. future quantum/unbounded adversaries. The collision-resistant hash function that we use is again the one inferred by the blockchain.

On Achieving Publicly Verifiable Zero Knowledge. A natural next step is to use our publicly verifiable NIWI to construct a publicly verifiable zero-knowledge proof. For example, we could plug our NIWI in the construction provided in [29], that works on any NIWI. This would seemingly produce a NIZK without the strong hardness assumptions (i.e., a NIWI in the standard model required by [29]). We observe, however, that the approach taken in [29] to achieve the zero-knowledge property is affected by some issues that can be apparently tackled only by making additional assumptions on the blockchain protocol that do not seem to be applicable to real-world scenarios. We discuss such issues of the construction of [29] in Sect. 4. In conclusion, achieving publicly verifiable zero knowledge with mild assumptions w.r.t. the most currently used real-world blockchains is an interesting open question.

2 Definitions

Preliminary. We denote the security parameter by λ and use "||" as concatenation operator (i.e., if a and b are two strings then by $a||b$ we denote the concatenation of a and b). For a finite set Q, $x \leftarrow Q$ sampling of x from Q with uniform distribution. We use the abbreviation PPT that stays for probabilistic polynomial time. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function. A *polynomial-time relation* $\mathcal{R}$ (or *polynomial relation*, in short) is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x, w) in $\mathcal{R}$ can be decided in time polynomial in $|x|$. For $(x, w) \in \mathcal{R}$, we call x the *instance* and w a *witness* for x. For a polynomial-time relation $\mathcal{R}$, we define the $\mathcal{NP}$-language $L_{\mathcal{R}}$ as $L_{\mathcal{R}} = \{x | \exists\ w : (x, w) \in \mathcal{R}\}$. Analogously, unless otherwise specified, for an $\mathcal{NP}$-language L we denote by $\mathcal{R}$ the corresponding polynomial-time relation (that is, $\mathcal{R}$ is such that $L = L_{\mathcal{R}}$). We will denote by $\mathcal{P}^{\mathsf{st}}$ a stateful algorithm $\mathcal{P}$ with state st.

Definition 1 (Computational indistinguishability). *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles, where X_λ's and Y_λ's are probability distribution over $\{0,1\}^l$, for same $l = \mathsf{poly}(\lambda)$. We say that $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are* computationally indistinguishable*, denoted $X \approx Y$, if for every* PPT *distinguisher $\mathcal{D}$ there exists a negligible function ν such that for sufficiently large $\lambda \in \mathbb{N}$,*

$$\left|\Pr\left[\, t \leftarrow X_\lambda : \mathcal{D}(1^\lambda, t) = 1 \,\right] - \Pr\left[\, t \leftarrow Y_\lambda : \mathcal{D}(1^\lambda, t) = 1 \,\right]\right| < \nu(\lambda).$$

We note that in the usual case where $|X_\lambda| = \Omega(\lambda)$ and λ can be derived from a sample of X_λ, it is possible to omit the auxiliary input 1^λ. In this paper we also use the definition of *Statistical Indistinguishability*. This definition is the same as Definition 1 with the only difference that the distinguisher $\mathcal{D}$ is unbounded. In this case use $X \equiv_s Y$ to denote that two ensembles are statistically indistinguishable.

The definitions of standard tools can be found in Appendix A.

2.1 Blockchain Protocols

The next two sections follow almost verbatim (with some changes) from [29,46]. A blockchain protocol Γ consists of 4 polynomial-time algorithms ($\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathtt{GetHash}$) with the following syntax.

- $\mathsf{UpdateState}(1^\lambda, \mathsf{st})$: It takes as input the security parameter λ, local state st and outputs the updated state st'.
- $\mathsf{GetRecords}(1^\lambda, \mathsf{st})$: It takes as input the security parameter λ and state st. It outputs the longest ordered sequence of valid blocks $\mathbf{B}$ (or simply blockchain) contained in the state variable, where each block in the chain itself contains an unordered sequence of records messages.
- $\mathsf{Broadcast}(1^\lambda, m)$: It takes as input the security parameter λ and a message m, and broadcasts the message over the network to all nodes executing the blockchain protocol. It does not give any output.
- $\mathsf{GetHash}(1^\lambda, \mathbf{B})$: It takes as input a security parameter 1^λ and a blockchain $\mathbf{B}$, and outputs the description of a collision-resistant hash function $\mathsf{h}(\cdot)$ publicly available in $\mathbf{B}$.

As in [24,46] the blockchain protocol is also parameterized by a validity predicate V that captures semantics of any particular blockchain application. We will indicate with Γ^{V} a blockchain protocol Γ that has validate predicate V.

Remark on the Algorithm $\mathsf{GetHash}$. We are assuming that a blockchain protocol Γ makes use of a collision-resistant hash function $\mathsf{h}(\cdot)$ to maintain the blockchain structure (i.e., to chain the blocks). We explicitly add this algorithm since the same collision-resistant hash function used to chain blocks will then be used in cryptographic protocols that make use of the blockchain. Our assumption is obviously satisfied by the existing blockchains.

Execution of Γ^{V}. At a very high level, the execution of the protocol Γ^{V} proceeds in rounds that model time steps. Each participant in the protocol runs the $\mathsf{UpdateState}$ algorithm to keep track of the current (latest) blockchain state. This corresponds to listening on the broadcast network for messages from other nodes. The $\mathsf{GetRecords}$ algorithm is used to extract an ordered sequence of blocks encoded in the blockchain state variable, which is considered as the common public ledger among all the nodes. The Broadcast algorithm is used by a party when she wants to post a new message m on the blockchain. Note that the message m is accepted by the blockchain protocol only if it satisfies the validity predicate V given the current state, (i.e., the current sequence of blocks).

Following prior works [24,33,46], we define the protocol execution following the activation model of the Universal Composability framework of [8] (though like [29] we will not prove UC-security of our results). For any blockchain protocol $\Gamma^{\mathsf{V}}(\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$, the protocol execution is directed by the environment $\mathcal{Z}(1^\lambda)$ where λ is the security parameter. The environment $\mathcal{Z}$ activates the parties as either honest or corrupt, and is also responsible for providing inputs/records to all parties in each round. All the corrupt parties are controlled by the adversary $\mathcal{A}$ that can corrupt them adaptively after that the execution of Γ^{V} started. The adversary is also responsible for delivery of all network messages. Honest parties start by executing $\mathsf{UpdateState}$ on input 1^λ with an empty local state $\mathsf{st} = \epsilon$.

- In round r, each honest party P_i potentially receives a message(s) m from $\mathcal{Z}$ and potentially receives incoming network messages (delivered by $\mathcal{A}$). It may then perform any computation, broadcast a message (using Broadcast algorithm) to all other parties (which will be delivered by the adversary; see below) and update its local state st_i. It could also attempt to "add" a new block to its chain (e.g., by running the mining procedure).
- $\mathcal{A}$ is responsible for delivering all messages sent by parties (honest or corrupted) to all other parties. $\mathcal{A}$ cannot modify the content of messages broadcast by honest parties, but it may delay or reorder the delivery of a message as long as it eventually delivers all messages within a certain time limit. The identity of the sender is not known to the recipient.
- At any point $\mathcal{Z}$ can communicate with adversary $\mathcal{A}$.

Blockchain Notation. With the notation $\mathbf{B} \leq \mathbf{B}'$ we will denote that the blockchain $\mathbf{B}$ is a prefix of the blockchain $\mathbf{B}'$. We denote by $\mathbf{B}^{\lceil n}$ the chain resulting from "pruning" the last n blocks in $\mathbf{B}$. In the paper we will consider a block in the blockchain as a string $\mathtt{s}$ and a sub-string of $\mathtt{s}$ as a part of a block (a sub-block).

Let P be a party playing in Γ^{V} protocol, the view of P consists of the messages received during the execution of Γ^{V}, along with its randomness and its inputs. Let $\mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ be the random variable denoting the joint view of all parties in the execution of protocol Γ^{V} with adversary $\mathcal{A}$ and set of honest parties $\mathcal{H}$ in environment $\mathcal{Z}$. This joint view view fully determines the execution. Let $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ denote an execution of $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ producing view as joint view.

Some Constraints on the Adversary. In order show that a blockchain enjoys some useful properties like chain quality, prior works [24,46] restrict their analysis to compliant executions of Γ^{V}. Such blockchain implementation assume some restrictions on the power of the adversary. For instance, they require that any broadcasted message is delivered in a maximum number of time steps, as we have specified earlier, or could require secure erasure for honest parties. Those works showed that certain desirable security properties are respected except with negligible probability in any compliant execution. Obviously when in our work

we claim results assuming some properties of the blockchain, we are taking into account compliant executions of the underlying blockchain protocol only. The same is done by [29].

