Blockchains from Non-idealized Hash Functions

Juan A. Garay[1], Aggelos Kiayias[2,3](✉), and Giorgos Panagiotakos[1,4]

[1] Department of Computer Science and Engineering, Texas A&M University, College Station, USA
garay@cse.tamu.edu

[2] School of Informatics, University of Edinburgh, Edinburgh, UK
akiayias@inf.ed.ac.uk

[3] IOHK, Hong Kong, China

[4] Department of Informatics and Telecommunications, University of Athens, Athens, Greece
geo.panagiotakos@gmail.com

Abstract. The formalization of concrete, non-idealized hash function properties sufficient to prove the security of Bitcoin and related protocols has been elusive, as all previous security analyses of blockchain protocols have been performed in the random oracle model. In this paper we identify three such properties, and then construct a blockchain protocol whose security can be reduced to them in the standard model assuming a common reference string (CRS).

The three properties are: *collision resistance*, *computational randomness extraction* and *iterated hardness*. While the first two properties have been extensively studied, iterated hardness has been empirically stress-tested since the rise of Bitcoin; in fact, as we demonstrate in this paper, any attack against it (assuming the other two properties hold) results in an attack against Bitcoin.

In addition, iterated hardness puts forth a new class of search problems which we term *iterated search problems* (ISP). ISPs enable the concise and modular specification of blockchain protocols, and may be of independent interest.

1 Introduction

Blockchain protocols, introduced by Nakamoto [46], are seen as a prominent application of the "proof of work" (PoW) concept to the area of consensus protocol design. PoWs were initially introduced in the work of Dwork and Naor [27] as a spam protection mechanism, and subsequently found applications in other domains such as Sybil attack resilience [26] and denial of service protection [4,41], prior to their application to the domain of distributed consensus hinted at early on by Aspnes *et al.* [3].

A. Kiayias—Research partly supported by H2020 project PRIVILEDGE #780477.
G. Panagiotakos—Work partly done while at the University of Edinburgh.

R. Pass and K. Pietrzak (Eds.): TCC 2020, LNCS 12550, pp. 291–321, 2020.
https://doi.org/10.1007/978-3-030-64375-1_11

A PoW scheme is typified by a *proving* algorithm, that produces a solution given an input instance, as well as a *verification* algorithm that verifies the correctness of the witness with respect to the input. The fundamental property of a PoW scheme is that the proving algorithm allows for no significant shortcuts, i.e., it is hard to significantly make it more expedient, and hence any verified solution implies an investment of computational effort on behalf of the prover. Nevertheless, this "moderate hardness" property alone has been found to be insufficient for the utilization of PoWs in the context of various applications and other properties have been put forth to complement it. These include: (i) *amortization resistance*, which guarantees that the adversary cannot speed up the computation when solving multiple PoW instances together, and (ii) *fast verification*, which suggests a significant gap between the complexities of the proving and verification algorithms.

Despite the evolution of our understanding of the PoW primitive, as exemplified in recent works (e.g., [1,6,13,36]), there has been no definitive analysis of the primitive in the context of blockchain protocol security in the standard model. Intuitively, PoWs are useful in the consensus setting because they make message passing (moderately) hard and hence generate stochastic opportunities for the parties running the protocol to unify their view of the current state of the system. This fundamentally relies on an assumption about the aggregate computational power of the honest parties, but not on their actual number, in relation to the computational power of the parties that may deviate from the protocol (the "Byzantine" parties)—a hallmark of the peer-to-peer setting Bitcoin is designed for. Despite the fact that the Bitcoin blockchain has been analyzed formally [5,31,33,49], the required PoW properties have not been fully identified and most of the existing analysis has been carried out in the random oracle (RO) model [10]. The same is true for a wide variety of other protocols in the space, including [2,34,42].

We stress that despite the fact that the RO model has been widely used in the security analysis of practical protocols and primitives, it has also received significant criticism. For example, Canetti *et al.* [20] showed that there exist implementations of signatures and encryption schemes that are secure in the RO model but insecure for any implementation of the RO in the standard model; Nielsen [47] proved that efficient non-committing encryption has no instantiation in the standard model but a straightforward implementation in the RO model, while Goldwasser and Kalai [40] showed that the Fiat-Shamir heuristic [29] does not necessarily imply a secure digital signature, which is in contrast with the result by Pointcheval and Stern [50] in the RO model.

It follows that it is critical to discover security arguments for blockchain protocols that do not rely on the RO model. Note that we are looking for arguments as opposed to proofs since it is easy to observe that some computational assumption would still be needed for deriving the security of a blockchain protocol (recall that blockchain security cannot be inferred information theoretically as it fundamentally requires at minimum the collision resistance of the underlying hash function). In fact, the formalization of non-idealized, concrete hash function assumptions sufficient to prove security of Bitcoin and related protocols has been identified as a "fascinating open question" [18].

Following the above, the main question that motivates the present work is the following:

Is it possible to prove the security of blockchain protocols in the standard model under non-idealized assumptions about the underlying hash function?

Our Results. In this paper we answer the above question in the positive, by identifying three properties of a hash function family $\{H_k(\cdot)\}_k$ and then constructing a blockchain protocol whose security can be reduced to these properties (together with NIZKs; see below) in the standard model.

The first property is *collision resistance*. Specifically, it should be hard for an adversary given a random key k, to find two distinct messages m, m' for which it holds $H_k(m) = H_k(m')$. This property is useful in the blockchain context, since intuitively collision resistance ensures that the hash-chain maintained by the parties ensures the chronologically correct encoding of information.

The second property of the underlying hash function family is that it should be *computational randomness extracting* (CRE). Specifically, there is a way to isolate a finite subset of the domain of the hash function family so that for any given key k, the function H_k is a (weak) computational randomness extractor. This property is useful in a few different ways in blockchain security. Firstly, it will help for symmetry breaking, making sure that parties work concurrently on independent instances of the underlying problem. Secondly, it will ensure that the problem instances generated by honest parties (in the form of new blocks), will be sufficiently unpredictable in the eyes of the adversary. Regarding the plausibility of a CRE hash function, note that pseudorandom functions (PRFs) are known to imply weak computational randomness extractors [22], and assuming that a hash function implies a PRF is a fairly standard assumption [7,25,43].

The third property asks for the *iterative hardness* of the underlying hash function as multiple pre-images with near-zero hashes are stringed together in the form of a chain. This assumption is implicit in the context of the Bitcoin protocol. In fact, as we show, an attack against iterative hardness would result in an attack against the protocol (assuming a CRE hash function). This implies that there is (monetary) incentive to break this assumption, which coupled with the fact that no significant attacks have been demonstrated in the context of the Bitcoin protocol, establishes iterated hardness of the underlying hash (in this case SHA-256) as a plausible assumption.[1]

Armed with the above, we show a novel blockchain protocol whose security can be reduced to the collision resistance, computational randomness extraction and iterative hardness of the underlying hash function. Our design adopts Bitcoin's hash-based blockchain structure, as well as the longest-chain selection rule. However, contrary to previous analyses of this type of protocols [5,31,49]

[1] Refer to Sect. 3 for further discussion on this assumption.

in the RO model, iterative hardness provides no guarantee that blocks are "non-malleable," in the sense that it may be easy to mine multiple blocks on the same height of the chain once you have mined the first one. Our solution is to instead construct a PoW that *is* malleable, and leverage it to show a reduction that breaks the underlying iterated hardness assumption given a common-prefix attack to the blockchain protocol. In order to achieve this, we also have to hide the block witnesses by taking advantage of NIZK proofs with efficient simulation, thus managing to efficiently extract a sequence of iterated witnesses despite the fact that the attacker may not produce consecutive blocks.

In order to describe and analyze the protocol modularly, we put forth a new class of search problems, which we call *iterated search problems* (ISP). Taking advantage of ISPs one can produce concise and modular specifications of blockchain protocols, as evidenced by the description of our protocol (Sect. 4.3); as such, ISPs can be of independent interest.

In a nutshell, an ISP instance is defined by a problem statement set X, a witness set W and a relation R that determines when a witness satisfies the problem statement. The ISP is also equipped with a *successor* algorithm S that given a statement x and a witness w, can produce a successor problem statement x'; a *solving* algorithm M which given an initial problem statement x can find a sequence of witnesses; and a *verification* algorithm V that takes a problem statement x and witness w and outputs 1 if $(x, w) \in R$, and 0 otherwise. Each witness corresponds to the next statement defined by algorithm S on input the previous statement and witness, starting from x. The *iterated hardness* property of the ISP asks that if the solving algorithm takes t steps to solve k instances iteratively, no alternative algorithm can substantially speed this process up and produce k iterative solutions with non-negligible probability.

We perform our analysis in the static-adversary setting with synchronous rounds as in [31], and prove that our protocol can thwart adversaries and environments that roughly take less than half the computational steps the honest parties collectively are allowed per round. To our knowledge this is the first work that achieves such a result in the permissionless setting without idealized assumptions and no PKI[2]. In principle we can extend our results to the Δ-synchronous setting of [49], following the techniques found in Section 7 of [32]; we leave the details to the full version of the paper. Further, we leave as an open question the extension of our results to the dynamic setting of [33], as well as matching the (less than) 50% threshold on adversarial computational power of the Bitcoin blockchain which can be shown in the RO model.