Properties of a Γ^{V} Protocol. The following section is taken verbatim from [29], and the following properties where defined in previous works [24,46].

Chain consistency predicate. Let $\mathsf{Consistent}$ be the predicate such that $\mathsf{Consistent}^{\eta}(\mathsf{view}) = 1$ iff for all rounds $r \leq \tilde{r}$ and all parties $\mathsf{P}_i, \mathsf{P}_j$ (potentially the same) in view such that P_i is honest at round r with blockchain $\mathbf{B}$ and P_j is honest at round $\tilde{r}$ with blockchain $\tilde{\mathbf{B}}$, we have that $\mathbf{B}^{\lceil \eta} \leq \tilde{\mathbf{B}}$.

Definition 2 *(Chain Consistency). A blockchain protocol Γ^{V} satisfies $n_0(\cdot)$-consistency with adversary $\mathcal{A}$, honest parties $\mathcal{H}$, and environment $\mathcal{Z}$, if there exists a negligible function $\nu(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\eta > n_0(\cdot)$ the following holds:*

$$\Pr\left[\, \mathsf{Consistent}^{\eta}(\mathsf{view}) = 1 \middle| \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^{\lambda}) \,\right] \geq 1 - \nu(\lambda).$$

Chain quality predicate. Let $\mathtt{Quality}$ be the predicate such that $\mathtt{Quality}^{\eta}_{\mathcal{A}}(\mathsf{view}, \mu) = 1$ iff for all rounds $r \geq \eta$ and all parties P_i in view such that P_i is honest at round r with blockchain $\mathbf{B}$, we have that out of η last blocks in $\mathbf{B}$ at least μ fraction of blocks are "honest".

Note that a block is said to be honest iff it is mined by an honest party.

Definition 3 *(Chain Quality). A blockchain protocol Γ^{V} satisfies $(\mu(\cdot), n_0(\cdot))$-chain quality with adversary $\mathcal{A}$, honest parties $\mathcal{H}$, and environment $\mathcal{Z}$, if there exists a negligible function $\nu(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\eta > \eta_0(\lambda)$ the following holds:*

$$\Pr\left[\, \mathtt{Quality}^{\eta}_{\mathcal{A}}(\mathsf{view}, \mu(\lambda)) = 1 \middle| \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^{\lambda}) \,\right] \geq 1 - \nu(\lambda).$$

In the rest of the paper we will indicate with $(\mu(\cdot), n_0(\cdot))$ the chain quality parameters of Γ^{V}.

2.2 Definitions of Publicly Verifiable WI Arguments of Knowledge

Here we define publicly verifiable proofs over a blockchain. The main insight of our definition is that the verification is non-interactive, and the verifier does not need to be a party involved in the blockchain. The prover instead needs to actively interact with the blockchain.

Definition 4. *A pair of stateful* PPT *algorithms $\Pi = (\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^{V} is a publicly verifiable argument system for the $\mathcal{NP}$-language $\mathcal{L}$ with witness relation $\mathcal{R}$ if it satisfies the following properties:*

Completeness. $\forall\ x, w$ *s.t.* $\mathcal{R}(x, w) = 1$, $\forall$ PPT *adversary* $\mathcal{A}$ *and set of honest parties* $\mathcal{H}$ *and environment* $\mathcal{Z}$*, assuming that* $\mathcal{P} \in \mathcal{H}$*, there exist negligible functions* $\nu_1(\cdot)$, $\nu_2(\cdot)$ *such that:*

$$\Pr\left[\mathcal{V}(x, \pi, \mathbf{B}) = 1 \;\vdots\; \begin{array}{r} \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda) \\ \pi \leftarrow \mathcal{P}^{\mathsf{st}_{\mathcal{P}}}(x, w) \\ \mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j) \end{array}\right] \geq 1 - \nu_1(|x|) - \nu_2(\lambda)$$

where $\mathsf{st}_{\mathcal{P}}$ *denotes the state of* $\mathcal{P}$ *during the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$*. The running time of* $\mathcal{P}$ *is polynomial in the size of the blockchain* $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j)$ *where* st_j *is the state of* $\mathsf{P}_j \in \mathcal{H}$ *at the end of the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$[7]*. Furthermore* st_j *is the state of an honest party* $\mathsf{P}_j \in \mathcal{H}$ *at the end of the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$*.*
If all message of π *are in the blockchain then the proof is* on-chain*, and is* off-chain *otherwise.*

Soundness. $\forall\ x \notin \mathcal{L}$, $\forall$ *stateful* PPT *adversary* $\mathcal{A}$ *and set of honest parties* $\mathcal{H}$ *and environment* $\mathcal{Z}$*, there exist negligible functions* $\nu_1(\cdot)$, $\nu_2(\cdot)$ *such that:*

$$\Pr\left[\mathcal{V}(x, \pi, \mathbf{B}) = 1 \;\vdots\; \begin{array}{r} \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda) \\ \pi, x \leftarrow \mathcal{A}^{\mathsf{st}_{\mathcal{A}}} \\ \mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j) \end{array}\right] \leq \nu_1(|x|) + \nu_2(\lambda)$$

where $\mathsf{st}_{\mathcal{A}}$ *denotes the state of* $\mathcal{A}$ *during the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$*. Furthermore* st_j *is the state of an honest party* $\mathsf{P}_j \in \mathcal{H}$ *at the end of the execution* ${\Gamma^{\mathsf{V}}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$*.*

Definition 5. *A public verifiable argument system $\Pi = (\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^{V} for the $\mathcal{NP}$-language $\mathcal{L}$ with witness relation $\mathcal{R}$ is an argument of Knowledge (AoK) if it satisfies the following property.*

Argument of Knowledge (AoK). There is a stateful PPT *algorithm $\mathcal{E}$ such that for all x, any stateful* PPT *adversary $\mathcal{A}$ and any set of honest parties $\mathcal{H}$ and environment $\mathcal{Z}$, there exist negligible functions $\nu_1(\cdot)$, $\nu_2(\cdot)$ such that:*

$$\left\{(\mathsf{view}_{\mathcal{A}}) \;:\; \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)\right\} \approx \left\{(\mathsf{view}_{\mathcal{A}}) \;:\; \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda)\right\}$$

and

$$\Pr\left[\mathcal{V}(x, \pi, \mathbf{B}) = 0 \vee\ \mathcal{R}(x, w) = 1 \;\vdots\; \begin{array}{r} \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda) \\ \mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j) \\ w \leftarrow \mathcal{E}(\pi, x), (\pi, x) \leftarrow \mathcal{A}^{\mathsf{st}_{\mathcal{A}}} \end{array}\right] \geq 1 - \nu_1(|x|) - \nu_2(\lambda)$$

[7] Note that after that $\mathcal{P}$ outputs π, the execution of $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ could still continue even though $\mathsf{st}_{\mathcal{P}}$ will not change anymore.

where $\mathsf{st}_{\mathcal{A}}$ *denotes the state of* $\mathcal{A}$ *during the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda)$ *and* $\mathsf{view}_{\mathcal{A}}$ *is the view of* $\mathcal{A}$ *in* view. *Furthermore* st_j *is the state of an honest party* $\mathsf{P}_j \in \mathcal{H}$ *at the end of the execution* $\Gamma^{\mathsf{V}}{}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$.

Definition 6. *A publicly verifiable argument system* $\Pi = (\mathcal{P}, \mathcal{V})$ *over a blockchain protocol* Γ^{V} *for the* $\mathcal{NP}$*-language* $\mathcal{L}$ *with witness relation* $\mathcal{R}$ *is witness indistinguishable (WI) if it satisfies the following property:*

$\forall\ x, w_0, w_1$ *s.t.* $\mathcal{R}(x, w_0) = 1$ *and* $\mathcal{R}(x, w_1) = 1$, $\forall$ *stateful* PPT *adversary* $\mathcal{A}$ *and set of honest parties* $\mathcal{H}$ *and environment* $\mathcal{Z}$, *assuming that* $\mathcal{P} \in \mathcal{H}$ *it holds that:*

$$\left\{(\mathsf{view}_{\mathcal{A}}, \pi) : \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda), \pi \leftarrow \mathcal{P}^{\mathsf{st}_{\mathcal{P}}}(x, w_0)\right\}$$

$$\approx$$

$$\left\{(\mathsf{view}_{\mathcal{A}}, \pi) : \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda), \pi \leftarrow \mathcal{P}^{\mathsf{st}_{\mathcal{P}}}(x, w_1)\right\}$$

where $\mathsf{st}_{\mathcal{P}}$ *denotes the state of* $\mathcal{P}$ *during the execution* $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ *and* $\mathsf{view}_{\mathcal{A}}$[8] *is the view of* $\mathcal{A}$ *in* view.