Related Work. A related but distinct notion of hardness is *sequential* (i.e., non-parallelizable) iterated hardness. This notion has been considered as early as [51], mainly in the domains of timed-release cryptography [15] and protocol fairness [37], and recently formalized in [14] under the term *iterated sequential functions* (ISF) in the context of Verifiable Delay Functions (VDFs). In addition, a number of candidate hard problems have been proposed, including squaring

[2] See [30] for an extensive discussion on known results in the peer-to-peer/diffusion setting.

a group element of composite-modulus groups [51], hashing, and computing the modular square root of an element on a prime order group [44]. Nevertheless, we observe that if we base the Bitcoin protocol on an ISF (or VDF for that matter) it will be *insecure*. The fundamental issue is that it does not allow for parallelization, which is crucial for proving the security of any (Bitcoin-like) blockchain protocol. Indeed, an attacker with a single processor whose sequential speed is slightly faster than that of honest parties, can outperform potentially hundreds of them and mine longer chains first.

Another notion related to iterative hardness is the notion of "correlation intractability" (CI) [18]. The difference is that while CI only bounds the success probability in solving a single challenge, ISP fundamentally requires multiple instances. Further, while CI talks about any sparse relation, the iterative hardness definition is concerned with a specific non-sparse relation.

Finally, another related work focusing on sufficient conditions for the consensus problem in the permissionless setting (and no PKI, while matching the less than 50% threshold on adversarial computational power) is [36], which introduced the concept of *signatures of work* (SoW) as the basic underlying assumption. The only known implementation of SoWs however is in the RO model, hence it is unknown (and an interesting open question) whether SoWs can be realized under non-idealized hash function assumptions like the ones we consider here.

Due to space limitations, most of the proofs are presented in the full version [35] of the paper.

2 Preliminaries

In this section we present basic notation and definitions that we will use in the rest of the paper.

For $k \in \mathbb{N}^+$, $[k]$ denotes the set $\{1, \ldots, k\}$. For strings x, z, $x||z$ is the concatenation of x and z, and $|x|$ denotes the length of x. We denote sequences by $(a_i)_{i \in I}$, where I is the index set which will always be countable. For a set X, $x \leftarrow X$ denotes sampling a uniform element from X. For a distribution $\mathcal{U}$ over a set X, $x \leftarrow \mathcal{U}$ denotes sampling an element of X according to $\mathcal{U}$. By $\mathcal{U}_m$ we denote the uniform distribution over $\{0,1\}^m$. For random variable X, we denote by $H_\infty(X)$ the min-entropy of X. We denote the statistical distance between two random variables X, Z with range U by $\Delta[X, Z]$, i.e., $\Delta[X, Z] = \frac{1}{2}\sum_{v \in U} |\Pr[X = v] - \Pr[Z = v]|$. A *random variable ensemble* $(X_i)_{i \in I}$, is a sequence of random variables indexed by I. By $(X_i)_i \approx (Z_i)_i$ (resp. $\overset{c}{\approx}$) we denote that two ensembles are statistical (resp. computational) indistinguishable. We let λ denote the security parameter.

Protocol Execution and Security Model. In this paper we will follow a more concrete approach [8,11,12,37] to security evaluation. We will use functions t, ϵ, whose range is $\mathbb{N}, \mathbb{R}$, respectively, and have possibly many different arguments, to denote concrete bounds on the running time (number of steps) and probability of adversarial success of an algorithm in some given computational model, respectively. When we speak about running time this will include the execution time plus the length of the code (cf. [12]; note also that we will be considering uniform machines). We will always assume that t is a polynomial in the security parameter λ, although we will sometimes omit this dependency for brevity.

Instead of using interactive Turing machines (ITMs) as the underlying model of distributed computation, we will use (interactive) RAMs. The reason is that we need a model where subroutine access and simulation do not incur a significant overhead. ITMs are not suitable for this purpose, since one needs to account for the additional steps to go back-and-forth all the way to the place where the subroutine is stored. A similar choice was made by Garay *et al.* [37]; refer to [37] for details on using interactive RAMs in a UC-like framework. Given a RAM M, we will denote by $\mathsf{Steps}_M(1^\lambda, x)$ the random variable that corresponds to the number of steps taken by M given input 1^λ and x. We will say that M is t-bounded if it holds that $\Pr[\mathsf{Steps}_M(1^\lambda, x) \leq t(\lambda)] = 1$.

Finally, we remark that in our analyses there will be asymptotic terms of the form $\mathsf{negl}(\lambda)$ and concrete terms; throughout the paper, we will assume that λ is large enough to render the asymptotic terms insignificant compared to the concrete terms.

The Bitcoin Backbone Model. In this section, we give an overview of the security model that we are going to use throughout this work, introduced in [36]. This model is a variant of the synchronous model presented in [31] for the analysis of the Bitcoin backbone protocol, extended to accommodate a standard-model analysis of PoW-based blockchain protocols. In turn the model of [31] is based on Canetti's formulation of "real world" execution for multi-party cryptographic protocols [16,17].

An execution of some protocol Π is defined with respect to an "environment" program $\mathcal{Z}$, a "control" program C, and an "adversary" program $\mathcal{A}$. At a high level, $\mathcal{Z}$ is responsible for providing inputs to and obtaining outputs from different instances of Π, C is responsible for supervising the spawning and communication of all these programs, and $\mathcal{A}$ aims to disrupt the goals set by the protocol. The programs in question can be thought of as "interactive RAMs" communicating through registers in a well-defined manner.

We consider executions where the set of of parties $\{P_1, ..., P_n\}$ running Π is fixed and hardcoded to C. Moreover, we consider a "hybrid" model of computation [19], where the adversary $\mathcal{A}$ as well as all parties in the execution can access a number of "ideal" functionalities as subroutines; the functionalities are also modeled as RAMs and are presented later in detail. Initially $\mathcal{Z}$ is activated. $\mathcal{Z}$ can make special requests that result in the spawning of different parties and $\mathcal{A}$. In turn, $\mathcal{A}$ can corrupt different parties by sending messages of the form (Corrupt, P_i) to C, with the limitation that the total number of parties corrupted should be at most t; t is a parameter of the execution. We assume an active static adversary.

We are working in the *synchronous* model of computation, where the current round is known to all parties, and messages sent at one round are received at the beginning of the next one. The influence of the adversary in the network is going to be actively malicious following standard cryptographic practice. While we assume the adversary to be rushing and communication not to be authenticated, messages sent by honest parties are guaranteed to reach their destination.

All the above concerns are captured by the diffusion functionality $\mathcal{F}_{\mathsf{diff}}$. The functionality maintains a Receive string defined for each party P_i. A party is allowed at any moment to fetch the messages sent to it at the previous round that are contained in its personal Receive string. Moreover, when the functionality receives an instruction to diffuse a message m from party P_i, it marks the party as complete for the current round, and forwards the message to the adversary; note that m is allowed to be empty. At any moment, the adversary $\mathcal{A}$ is allowed to specify the contents of the Receive string for each party P_i. The adversary has to specify when it is complete for the current round. When all parties are complete for the current round, the functionality inspects the contents of all Receive tapes and includes any messages that were diffused by the parties in the current round but not contributed by the adversary to the Receive tapes. The variable round is then incremented. In the protocol description, we will use Diffuse as the message transmission command.

In addition, we assume the existence of a *common reference string* (CRS) functionality that samples the CRS in a trusted manner from a known efficiently samplable distribution, and is available for all parties to fetch at the start of the execution. Note, that from our modeling it is implicit that the adversary and the honest parties get access to the CRS at the same round.

Based on the above, we denote by $\{\text{VIEW}^{P,t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ the random variable ensemble that corresponds to the view of party P at the end of an execution where $\mathcal{Z}$ takes z as input. We will consider stand-alone executions, hence z will always be of the form 1^λ, for $\lambda \in \mathbb{N}$. For simplicity, to denote this random variable ensemble we will use $\text{VIEW}^{P,t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$. By $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$ we denote the concatenation of the views of all parties. The probability space where these variables are defined depends on the coins of all honest parties, $\mathcal{A}$, $\mathcal{Z}$ and the CRS generation procedure.

Furthermore, we are going to define a predicate on executions and prove our properties in disjunction with this predicate, i.e., either the property holds or the execution is not good.

Definition 1. *Let $(t_{\mathcal{A}},\theta)$-good be a predicate defined on executions in the hybrid setting described above. Then E* is $(t_{\mathcal{A}},\theta)$-good*, where E is one such execution, if*

- *the total number of steps taken by $\mathcal{A}$ and $\mathcal{Z}$ per round is no more than $t_{\mathcal{A}}$;*[3]
- *the adversary sends at most θ messages per round.*

Definition 2. *Given a predicate Q and bounds $t_{\mathcal{A}},\theta,t,n \in \mathbb{N}$, with $t < n$, we say that protocol Π satisfies property Q for n parties assuming the number of corruptions is bounded by t, provided that for all PPT $\mathcal{Z},\mathcal{A}$, the probability that $Q(\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}})$ is false and the execution is $(t_{\mathcal{A}},\theta)$-good is negligible in λ.*

[3] The adversary cannot use the running time of honest parties that it has corrupted; it is activated instead of them during their turn.

Cryptographic Primitives and Building Blocks. We will make use of the following cryptographic primitives: Cryptographic hash functions, (computational) randomness extractors [22,48] and robust non-interactive zero-knowledge (NIZK) [52]. Refer to Appendix A for the corresponding security definitions.