Definition 7 (Min-Entropy). *Let* X *be a random variable with finite support* $\mathcal{X}$. *The min-entropy* $H_\infty(X)$ *of* X *is defined by*

$$H_\infty(X) = \min_{x \in \mathcal{X}} \log_2(1/\Pr[X = x]).$$

For $X \in \{0,1\}^n$, we call X a (n, λ)-source, where λ is the min-entropy of X (i.e., $\lambda = H_\infty(X)$).

Definition 8 (Honest Block Generation Algorithm). *An honest block-generation algorithm is a randomized* PPT *algorithm* $\mathsf{HB} : \{0,1\}^* \to \{0,1\}^n$, *where* $n = \mathsf{poly}(\lambda)$, *such that there exists a deterministic function* s *such that for all* $x \in \{0,1\}^*$, $v = s(\mathsf{HB}(x))$, $|v| = n$ *it holds that*

$$H_\infty(s(\mathsf{HB}(x))) \geq \lambda$$

and therefore $s(\mathsf{HB}(\cdot))$ *is a* (n, λ)*-source.*

Assumption 1. *Let* Γ^{V} *be a blockchain protocol. There exists* $t = \mathsf{poly}(\lambda)$ *such that the probability that a sequence of* t *consecutive blocks in a blockchain* $\mathbf{B}$ *generated via* Γ^{V} *does not include at least* 3 *blocks computed by a* HB *is negligible in* λ.

[8] Note that $\mathsf{view}_{\mathcal{A}}$ can contain auxiliary inputs from the execution of $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ that could continue after that π is computed.

3 Publicly Verifiable WI AoK

In order to construct an off-chain publicly verifiable non-interactive witness indistinguishable argument of knowledge $\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ over a blockchain protocol $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$ for the $\mathcal{NP}$-language $\mathcal{L}$[9] we make use of the following tools:

- A 3-round delayed-input public-coin adaptive-input WI adaptive-input special sound proof system $\Pi_{\Sigma} = (\mathcal{P}_{\Sigma}, \mathcal{V}_{\Sigma})$ for the $\mathcal{NP}$-language $\mathcal{L}$ with instance length ℓ;
- An efficient procedure $\mathtt{ExtProc}$ that on input a 3-source randomness extractor $\mathsf{E}_{n,\lambda}$ and a sequence of $t = 3 \cdot q$[10] blocks $B_1, \ldots, B_t$ computes the following steps:
  1. Construct a set of sub-blocks (through the function s) adding a sub-block for each block in the sequence $\{B_1, \ldots, B_t\}$.
  2. Evaluate $\mathsf{E}_{n,\lambda}$ on all the possible subsets of 3 elements of the set of sub-blocks.
  3. Output all the $\binom{t}{3}$ evaluations of $\mathsf{E}_{n,\lambda}$.

$\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ works as follows.

$\mathcal{P}_{\mathsf{wi}}$ on input parameters (ℓ, s, t), an instance x and a witness w s.t. $\mathcal{R}(x, w) = 1$ computes the following steps, where x and w are used in the 7th step.

1. Set $st_{\mathcal{P}} = \epsilon$ and run $\mathsf{st}_{\mathcal{P}} = \mathsf{UpdateState}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$.
2. Run $\mathbf{B} = \mathsf{GetRecords}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$, $\mathsf{h}(\cdot) = \mathsf{GetHash}(1^{\lambda}, \mathbf{B})$.
3. Let $\tau = \binom{t}{3}$. For $i = 1, \ldots, \tau$: compute $\Sigma_i^1 \leftarrow \mathcal{P}_{\Sigma}(1^{\lambda}, \ell)$.
4. Compute $\alpha \leftarrow \mathsf{h}(\Sigma_1^1, || \ldots || \Sigma_{\tau}^1)$[11] and post α on the blockchain by running $\mathsf{Broadcast}(1^{\lambda}, \alpha)$.
5. Run $\mathsf{st}_{\mathcal{P}} = \mathsf{UpdateState}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$, $\mathbf{B} = \mathsf{GetRecords}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$ and wait until α is posted on the blockchain and further the chain is extended by t blocks.
6. Let B^* be the block of the blockchain $\mathbf{B}$ where the message α is posted and let $B_1, ..., B_t$ be the t consecutive blocks of the blockchain $\mathbf{B}$ after B^*. Run $\{\Sigma_i^2\}_{i=1}^{\tau} = \mathtt{ExtProc}(\mathsf{E}_{n,\lambda}, B_1, ..., B_t)$.
7. For $i = 1, \ldots, \tau$ compute $\Sigma_i^3 \leftarrow \mathcal{P}_{\Sigma}(\Sigma_i^2, x, w)$.
8. Run $\mathsf{st}_{\mathcal{P}} = \mathsf{UpdateState}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$ and $\mathbf{B}' = \mathsf{GetRecords}(1^{\lambda}, \mathsf{st}_{\mathcal{P}})$.
9. Set $\pi = (x, \alpha, \{\Sigma_i^1, \Sigma_i^2, \Sigma_i^3\}_{i=1}^{\tau}, \mathbf{B}')$ and output π.

$\mathcal{V}_{\mathsf{wi}}$ on input the statement x, $\pi = (\alpha, \{\Sigma_i^1, \Sigma_i^2, \Sigma_i^3\}_{i=1}^{\tau}, \mathbf{B}')$, and a blockchain $\tilde{\mathbf{B}}$ works as follows. If the message α is not posted on the blockchain $\mathbf{B}'$ then $\mathcal{V}_{\mathsf{wi}}$ outputs 0 otherwise she continues with the following steps.

[9] We remark that our results require that Assumption 1 is not violated.

[10] q is s.t. $q \geq n_0(\lambda)$ where $(\mu(\cdot), n_0(\cdot))$ are the chain quality parameters of Γ^{V}.

[11] The hash value of the string $\Sigma_1^1, || \ldots || \Sigma_{\tau}^1$ is computed through a Merkle Tree [43], therefore α corresponds to the root of a Merkle Tree.

Let B^* be the block of the blockchain $\mathbf{B}'$ where the message α is posted. Let $B_1, \ldots, B_t$ be t blocks of the blockchain $\mathbf{B}'$ after B^*. $\mathcal{V}_{\mathsf{wi}}$ computes $\mathsf{h}(\cdot) = \mathsf{GetHash}(1^\lambda, \mathbf{B}')$, $\{\Sigma_i^2\}_{i=1}^\tau = \mathtt{ExtProc}(\mathsf{E}_{n,\lambda}, B_1, \ldots, B_t)$ and outputs 1 if the following conditions are satisfied:

1. $\mathbf{B}' \leq \tilde{\mathbf{B}}$;
2. $\alpha = \mathsf{h}(\Sigma_1^1, || \ldots || \Sigma_\tau^1)$;
3. $\mathcal{V}_\Sigma(x, \Sigma_i^1, \Sigma_i^2, \Sigma_i^3) = 1$ for $i = 1, \ldots, \tau$.

Theorem 1. *Under Assumption 1 and assuming the existence of one-to-one one-way functions*[12]*, $\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ is a publicly verifiable off-chain adaptive-input witness-indistinguishable argument of knowledge over a blockchain protocol $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$ for $\mathcal{NP}$.*

Completeness. Completeness follows by the chain consistency of Γ^{V}, the completeness of Π_Σ and the definitions of $\mathsf{h}, \mathsf{E}_{n,\lambda}$. We note that a candidate to instantiate $\Pi_\Sigma = (\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is the construction of [20] that is delayed-input, and adaptive-input secure in the variant of [10].

A Note on the Delayed-Input Property of Π_{wi}. Fixing any x, w, s.t. $\mathcal{R}(x, w) = 1$, we note that $\mathcal{P}_{\mathsf{wi}}$ is using x, w just to compute the 7th step of Π_{wi}. Therefore, Π_{wi} can compute the first 6 steps of Π_{wi} as a preprocessing phase, without knowing x or w (just the size is required). Then when (in any point in the future) x, w will be available $\mathcal{P}_{\mathsf{wi}}$ computes the last 3 steps of Π_{wi}.

We also want to point out that Theorem 1 holds even when there is no delayed-input property, therefore for any WI Σ-protocol, however in this case x, w are needed by $\mathcal{P}_{\mathsf{wi}}$ already when she computes the 1st step of Π_{wi}.