Robust Public Transaction Ledgers. Our work is concerned with necessary and sufficient conditions to implement a public transaction ledger. Next, we give the transaction ledger definition introduced in [31], with the liveness property slightly strengthened, as in [49].

A *public transaction ledger* is defined with respect to a set of valid ledgers $\mathcal{L}$ and a set of valid transactions $\mathcal{T}$, each one possessing an efficient membership test. A ledger $\mathbf{x} \in \mathcal{L}$ is a vector of sequences of transactions $\mathrm{tx} \in \mathcal{T}$. Ledgers correspond to chains of blocks in the Bitcoin protocol. It is possible for the adversary to create two transactions that are conflicting; valid ledgers must not contain conflicting transactions. Moreover, it is assumed that in the protocol execution there also exists an oracle Txgen that generates valid transactions, and is unambiguous, i.e., the adversary cannot create transactions that come in 'conflict' with the transactions generated by the oracle. A transaction is called *neutral* if there does not exist any transactions that come in conflict with it. Any ledger that contains neutral or non-conflicting transactions is considered to be valid.

Definition 3. *A protocol Π implements a* robust public transaction ledger *if it organizes the ledger as a chain of blocks of transactions and it satisfies the following two properties:*

- **Consistency** (parameterized by the "depth" parameter $k \in \mathbb{N}$): *If in a certain round an honest player reports a ledger that contains a transaction* tx *in a block more than k blocks away from the end of the ledger, where $k \in \mathbb{N}$ is the "depth" parameter (such transactions are called* stable*), then* tx *will be reported as stable and in the same position in the ledger by any honest player from this round on.*
- **Liveness** (parameterized by $k, u \in \mathbb{N}$—the "depth" and "wait time" parameters, resp.): *For every u consecutive rounds, there exists a round and an honest party, such that the transactions given as input to that party at this round that are either (i) issued by* Txgen *or (ii) neutral, will be reported by all honest parties as stable at the end of this round interval.*

3 Hash Functions Properties for Blockchain Security

In this section we describe the three falsifiable assumptions about hash functions which the security of our protocol is going to be based on. Two of these properties, namely, collision resistance [23] and weak computational randomness extraction [22], have been extensively studied in the hash function literature. The third one is new, and has to do with the moderate hardness of computing *sequences* of small hashes. We proceed to discuss each of the properties in detail.

We start with collision resistance. Most known blockchain protocols make use of a collision-resistant hash function in order to establish basic structural properties, e.g., that the adversary cannot create a blockchain that contains a cycle. That is exactly the way we are going to use this property here. We will use the following security definition [39].[4]

Definition 4. *Let $\mathcal{H} = \{\{H_k : \{0,1\}^* \rightarrow \{0,1\}^\lambda\}_{k\in K(\lambda)}\}_{\lambda\in\mathbb{N}}$ be a hash-function family, and $\mathcal{A}$ be a PPT adversary. Then $\mathcal{H}$ is* **collision resistant** *if and only if for any $\lambda \in \mathbb{N}$ and corresponding $\{H_k\}_{k\in K}$ in $\mathcal{H}$,*

$$\Pr_{k\leftarrow K}[(m,m') \leftarrow \mathcal{A}(1^\lambda, k) : (m \neq m') \wedge (H_k(m) = H_k(m'))] \leq \mathsf{negl}(\lambda).$$

Our second security assumption has to do with the existence of a fixed-length-input hash function family that is a *weak* computational randomness extractor. As explained in [22], this assumption is weaker than assuming a fixed-length-input pseudorandom function family (FI-PRF), a common assumption in the hash function literature [7,25,43]. We adapt the definition of a weak computational randomness extractor to the context of a hash function family.

Definition 5. *Let $\mathcal{H} = \{\{H_k : \{0,1\}^{d\lambda} \rightarrow \{0,1\}^\lambda\}_{k\in K(\lambda)}\}_{\lambda\in\mathbb{N}}$, for some $d \in \mathbb{N}$, $d > 1$, be a fixed-length input hash-function family. $\mathcal{H}$ is a* **computational randomness extracting** *(CRE) hash function family if for some $c \in \mathbb{N}^+$, $c < d$, the function family $E = \{E_\lambda : \{0,1\}^{(c+1)\lambda} \times \{0,1\}^{(d-c-1)\lambda} \rightarrow \{0,1\}^\lambda\}_\lambda$, where $E_\lambda(x,i) \overset{\text{def}}{=} H_k(x||i)$, is a weak $(c\lambda)$-computational extractor, for any $k \in K(\lambda)$.*

This property will be useful in our protocol for two reasons. First, to ensure that the distributions of blocks generated by honest parties are identical and independent. Second, to establish that the blocks generated by honest parties, and which the adversary has the choice to mine on, look sufficiently random and hence the moderate hardness of the underlying problem is preserved.

Our third assumption about hash functions has to do with the hardness of finding sequences of small hashes in the hash-based (SHA-256) PoW construction proposed for Bitcoin. In more detail, given the hash x of some block, computing a valid PoW for this construction consists of finding witnesses w_1, w_2 such that $H_k(H_k(x||w_1)||w_2) < T$. In turn, our hardness property requires that any adversary should take a number of steps proportional to the number of PoWs computed, when these PoWs form a sequence starting from a uniformly random string x. The property is parameterized by t, the number of steps the adversary takes to generate each PoW on average.

Definition 6. *Let $\mathcal{H} = \{\{H_k : \{0,1\}^{d\lambda} \rightarrow \{0,1\}^\lambda\}_{k\in K(\lambda)}\}_{\lambda\in\mathbb{N}}$, for some $d \in \mathbb{N}$, $d > 1$, be a fixed-length input hash-function family, and let T be some hardness parameter. $\mathcal{H}$ is* **t-iteratively hard** *iff there exists a polynomial $k_0(\cdot)$, such that for any PPT RAM $(\mathcal{A}_1, \mathcal{A}_2)$, $\lambda \in \mathbb{N}$, and $k \geq k_0(\lambda)$, it holds that:*

[4] Throughout our exposition for simplicity we will assume that $\mathcal{H}$ takes one step to be evaluated. We note that our results can be generalized to the case where $\mathcal{H}$ takes more time.

$$\Pr_{\substack{\sigma \leftarrow K(\lambda);\\ x_0 \leftarrow [0,T]}} \left[\begin{array}{l} st \leftarrow \mathcal{A}_1(1^\lambda, \sigma); (w_i, w_i')_{i \in [k]} \leftarrow \mathcal{A}_2(1^\lambda, st, x_0) : \\ \forall i \in [k] : x_i := H_\sigma(H_\sigma(x_{i-1} || w_i) || w_i') < T \\ \wedge \ \mathsf{Steps}_{\mathcal{A}_2}(st, x_0) < k \cdot t \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Our choice to base the security of our protocol on the iterated hardness of Bitcoin's PoW construction is not accidental. The fact that any attack on iterated hardness implies an attack on Bitcoin, as we show in Appendix B, as well as the fact that no attacks have been publicly disclosed in the last ten years that this construction has been actively used in Bitcoin, constitute empirical evidence in its favor. Note that this would not necessarily be the case if we based security on a stronger hardness property that was not necessary to prove Bitcoin secure, as it would then be possible that an attack against the property is known and the adversary does not have any incentive to reveal/deploy it, as it does not affect the security of the protocol in any way.[5]

We note that to prove the security of our protocol both properties in Definitions 5 and 6 should hold for the same hash function and for suitable parameters [6], which we discuss in the next section; collision resistance may hold for a different hash function. As argued above, SHA-256 is a natural candidate for these assumptions. Finally, in our protocol analysis we will also make use of a number of other standard assumptions, such as the existence of a NIZK-PoK scheme and that the honest parties control the majority of the computational power. The theorem we prove is as follows:

Theorem 3 (Informal). *Assume the existence of collision-resistant hash functions, a hash function family that is CRE and iteratively hard for appropriate parameters, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and that $t_\mathcal{A}$ is (roughly) less than half the total running time of honest parties per round. Then there exists a protocol that implements a robust public transaction ledger.*

Finally, Gentry and Wichs [38] define as falsifiable the cryptographic assumptions that can be expressed as a game between an efficient challenger and an adversary. All cryptographic assumptions of Theorem 3 are falsifiable in this sense, with two caveats: First, due to the concrete security approach our work takes, the challenger should take as input the number of steps of the adversary. Second, in the computational randomness extraction property we quantify over all keys of the hash and all efficiently samplable distributions with sufficient min-entropy, which is not immediate to express in the framework of [38]. Instead, we could choose the key randomly, and expresses the extraction property w.r.t. a single family of source distributions that the adversary can influence. To simplify

[5] The profitability of an attack may also work as a counterincentive to revealing it. Nevertheless, there is merit in our argument if we take into consideration "white hat" actors who have tried breaking Bitcoin.

[6] Intuitively, the adversary should not be able to compute small hashes much faster than the rate at which honest parties generate blocks that is guaranteed by the computational extractor property.

our presentation we adopt the former version of the definition. However, we note that the proof techniques we use can be adapted to handle the latter.