Adaptive-Input Witness Indistinguishability. In order to show that Π_{wi} enjoys the witness indistinguishability property we will consider the following 2 hybrid experiments.

Let $H_0(\lambda)$ be defined as the execution of Π_{wi}, where $\mathcal{P}_{\mathsf{wi}}$ uses the witness w_0. Let $H_1(\lambda)$ be defined as the execution of Π_{wi}, where $\mathcal{P}_{\mathsf{wi}}$ uses the witness w_1. Let $\mathcal{A}$ be the adversary as defined in Definition 6. The output of each experiment is the pair $(\pi, \mathsf{view}_{\mathcal{A}})$, where π is the transcript of Π_{wi} computed in the experiment and $\mathsf{view}_{\mathcal{A}}$ is the view of $\mathcal{A}$ in the experiment.

Claim 1. For every x, w_0, w_1 s.t. $\mathcal{R}(x, w_0) = 1$ and $\mathcal{R}(x, w_1) = 1$ chosen adaptively by $\mathcal{A}$ it holds that $H_0(\lambda) \approx H_1(\lambda)$.

Proof. Suppose by contradiction the above claim does not hold, then it is possible to construct a malicious verifier $\mathcal{V}_\Sigma^*$ that breaks the adaptive-input WI property of Π_Σ. Let $\mathcal{CH}$ be the challenger of adaptive WI game of Π_Σ. $\mathcal{V}_\Sigma^*$ will interact as a proxy between $\mathcal{CH}$ and $\mathcal{A}$ for the messages $\{\Sigma_i^1, \Sigma_i^3\}_{i=1}^\tau$ and she will compute

[12] The need of one-to-one one-way functions will be removed by Corollary 1. Theorem 1 also needs the existence of CRHFs, but as specified earlier we are assuming that a blockchain protocol along with a genesis block already specifies a CRHF.

all other messages following $\mathcal{P}_{\mathsf{wi}}$ of H_0 (of H_1). In the end of the interaction $\mathcal{V}^*_\Sigma$ will output the output of $\mathcal{A}$.

In more details, $\mathcal{V}^*_\Sigma$ receives $\{\tilde{\Sigma}^1_i\}^\tau_{i=1}$ from $\mathcal{CH}$ and sets $\Sigma^1_i = \tilde{\Sigma}^1_i$ for $i = \{1, \ldots, \tau\}$, then to compute the other steps of Π_{wi}, until Step 7, she acts as $\mathcal{P}_{\mathsf{wi}}$ of H_0 (of H_1). In particular in Step 6 $\mathcal{V}^*_\Sigma$ computes $\{\Sigma^2_i\}^\tau_{i=1}$ as $\mathcal{P}_{\mathsf{wi}}$ of H_0 (of H_1) does. $\mathcal{V}^*_\Sigma$ sends $\{\Sigma^2_i\}^\tau_{i=1}$ to $\mathcal{CH}$ along with x, w_0, w_1 obtained from $\mathcal{A}$. $\mathcal{V}^*_\Sigma$ receives $\{\tilde{\Sigma}^3_i\}^\tau_{i=1}$ from $\mathcal{CH}$ and sets $\Sigma^3_i = \tilde{\Sigma}^3_i$ for $i = \{1, \ldots, \tau\}$, $\mathcal{V}^*_\Sigma$ completes the computations of π precisely as $\mathcal{P}_{\mathsf{wi}}$ does in both H_0 and H_1. At the end of the execution $\mathcal{V}^*_\Sigma$ outputs what $\mathcal{A}$ outputs. The proof is concluded observing that if $\mathcal{CH}$ uses the witness w_0 to compute $\{\tilde{\Sigma}^3_i\}^\tau_{i=1}$ then the reduction is distributed as H_0. Instead if $\mathcal{CH}$ uses the witness w_1 to compute $\{\tilde{\Sigma}^3_i\}^\tau_{i=1}$ then the reduction is distributed as H_1.

From Claim 1 we can conclude that $H_0(\lambda) \approx H_1(\lambda)$.

High-Level Overview of the Proof of Adaptive-Input Soundness. Assume by contradiction that there exists $\mathcal{P}^\star$ that produces with probability non-negligible p an accepting π of Π_{wi} w.r.t. $x \notin L$, where x is adaptively chosen by $\mathcal{P}^\star$.

The proof will proceeds in 3 steps:

(1) We will describe an efficient procedure `Proc` that internally executes $\Gamma^{\mathsf{V}}(\mathcal{P}^\star, \mathcal{H}, \mathcal{Z}, 1^\lambda)$. `Proc` will use a specific rewinding strategy.
(2) We will prove that with non-negligible probability `Proc` outputs $(x, \Sigma^1_y, \Sigma^2_y, \Sigma^3_y), (\tilde{x}, \tilde{\Sigma}^1_y, \tilde{\Sigma}^2_y, \tilde{\Sigma}^3_y)$ s.t. $\Sigma^1_y = \tilde{\Sigma}^1_y$ and $\Sigma^2_y \neq \tilde{\Sigma}^2_y$.
(3) We will use the output of `Proc` to reach a contradiction. In more details, we note that by the adaptive-input soundness of Π_Σ there exists an algorithm that on input $(x, \Sigma^1_y, \Sigma^2_y, \Sigma^3_y)$ and $(\tilde{x}, \Sigma^1_y, \tilde{\Sigma}^2_y, \tilde{\Sigma}^3_y)$ in polynomial time outputs $w, \tilde{w}$ s.t. $\mathcal{R}(x, w) = 1$ and $\mathcal{R}(\tilde{x}, \tilde{w}) = 1$. Therefore we reach a contradiction since we were assuming that $x \notin \mathcal{L}$.

Adaptive-Input Soundness. We will now proceed more formally. Assume by contradiction that there exists $\mathcal{P}^\star$ that produces with probability non-negligible p an accepting π of Π_{wi} w.r.t. $x \notin L$, where x is adaptively chosen by $\mathcal{P}^\star$.

Let us fix $y \in \{1, ..., \tau\}$ and consider the following experiment `Proc`(y).

`Proc`(y):

1. Sample at random a long enough string ω and execute $\Gamma^{\mathsf{V}}(\mathcal{P}^\star, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ emulating all the honest parties $\mathcal{H}$ using different substrings of ω as randomnesses.
2. If $\mathcal{P}^\star$ sends x and an accepting $\pi = (\alpha, \{\Sigma^1_i, \Sigma^2_i, \Sigma^3_i\}^\tau_{i=1}, \mathbf{B})$ w.r.t x compute step 3 and `abort` otherwise.
3. Let $\tilde{\mathbf{B}}$ be the blockchain that is defined by the state of some honest party after that the message α is posted by $\mathcal{P}^\star$, and let B^* be the block in $\tilde{\mathbf{B}}$ where this message is posted.s that participate in the execution of Γ only after the block B^*. Rewind the execution of $\Gamma^{\mathsf{V}}(\mathcal{P}^\star, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ just after the block B^* is created.
4. Sample at random a long enough string ω' and continue the execution of $\Gamma^{\mathsf{V}}(\mathcal{P}^\star, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ emulating all the honest parties $\mathcal{H}$ using different substrings of ω' as randomnesses.

5. If $\mathcal{P}^\star$ sends $\tilde{x}$ and an accepting $\tilde{\pi} = (\tilde{\alpha}, \{\tilde{\Sigma}_i^1, \tilde{\Sigma}_i^2, \tilde{\Sigma}_i^3\}_{i=1}^{\tau}, \mathbf{B})$ w.r.t. $\tilde{x}$ compute the following steps and `abort` otherwise.
   5.1. If $(\tilde{\Sigma}_y^1 \neq \Sigma_y^1)$ stop and output $(\tilde{\Sigma}_1^1, ||\ldots||\tilde{\Sigma}_y^1||\ldots||\tilde{\Sigma}_\tau^1, \Sigma_1^1, ||\ldots|| \Sigma_y^1||\ldots||\Sigma_\tau^1)$.
   5.2. If $(\tilde{\Sigma}_y^2 \neq \Sigma_y^2)$ stop and output $(x, \Sigma_y^1, \Sigma_y^2, \Sigma_y^3)$, $(\tilde{x}, \Sigma_y^1, \tilde{\Sigma}_y^2, \tilde{\Sigma}_y^3)$.
   5.3. If $(\tilde{\Sigma}_y^2 = \Sigma_y^2)$ stop and output 0.

Claim 2. The probability that `Proc` obtains from $\mathcal{P}^\star$ two accepting transcripts π, $\tilde{\pi}$ of Π_{wi} respectively in Step 2 and in Step 5 is at least p^2.