4 Blockchains from Non-idealized Hash Functions

In this section we present and prove secure a protocol that implements a transaction ledger and is based on a hash function that satisfies the properties described in Sect. 3. We modularize our presentation and analysis by first introducing the concept of *iterated search problems* (ISP) in Sect. 4.1, and then presenting a technical overview in Sect. 4.2, followed by an ISP-based blockchain protocol in Sect. 4.3. Then, in Sect. 4.4, we introduce a "blockchain friendly" ISP security definition, that we show in Sect. 4.5 to be sufficient to prove our protocol secure. Finally, in Sect. 4.6 we construct a secure ISP based on the hash properties defined in Sect. 3, which in combination with our protocol can be shown to satisfy Theorem 3.

The choice of modularizing the protocol analysis has multiple benefits. In particular, it first allows us to formally capture all required properties that the moderately hard problem our protocol is built on should satisfy for the analysis to go through. We hope that this will motivate building other constructions in the future. Secondly, it makes it easier to take advantage of previous efforts to analyze relevant protocols [31,36,49]. While we adapt some of the proof techniques presented there, an important contribution of our work is that the ISP notion which we built on is considerably weaker and can be instantiated in the standard model from fairly simple assumptions.

4.1 Iterated Search Problems

In this section we introduce a class of problems inspired by Bitcoin's underlying computational problem. The straightforward properties that this class should have, are the ability to find a witness for a problem statement and to verify that the witness is correct, matching Bitcoin's block mining and block verification procedures, respectively. In addition, the notion models the ability to generate a new problem statement from a valid statement/witness pair. This captures the fact that in Bitcoin the problem that a miner solves depends on a previous block (i.e., a statement/witness pair). This concept has appeared before in the study of iterated *sequential* functions [14], whose name we draw from. Syntactically, the key difference here is that in each iteration we are not evaluating a function, but instead we are solving a search problem with possibly many witnesses. Moreover, as we already commented in Sect. 1 iterated sequential functions are *not* the correct abstractions for Bitcoin's underlying computational problem, as they allow for an attack against the protocol. We proceed to give a formal definition of ISPs.

Definition 7 (Iterated Search Problem). *An* iterated search problem *(ISP) $\mathcal{I}$ specifies a collection $(I_\lambda)_{\lambda\in\mathbb{N}}$ of distributions.*[7] *For every value of the*

[7] Here we follow the notation used in [21] to define *subset membership problems*. We remark that no other connection exists between the two papers.

security parameter $\lambda \geq 0$, I_λ is a probability distribution of instance descriptions. *An instance description Λ specifies*

1. *finite, non-empty sets X, W, and*
2. *a binary relation $R \subset X \times W$.*

We write $\Lambda[X, W, R]$ to indicate that the instance Λ specifies X, W and R as above.

An ISP also provides several algorithms. For this purpose, we require that the instance descriptions, as well as the elements of the sets X and W, can be uniquely encoded as bit strings of length polynomial in λ, and that both X and $(I_\lambda)_{\lambda \in \mathbb{N}}$ have polynomial-time samplers. The ISP algorithms are as follows, all parameterized by $\Lambda[X, W, R]$:

– *Verification* algorithm $V_\Lambda(x, w)$: A deterministic algorithm that takes as input a problem statement x and a witness w and outputs 1 if $(x, w) \in R$ and 0 otherwise.
– *Successor* algorithm $S_\Lambda(x, w)$: A deterministic algorithm that takes as input a problem statement[8] x and a valid witness w and outputs a new instance $x' \in X$.
– *Solving* algorithm $M_\Lambda(x, k)$: A probabilistic algorithm that takes as input a problem statement x and a number $k \in \mathbb{N}^+$ and outputs a sequence of k witnesses $(w_i)_{i \in [k]}$.

In the sequel, we will omit writing Λ as a parameter of V, S, M when it is clear from the context. In order to ease the presentation, we recursively extend the definitions of S and R to sequences of witnesses as follows: Let $S(x, \emptyset) := x$ and for any $k > 1$, $S(x, (w_i)_{i \in [k]}) := S(S(x, (w_i)_{i \in [k-1]}), w_k)$ and $(x, (w_i)_{i \in [k]}) \in R$ iff $\bigwedge_{i=1}^{k} (S(x, (w_j)_{j \in [i-1]}), w_i) \in R$. Further, we assume that M is *correct*, i.e., for $(w_i)_{i \in [k]} \leftarrow M(x, k)$, it holds that $(x, (w_i)_{i \in [k]}) \in R$.

Example. Next, we present as an example Bitcoin's underlying computational problem captured as an ISP.

Construction 1. *Let T be a protocol parameter representing how hard it is to solve a problem instance.*[9] *Then:*

– *I_λ is the uniform distribution over functions $H : \{0,1\}^* \to \{0,1\}^\lambda$ in some family of hash functions $\mathcal{H}$, i.e., $\Lambda = \{H\}$;*
– *$X = \{x | x < T \wedge x \in \{0,1\}^\lambda\}$ and $W = \{0,1\}^* \times \{0,1\}^\lambda$;*
– *$R = \{(x, w) | H(H(x||m)||ctr) < T$, for $w = m||ctr\}$;*
– *$V(x, w)$ checks whether $H(H(x||m)||ctr) < T$, for $w = m||ctr$;*

[8] We could formalize S more generally, to take as input a sequence of problem statements. However, for our exposition the current formulation suffices. Note, that a more general definition would be needed for the variable difficulty case [33], which we do not study here, where the next block's difficulty depends on the whole chain.