We note that the views defined by $\mathsf{Exec}^{\Gamma^\vee}(\mathcal{P}^\star, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ before and after a rewind are statistically close because the procedure `Proc` acts in the same way after and before a rewind, using just a different randomness for emulating the honest parties. Therefore the view of $\mathcal{P}^\star$ before a rewind is statistically close to the view of $\mathcal{P}^\star$ after a rewind. Since, before step 3 we are assuming (by contradiction) that $\mathcal{P}^\star$ will compute an accepting π of Π_{wi} w.r.t. $x \notin \mathcal{L}$ with non-negligible probability p, after a rewind $\mathcal{P}^\star$ will do the same with the same probability. From the above arguments we can conclude that the claim holds.

Claim 3. For every $y \in \{1, ..., \tau\}$ if $\mathtt{Proc}(y)$ receives an accepting $\tilde{\pi}$ in Step 5, then $\mathtt{Proc}(y)$ outputs 0 with negligible probability.

Let $B_1, ..., B_t$ be the blocks used by $\mathcal{P}^\star$ to run $\{\Sigma_i^2\}_{i=1}^{\tau} = \mathtt{ExecProc}(\mathsf{E}_{n,\lambda}, B_1, ..., B_t)$. From Assumption 1 it follows that at least 3 independent sub-blocks have enough[13] min-entropy. Therefore at least one 2nd round of Π_Σ obtained by `ExecProc` is distributed statistically close to the uniform distribution over $\{0,1\}^\lambda$. Let us call this 2nd round of Π_Σ the *good challenge*. It also follows from Assumption 1 that this min-entropy comes from the randomnesses used to run HB by honest parties. Let us call the independent sub-blocks with enough min-entropy the *good sub-blocks*. Note that the procedure `Proc` after a rewind changes the randomness used by the honest parties and thus from the definition of $\mathsf{E}_{n,\lambda}$ the *good challenge* produced by `ExecProc` will change before and after the rewind with overwhelming probability.

Claim 4. For every $y \in \{1, ..., \tau\}$ if $\mathtt{Proc}(y)$ receives an accepting $\tilde{\pi}$ in Step 5, then $\mathtt{Proc}(y)$ outputs $(\tilde{\Sigma}_1^1, ||\ldots||\tilde{\Sigma}_y^1||\ldots||\tilde{\Sigma}_\tau^1, \Sigma_1^1, ||\ldots||\Sigma_y^1||\ldots||\Sigma_\tau^1)$ s.t. $(\tilde{\Sigma}_y^1 \neq \Sigma_y^1)$ with negligible probability.

Suppose that the claim does not hold. We will show a PPT $\mathcal{A}_{\mathsf{h}}$ that breaks the collision resistance of $\mathsf{h}(\cdot)$. $\mathcal{A}_{\mathsf{h}}$ chooses y at random from $\{1, \ldots, \tau\}$ and follows the steps of $\mathtt{Proc}(y)$. $\mathcal{A}_{\mathsf{h}}$ outputs what $\mathtt{Proc}(y)$ outputs.

From Claim 3 it follows that if $\mathtt{Proc}(y)$ receives an accepting $\tilde{\pi}$ in Step 5 it outputs 0 with negligible probability, therefore $\mathcal{P}^\star$ produces two accepting proofs π and $\tilde{\pi}$ that contain two accepting transcripts of Π_Σ, namely $(x, \Sigma_y^1, \Sigma_y^2, \Sigma_y^3)$

[13] From Assumption 1, it follows that there are at least λ bits of min-entropy in each of the 3 sub-blocks.

and $(\tilde{x}, \Sigma_y^1, \tilde{\Sigma}_y^2, \tilde{\Sigma}_y^3)$. These two accepting transcripts of Π_Σ (by contradiction) differ also in the first round, but $\tilde{\Sigma}_y^1$, Σ_y^1 are s.t. $\mathsf{h}(\tilde{\Sigma}_1^1, || \ldots || \tilde{\Sigma}_y^1 || \ldots || \tilde{\Sigma}_\tau^1) = \mathsf{h}(\Sigma_1^1, || \ldots || \Sigma_y^1 || \ldots || \Sigma_\tau^1)$ since both π and $\tilde{\pi}$ are accepting. We can conclude that $\mathcal{A}_\mathsf{h}$ succeeds to find a collision for $\mathsf{h}(\cdot)$ with non-negligible probability.

From Claims 2, 3, 4 it follows that `Proc` outputs $(x, \Sigma_y^1, \Sigma_y^2, \Sigma_y^3)$, $(\tilde{x}, \Sigma_y^1, \tilde{\Sigma}_y^2, \tilde{\Sigma}_y^3)$ with probability at least p^2 therefore, due to the adaptive-input soundness of Π_Σ, using $(x, \Sigma_y^1, \Sigma_y^2, \Sigma_y^3)$ and $(\tilde{x}, \Sigma_y^1, \tilde{\Sigma}_y^2, \tilde{\Sigma}_y^3)$ it is possible to compute and extract $w, \tilde{w}$ s.t. $\mathcal{R}(x, w) = 1$ and $\mathcal{R}(\tilde{x}, \tilde{w}) = 1$. Therefore we reach a contradiction since we were assuming that $x \notin L$.

AoK. Let $\mathcal{P}^\star$ be an adversary that with non-negligible probability produces an accepting π of Π_wi w.r.t. x, where x is adaptively chosen by $\mathcal{P}^\star$. Then, we show an extractor $\mathcal{E}$ that with oracle access to $\mathcal{P}^\star$ in expected polynomial time outputs w s.t. $\mathcal{R}(x, w) = 1$.

$\mathcal{E}$ works as follows. $\mathcal{E}$ runs the firs 2 steps of `Proc` and if it obtains a first accepting transcript of Π_wi w.r.t. x, then it rewinds $\mathcal{P}^\star$ until it obtains a second accepting transcript of Π_wi, or a specific bound on the number of attempts is reached. $\mathcal{E}$ applies the same rewinding procedure described in steps 3 and 4 of `Proc`.

Let us denote as colliding transcripts, two transcripts $(\Sigma^1, \Sigma^2, \Sigma^3)$ and $(\tilde{\Sigma}^1, \tilde{\Sigma}^2, \tilde{\Sigma}^3)$ of Π_Σ w.r.t. x and $\tilde{x}$, s.t. $\Sigma^1 = \tilde{\Sigma}_1$ and $\Sigma^2 \neq \tilde{\Sigma}_2$. We make the following observations:

Obs. (1) If in one of the rewinds $\mathcal{P}^\star$ gives a second accepting transcripts of Π_wi then from Claims 2, 3, 4 it follows that $\mathcal{E}$ obtains two colliding transcripts of Π_Σ.

Obs. (2) If $\mathcal{E}$ is able to obtain from $\mathcal{P}^\star$ two colliding transcripts of Π_Σ for statements $x, \tilde{x}$ then $\mathcal{E}$ runs the extractor of Π_Σ and obtains in polynomial time $w, \tilde{w}$ s.t. $\mathcal{R}(x, w) = 1$ and $\mathcal{R}(\tilde{x}, \tilde{w}) = 1$.

Obs. (3) For the same arguments exposed in Claim 2 in each rewind the view of $\mathcal{P}^\star$ before a rewind is statistically close to the view of $\mathcal{P}^\star$ after a rewind.

Therefore from standard arguments it follows that in expected polynomial time $\mathcal{E}$ outputs w s.t. $\mathcal{R}(x, w) = 1$ with overwhelming probability.

We note that a candidate to instantiate $\Pi_\Sigma = (\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is the construction of LS [20] that is delayed-input, and adaptive-input in the variant of [10]. Furthermore if the underling commitment scheme of LS is instantiate from CRHFs, then LS enjoys the statistical WI property. Since we can obtain the description of a CRHF from $\mathsf{GetHash}$, it follows that it is possible to instantiate Π_wi over a blockchain protocol Γ^V without requiring additional computational assumptions.

Corollary 1. *If Assumption 1 holds, then $\Pi_\mathsf{wi} = (\mathcal{P}_\mathsf{wi}, \mathcal{V}_\mathsf{wi})$ is a publicly verifiable off-chain statistical adaptive-input witness indistinguishable AoK over a blockchain protocol $\Gamma^\mathsf{V} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$ for $\mathcal{NP}$*[14].

[14] Again, we are implicitly assuming that a CRHF comes for free from a blockchain.

3.1 An On-Chain Publicly Verifiable WI AoK

In order to construct an on-chain publicly verifiable non-interactive witness indistinguishable argument of knowledge $\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ over blockchain protocol $\Gamma_n^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$ for the $\mathcal{NP}$-language $\mathcal{L}$ we make use of the following tools:

- A 3-round communication-efficient[15] delayed-input public-coin adaptive-input WI adaptive-input special sound proof system $\Pi_\Sigma = (\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ for $\mathcal{L}$ with instance length ℓ;
- An efficient procedure $\mathtt{ExtProc}$ that on input a 3-source randomness extractor $\mathsf{E}_{n,\lambda}$ and a sequence of $t = 3 \cdot q$[16] blocks $B_1, \ldots, B_t$ computes the following steps:
  1. Construct a set of sub-blocks (through the function s) adding a sub-block for each block in the sequence $\{B_1, \ldots, B_t\}$.
  2. Evaluate $\mathsf{E}_{n,\lambda}$ on all the possible subsets of 3 elements of set of sub-blocks.
  3. Output all the $\binom{t}{3}$ evaluations of $\mathsf{E}_{n,\lambda}$.

$\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ works as follow.

$\mathcal{P}_{\mathsf{wi}}$ on input the parameter ℓ, an instance x and a witness w s.t. $\mathcal{R}(x, w) = 1$ computes the following steps, where x, w are used in the 5th step.

1. Set $st_{\mathcal{P}} = \epsilon$ and run $\mathsf{st}_{\mathcal{P}} = \mathsf{UpdateState}(1^\lambda, \mathsf{st}_{\mathcal{P}})$.
2. Set $\tau = \binom{t}{3}$. For $i = 1, \ldots, \tau$ compute $\Sigma_i^1 \leftarrow \mathcal{P}_\Sigma(1^\lambda, \ell)$ and post Σ_i^1 on the blockchain by executing $\mathsf{Broadcast}(1^\lambda, \Sigma_i^1)$.
3. Run $\mathsf{st}_{\mathcal{P}} = \mathsf{UpdateState}(1^\lambda, \mathsf{st}_{\mathcal{P}})$ and $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_{\mathcal{P}})$ wait until the messages $\{\Sigma_i^1\}_{i=1}^\tau$ are posted on the blockchain and further the chain is extended by t blocks.
4. Let B^* be the last block of the blockchain $\mathbf{B}$ where the messages $\{\Sigma_i^1\}_{i=1}^\tau$ are posted and let $B_1, ..., B_t$ be t blocks of the blockchain $\mathbf{B}$ after B^*. Run $\{\Sigma_i^2\}_{i=1}^\tau = \mathtt{ExtProc}(\mathsf{E}_{n,\lambda}, B_1, \ldots, B_t)$.
5. For $i = 1, \ldots, \tau$ compute $\Sigma_i^3 \leftarrow \mathcal{P}_\Sigma(\Sigma_i^2, x, w)$ and post Σ_i^3 on the blockchain executing $\mathsf{Broadcast}(1^\lambda, \Sigma_i^3)$.
6. Run $\mathsf{st}'_{\mathcal{P}} = \mathsf{UpdateState}(1^\lambda, \mathsf{st}_{\mathcal{P}})$, $\mathbf{B}' = \mathsf{GetRecords}(1^\lambda, \mathsf{st}'_{\mathcal{P}})$ and wait for messages $\{\Sigma_i^3\}_{i=1}^\tau$ to be posted on the blockchain.
7. Set $\pi = (\{\Sigma_i^1, \Sigma_i^2, \Sigma_i^3\}_{i=1}^\tau, \mathbf{B}')$.

$\mathcal{V}_{\mathsf{wi}}$ on input the statement x, $\pi = (\{\Sigma_i^1, \Sigma_i^2, \Sigma_i^3\}_{i=1}^\tau, \mathbf{B}')$, and a blockchain $\tilde{\mathbf{B}}$. If the messages $\{\Sigma_i^1\}_{i=1}^\tau$ are not posted on the blockchain $\mathbf{B}'$ $\mathcal{V}_{\mathsf{wi}}$ outputs 0 otherwise she continues with the following steps.

Let B^* be the last block of the blockchain $\mathbf{B}'$ where the messages $\{\Sigma_i^1\}_{i=1}^\tau$ are posted. Let $B_1, \ldots, B_t$ be t blocks of the blockchain $\mathbf{B}'$ after B^*. $\mathcal{V}_{\mathsf{wi}}$ computes $\{\Sigma_i^2\}_{i=1}^\tau = \mathtt{ExtProc}(\mathsf{E}_{n,\lambda}, B_1, \ldots, B_t)$ and outputs 1 if the following conditions are satisfied:

[15] For this construction we require that the messages of Π_Σ are small enough to be posted in a block of the blockchain.

[16] q is s.t. $q \geq n_0(\lambda)$ where $(\mu(\cdot), n_0(\cdot))$ are the chain quality parameters of Γ^{V}.

1. $\mathbf{B}' \leq \tilde{\mathbf{B}}$;
2. The blockchain $\mathbf{B}'$ contains the messages $\{\Sigma_i^1, \Sigma_i^3\}_{i=1}^{\tau}$, and the messages $\{\Sigma_i^3\}_{i=1}^{\tau}$ are posted at least t blocks after B^*;
3. $\mathcal{V}_{\Sigma}(x, \Sigma_i^1, \Sigma_i^2, \Sigma_i^3) = 1$ for $i = 1, \ldots, \tau$.

Theorem 2. *If Assumption 1 holds, then $\Pi_{\mathsf{wi}} = (\mathcal{P}_{\mathsf{wi}}, \mathcal{V}_{\mathsf{wi}})$ is a publicly verifiable on-chain adaptive-input witness indistinguishable argument of knowledge over a blockchain protocol $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GetHash})$ for $\mathcal{NP}$-language $\mathcal{L}$.*

The proof is almost identical to the one showed for Theorem 1 and therefore we omit it.

A Note on the Delayed-Input Property of Π_{wi}. Fixing any x, w, s.t. $\mathcal{R}(x, w) = 1$, as for the off-chain construction also in this construction $\mathcal{P}_{\mathsf{wi}}$ is using x, w just to compute the 5th step of Π_{wi}. Therefore, Π_{wi} can compute the first 4 steps of Π_{wi} as a preprocessing phase, without knowing x or w (just the size is required). Then when (in any point in the future) x, w will be available $\mathcal{P}_{\mathsf{wi}}$ computes the last 2 steps of Π_{wi}.

We also want to point out that Theorem 2 holds for any WI Σ-protocol, even without the delayed-input property. However in this case x, w are needed by $\mathcal{P}_{\mathsf{wi}}$ already when she computes the 1st step of Π_{wi}.

On the Instantiation of the Adaptive-Input Special-Sound Π_{Σ}. We note that the work of [12] shows a compiler that works for a class of delayed-input perfect Σ-protocol described in [14,15,42]. This compiler on input a perfect Σ-protocol Π outputs a variant of Π that is adaptive-input special sound. The compiler does not require any additional assumption.

4 On Publicly Verifiable Zero Knowledge via [29]

Our publicly verifiable WI argument of knowledge in the blockchain model focuses on using a blockchain (as much as possible) as a black-box, therefore using the generic properties that a blockchain offers (in a black-box sense), such as chain consistency and chain quality, and some other natural assumptions that seemingly make sense with respect to known real-world blockchains. A natural challenging open question consists of obtaining a publicly verifiable zero-knowledge argument using a generic blockchain. The reason why we see this very challenging is that there are several subtleties that seem to be very non-trivial to address without making strong assumptions on the underlying blockchain protocol, and therefore losing generality.

Consider the NIZK constructed in [29]. Their construction works for proof-of-stake based blockchains only and the underlying assumption is that no adversary can control the majority of stake, at any point in time, and thus she cannot compute a fork. This assumption is leveraged in the zero-knowledge proof where one assumes that the simulator, controlling the honest parties, controls a majority of the stake (technically the secret keys associated to the public addresses owning a

majority of the stake), and this information can be used to compute a fork at any point in time. Given such special power for the simulator, the zero-knowledge argument of [29] consists of a set of n encryptions $e_1, \ldots, e_n$ and a NIWI proof for the theorem: "$(e_1, \ldots, e_n)$ are valid encryptions under public keys of n stakeholders" AND "either they are encryptions of shares of a witness for $x \in L$ or of shares of a valid fork of the blockchain". One of the most appealing properties of this scheme is that the size of the NIZK is independent of the number of total stakeholders, but depends only on parameters concerning the blockchain chain-quality property.

First of all notice that construction of [29] focuses on proof-of-stake based blockchains in order to have a proof that can be sound for ever. Indeed the same approach would fail if a proof-of-work is used instead of a proof-of-stake since clearly in the future an adversary would be able to compute a fork in the past, and therefore an accepting proof of a false theorem.

We note, however, that the approach of [29] has a subtle issue that prevents this construction to be usable in generic proof-of-stake blockchains. The issue stems from the fact that the non-interactive proof consists of encryptions of shares of the witness under the public keys of n stakeholders. The idea behind this approach was that as long as the majority of such keys belongs to a honest "stake" (and assume that the latter will never collude), one can assume that the adversary will never collect enough keys to decrypt the witness. However, this assumption seems to be unsubstantiated in general, if we don't make any assumption on the proof-of-stake blockchain protocol. To see why, assume that honest stakeholders decide to refresh their keys often, in particular, assume that upon each transaction they decide to move their stake from a public key pk_i to a freshly computed public key pk'_i and, in order to publicly disable the old pk_i, they will simply publish sk_i. This behavior could even be required in the blockchain protocol, and therefore always executed by honest parties. Note that, in this case, the assumption on the majority of stake is still preserved. Indeed, the majority of stake is still controlled by the honest parties. However, the keys have *evolved* and thus the keys used at time t in a zero-knowledge proof might be completely exposed at time $t + \delta$ (for some $\delta > 0$) thus invalidating the ZK property. We note that this issue exists even in presence of static adversaries (which is the assumption in [29]) since the honest parties remain honest parties throughout, they simply change their keys and this is not prohibited by the blockchain protocol (and in general it could be even enforced).

More in general, the scenario described above suggests that the above approach to design a non-interactive zero-knowledge proof system cannot retain any security in presence of an adversary who can somehow obtain the keys *after* having observed the zero-knowledge proof. Even assuming that keys do not evolve over time (and a party would never expose her old secret on the blockchain), there are few realistic scenarios that would allow an adversary to obtain such keys, in a blockchain setting. In such setting is indeed more natural to assume that the adversary is adaptive, and the corrupted parties can be chosen over time, for example, depending on the content of the blockchain, or the stake

gained or lost by a certain key. Since parties are rational, it might be convenient to them to "sell" their secret keys with lower stake in exchange for a public key with slightly higher stake. Thus, assuming that a zero-knowledge proof was computed using keys $(k_{i_1}, \ldots, k_{i_n})$, an adversary could target such keys, and at later stage, when the total stake of the system has increased, the adversary can corrupt the stakeholders associated to those keys, in such a way that the adversary still does not possess the majority of the stake – and thus the proof of stake assumption is not invalidated– but she has enough information to break the zero-knowledge property (for instance in the case of [29] the adversary has enough informations to decrypt the witness).

Finally we also remark that current blockchains exist because of the rewards that participants hope to obtain sharing their resources for the execution of the blockchain protocol. It is therefore natural to think that an honest party would be fine with giving up the secret key corresponding to a currently empty wallet receiving back a revenue. It is completely unknown to an honest party of a blockchain protocol the fact that there could be a cryptographic protocol designed on top of the blockchain that relies on honest parties keeping private some secret keys for ever, even in case they do not have any value.

A Standard Tools

Definition 9 (One-way function (OWF)). *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is called one way if the following two conditions hold:*

- *there exists a deterministic polynomial-time algorithm that on input* y *in the domain of* f *outputs* $f(y)$*;*
- *for every* PPT *algorithm* $\mathcal{A}$ *there exists a negligible function* ν*, such that for every auxiliary input* $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$*:*

$$\Pr\left[\, y \leftarrow \{0,1\}^* : \mathcal{A}(f(y), z) \in f^{-1}(f(y)) \,\right] < \nu(\lambda).$$

We say, also, that a OWF f *is a* one-way permutation (OWP) *if* f *is a permutation.*

Definition 10 (Hash Function [32]). *An hash function is a pair of* PPT *algorithms* $\Pi = (\mathtt{Gen}, H)$ *fulfilling the following:*

- $\mathtt{Gen}$ *is a probabilistic algorithm which takes as input a security parameter* λ *and outputs a key* s*.*
- *There exists* $l = \mathsf{poly}(\lambda)$ *such that* H *is (deterministic) polynomial time algorithm that takes as input a key* s *and any string* $x \in \{0,1\}^*$ *and outputs a string* $H(s,x) \in \{0,1\}^l$*.*

Definition 11 (Collision-Resistant Hash Functions (CRHFs) [32]). *A hash function* $\Pi = (\mathtt{Gen}, H)$ *is collision resistant if for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\Pr\left[\, H(s,x) = H(s,x') \wedge x \neq x' : s \leftarrow \mathtt{Gen}(1^\lambda), (x,x') \leftarrow \mathcal{A}(s) \,\right] \leq \nu(\lambda)$$

In this paper we denote by $\mathsf{h}(\cdot)$ a CRHFs where the description of the hash function (i.e., the key s) is publicly available either in the blockchain protocol or in the genesis block of the blockchain.

Definition 12 (Witness Indistinguishable (WI)). *An argument/proof system $\Pi = (\mathcal{P}, \mathcal{V})$, is* Witness Indistinguishable (WI) *for a relation $\mathcal{R}$ if, for every malicious* PPT *verifier $\mathcal{V}^*$, there exists a negligible function ν such that for all x, w, w' such that $(x, w) \in \mathcal{R}$ and $(x, w') \in \mathcal{R}$ it holds that:*

$$\left| \Pr \langle \mathcal{P}(w), \mathcal{V}^* \rangle(x) = 1 - \Pr \langle \mathcal{P}(w'), \mathcal{V}^* \rangle(x) = 1 \right| < \nu(|x|).$$

Obviously one can generalize the above definitions of WI to their natural adaptive-input variants, where the adversarial verifier can select the statement and the witnesses adaptively, before the prover plays the last round. We note that [23] prove that WI is preserved under self-concurrent composition, i.e. when multiple instance of Π are played concurrently.

Definition 13 (Proof/argument system). *A pair of* PPT *interactive algorithms $\Pi = (\mathcal{P}, \mathcal{V})$ constitute a* proof system *(resp., an* argument system*) for an $\mathcal{NP}$-language L, if the following conditions hold:*

Completeness: *For every $x \in L$ and w such that $(x, w) \in \mathcal{R}_{\mathsf{L}}$, it holds that:*

$$\Pr [\ \langle \mathcal{P}(w), \mathcal{V} \rangle(x) = 1\ ] = 1.$$

Soundness*: For every interactive (resp.,* PPT *interactive) algorithm $\mathcal{P}^\star$, there exists a negligible function ν such that for every $x \notin L$ and every z:*

$$\Pr [\ \langle \mathcal{P}^\star(z), \mathcal{V} \rangle(x) = 1\ ] < \nu(|x|).$$

A proof/argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an $\mathcal{NP}$-language L, enjoys *delayed-input* completeness if $\mathcal{P}$ needs x and w only to compute the last round and $\mathcal{V}$ needs x only to compute the output. Before that, $\mathcal{P}$ and $\mathcal{V}$ run having as input only the size of x. The notion of delayed-input completeness was defined in [12].

An interactive protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is *public coin* if, at every round, $\mathcal{V}$ simply tosses a predetermined number of coins (i.e. a random challenge) and sends the outcome to the prover. Moreover we say that the transcript τ of an execution $b = \langle \mathcal{P}(z), \mathcal{V} \rangle(x)$ is *accepting* if $b = 1$.

A *3-round protocol* $\Pi = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is an interactive protocol played between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ on common input x and private input w of $\mathcal{P}$ s.t. $(x, w) \in \mathcal{R}$. In a 3-round protocol the first message $\mathtt{a}$ and the third message $\mathtt{z}$ are sent by $\mathcal{P}$ and the second messages $\mathtt{c}$ is played by $\mathcal{V}$. At the end of the protocol $\mathcal{V}$ decides to accept or reject based on the data that he has seen, i.e. $x, \mathtt{a}, \mathtt{c}, \mathtt{z}$.

We usually denote the message $\mathtt{c}$ sent by $\mathcal{V}$ as a *challenge*, and as *challenge length* the number of bit of $\mathtt{c}$.

Definition 14 (Σ-Protocol). *A 3-round public-coin protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is a Σ-Protocol if the following properties hold:*

- *Completeness: if $(\mathcal{P}, \mathcal{V})$ follow the protocol on input x and private input w to $\mathcal{P}$ s.t. $(x, w) \in \mathcal{R}$, $\mathcal{V}$ always accepts.*
- *Special soundness: if there exists a polynomial time algorithm such that, for any pair of accepting transcripts on input x, $(\mathtt{a}, \mathtt{c}_1, \mathtt{z}_1)$ $(\mathtt{a}, \mathtt{c}_2, \mathtt{z}_2)$ where $\mathtt{c}_1 \neq \mathtt{c}_2$, outputs witnesses w such that $(x, w) \in \mathcal{R}$.*
- *Special Honest Verifier Zero-knowledge (SHVZK): there exists a* PPT *simulator algorithm S that for any $x \in L$, security parameter λ and any challenge $\mathtt{c}$ works as follow: $(\mathtt{a}, \mathtt{z}) \leftarrow \mathsf{S}(1^\lambda, x, \mathtt{c})$. Furthermore, the distribution of the output of S is computationally indistinguishable from the distribution of a transcript obtained when $\mathcal{V}$ sends S as challenges and $\mathcal{P}$ runs on common input x and any w such that $(x, w) \in \mathcal{R}$.*

Definition 15. *A perfect Σ-Protocol is Σ-Protocol that satisfies a strong SHVZK requirement, that is:*

Perfect Special Honest Verifier Zero-knowledge: there exists a PPT *simulator algorithm S that for any $x \in L$, security parameter λ and any challenge $\mathtt{c}$ works as follow: $(\mathtt{a}, \mathtt{z}) \leftarrow \mathsf{S}(1^\lambda, x, \mathtt{c})$. Furthermore, the distribution of the output of S is perfect indistinguishable from the distribution of a transcript obtained when $\mathcal{V}$ sends S as challenges and $\mathcal{P}$ runs on common input x and any w such that $(x, w) \in \mathcal{R}$.*

Theorem 3 [16]**.** *Every perfect Σ-protocol is perfect WI.*

Theorem 4 [25]**.** *The OR-composition of Σ-Protocols is WI.*

Definition 16. *A delayed-input 3-round system $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ enjoys* adaptive-input special soundness *if there exists a polynomial time algorithm* `Ext` *such that, for any pair of accepting transcripts $\mathtt{a}, \mathtt{c}_1, \mathtt{z}_1$ for input x_1 and $\mathtt{a}, \mathtt{c}_2, \mathtt{z}_2$ for input x_2 with $\mathtt{c}_1 \neq \mathtt{c}_2$, outputs witnesses w_1 and w_2 such that $(x_1, w_1) \in \mathcal{R}$ and $(x_2, w_2) \in \mathcal{R}$.*

Definition 17 (Proof of Knowledge [37]**).** *A protocol that is complete $\Pi = (\mathcal{P}, \mathcal{V})$ is a* proof of knowledge *(PoK) for the relation $\mathcal{R}_\mathsf{L}$ if there exist a probabilistic expected polynomial-time machine* `Ext`*, called the extractor, such that for every algorithm $\mathcal{P}^\star$, there exists a negligible function ν, every statement $x \in \{0,1\}^\lambda$, every randomness $r \in \{0,1\}^\star$ and every auxiliary input $z \in \{0,1\}^\star$,*

$$\Pr\left[\, \langle \mathcal{P}^\star_r(z), \mathcal{V} \rangle(x) = 1 \,\right] \leq \Pr\left[\, w \leftarrow \mathtt{Ext}^{\mathcal{P}^\star_r(z)}(x) : (x, w) \in \mathcal{R} \,\right] + \nu(\lambda).$$

We also say that an argument system Π is a argument of knowledge *(AoK) if the above condition holds w.r.t. any* PPT *$\mathcal{P}^\star$.*

In this paper we also consider the *adaptive-input* PoK/AoK property for all the protocols that enjoy delayed-input completeness. Adaptive-input PoK/AoK ensures that the PoK/AoK property still holds when a malicious prover can choose the statement adaptively at the last round.

Definition 18. *Let X, Y be two random variables that takes values in V (i.e., V is the union of supports of X and Y). The statistical distance between X and Y is defined as follows:*

$$\frac{1}{2}\sum_{v\in V}|\Pr[\,X=v\,]-\Pr[\,Y=v\,]|.$$

Definition 19 [36] *[s - Source Extractor]. A function $\mathsf{E}_{n,\lambda} : \{\{0,1\}^n\}^s \to \{0,1\}^m$ is an extractor for independent (n,λ) sources that uses s sources and outputs m bits with error ϵ, if for any s independent (n,λ) sources $X_1, X_2, \ldots, X_s$, we have that*

$$|\mathsf{E}_{n,\lambda}(X_1, X_2, \ldots, X_s) - \mathcal{U}_m| \leq \epsilon$$

where $|\cdot|$ denotes the statistical distance.

The author of [36] gave a construction of a 3-source extractor, with parameters $\lambda \geq \log^{12} n$, $m = 0.9\lambda$ and $\epsilon = 2^{-\lambda^{\omega(1)}}$.

References

1. Badertscher, C., Garay, J., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? A rational protocol design treatment of bitcoin. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 34–65. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_2
2. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: a composable treatment. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 324–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_11
3. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, 18–21 May 2014, pp. 459–474. IEEE Computer Society (2014)
4. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. CoRR abs/1605.04559 (2016). http://arxiv.org/abs/1605.04559
5. Bitansky, N., Paneth, O.: ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 401–427. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_16
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pp. 103–112. ACM, New York (1988)
7. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. IACR Cryptology ePrint Archive 2015, 1015 (2015). http://eprint.iacr.org/2015/1015
8. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, Las Vegas, Nevada, USA, 14–17 October 2001, pp. 136–145. IEEE Computer Society (2001)
9. Cardano: https://www.cardano.org/en/home/

10. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 711–742. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_24
11. Ciampi, M., Persiano, G., Scafuro, A., Siniscalchi, L., Visconti, I.: Improved OR-composition of sigma-protocols. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 112–141. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_5
12. Ciampi, M., Persiano, G., Scafuro, A., Siniscalchi, L., Visconti, I.: Online/offline OR composition of sigma protocols. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 63–92. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_3
13. Ciampi, M., Persiano, G., Siniscalchi, L., Visconti, I.: A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 83–111. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_4
14. Cramer, R.: Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, University of Amsterdam (1996)
15. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055745
16. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
17. Damgård, I.: On Σ-protocol (2010). http://www.cs.au.dk/~ivan/Sigma.pdf
18. Dwork, C., Naor, M.: Zaps and their applications. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, Redondo Beach, California, USA, 12–14 November 2000, pp. 283–293 (2000)
19. Ethereum: https://www.ethereum.org/
20. Feige, U.: Alternative models for zero knowledge interactive proofs. Master's thesis (1990). Ph.D. thesis
21. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, 22–24 October 1990, vol. I, pp. 308–317. IEEE Computer Society (1990)
22. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_46
23. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Ortiz, H. (ed.) Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pp. 416–426. ACM, New York (1990)
24. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
25. Garay, J.A., MacKenzie, P., Yang, K.: Strengthening zero-knowledge protocols using signatures. J. Cryptology **19**(2), 169–209 (2006)

26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, Berkeley, CA, USA, 26–29 October 2013, pp. 40–49 (2013)
27. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, CA, USA, 1–4 June 2013, pp. 467–476 (2013)
28. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
29. Goyal, R., Goyal, V.: Overcoming cryptographic impossibility results using blockchains. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 529–561. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_18
30. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_18
31. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_6
32. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press, Boca Raton (2007)
33. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols. IACR Cryptology ePrint Archive 2015, 1019 (2015)
34. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
35. Lapidot, D., Shamir, A.: Publicly verifiable non-interactive zero-knowledge proofs. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 353–365. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_26
36. Li, X.: Three-source extractors for polylogarithmic min-entropy. In: IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October 2015, pp. 863–882 (2015)
37. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: Fortnow, L., Vadhan, S.P. (eds.) Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, pp. 705–714. ACM, New York (2011)
38. Lindell, Y.: An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 93–109. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_5
39. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_10
40. Lipmaa, H.: Efficient NIZK arguments via parallel verification of benes networks. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 416–434. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_24
41. Lipmaa, H., Zhang, B.: A more efficient computationally sound non-interactive zero-knowledge shuffle argument. J. Comput. Secur. **21**(5), 685–719 (2013)
42. Maurer, U.: Zero-knowledge proofs of knowledge for group homomorphisms. Des. Codes Crypt. 1–14 (2015). http://dx.doi.org/10.1007/s10623-015-0103-5

43. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_32
44. Micali, S.: Computationally sound proofs. SIAM J. Comput. **30**(4), 1253–1298 (2000). https://doi.org/10.1137/S0097539795284959
45. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008, unpublished)
46. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
47. Pass, R., Shi, E.: FruitChains: a fair blockchain. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, 25–27 July 2017, pp. 315–324 (2017)
48. Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 380–409. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_14
49. Ripple: https://ripple.com/
50. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22