[9] For simplicity, in our exposition the hardness parameter for each ISP is fixed, and we do not capture it explicitly.

- *$S(x, w) = H(H(x||m)||ctr)$, and*
- *$M(x, 1)$ tests whether $V(x, (m, ctr))$ is true, for different (m, ctr) pairs, until it finds a solution. $M(x, k)$ is defined inductively, by running successively $M(x, k-1)$ and $M(x', 1)$, for $x' := S(x, M(x, k-1))$. The output consists of the witnesses output by the two programs.*

4.2 Technical Overview

Next, we give a complete overview of the technical results of this section regarding the implementation of a transaction ledger based on non-idealized hash functions.

First, we describe our ISP-based protocol in Sect. 4.3. The main challenge to overcome is that while the protocol's security is going to be based on iterated hardness (Definition 6), it operates in a setting where the adversary can also take advantage of the work of honest parties. This includes the adversary being able to see the information leaked by the honestly produced blocks, as well as honest parties directly working on the chain it is extending. In contrast, the iterated hardness experiment does not provide any guarantees about these cases, as the adversary does not receive any externally computed witnesses.

Towards this end, blocks in our protocol, instead of exposing the relevant computed witness, contain a proof of knowledge (PoK) of such a valid witness through a non-interactive zero-knowledge (NIZK) proof. At first, the fact that we use NIZK proofs for a language that is moderately hard may seem counterintuitive, due to the fact that a trivial simulator and extractor would exist for the zero-knowledge and soundness properties, since computing a new witness for a given statement takes polynomial time. Instead, following our general approach, we make concrete assumptions regarding the efficiency of both the simulator and the extractor. Informally, we require that the time it takes to simulate a proof or extract a witness is a lot smaller than the time it takes for honest parties to compute a witness (see Assumption 2). Note that in practice this can be achieved by making the underlying problem hard enough, which on the flip side will affect the performance of the resulting ledger being implemented.

Regarding chain selection, we adopt the longest-chain rule of the Bitcoin protocol. As we will see later, this will allow our protocol to operate even if the witnesses of the ISP are malleable. To make our analysis cleaner, the hash-chain structure of blocks is decoupled from the underlying computational problem.

As an intermediate step, in Sect. 4.4, we present a set of ISP properties sufficient to prove our protocol secure. First, an ISP is *iteratively hard* iff the ISP solving algorithm takes t steps to solve k instances iteratively, and no alternative algorithm can substantially speed up this process and produce k iterative solutions with non-negligible probability. Next, an ISP is (t, α)-*successful* when the number of steps of the solving algorithm is below t with probability at least α. The ISP is *next-problem simulatable* if the output of the successor algorithm applied on a witness w corresponding to an instance x can be simulated independently of x and the same is the case for the running time of the solver. Finally, an ISP is *witness-malleable* if, given a witness w for a problem instance x, it is possible to sample an alternative witness whose resulting distribution via the successor algorithm is computationally indistinguishable with the output of the successor over a random witness produced by the solving algorithm.

Armed with the above definitions we prove in Sect. 4.5 that our novel blockchain protocol implements a transaction ledger. We note that the main technical difficulty of our blockchain security proof is to construct a reduction that breaks the underlying iterated hardness assumption given a common-prefix attack to the blockchain protocol. The reduction takes advantage of the fact that the ISP is witness malleable and next-problem simulatable to cheaply simulate honest parties' work, as well as amenable to zero-knowledge proof simulation and extraction to extract a sequence of iterated witnesses despite the fact that the attacker may not produce consecutive blocks. After some more work, we are able to prove the following theorem:

Theorem 3 (Informal). *Assume the existence of collision-resistant hash functions, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK) and a secure ISP problem with appropriate parameters, and that $t_{\mathcal{A}}$ is (roughly) less than half the total running time of honest parties per round. Then there exists a protocol that implements a robust public transaction ledger.*

Finally, in Sect. 4.6, we present a secure ISP problem assuming the existence of a hash function that satisfies both the computational extraction and iterated hardness properties presented in Sect. 3. The main characteristic of this new ISP (Construction 2) is that, similarly to the Bitcoin ISP (Construction 1), it uses a double hash, but, in contrast, it requires the inner hash value to be below the target threshold, as opposed to the outer value. In more detail, given a problem statement x and witnesses w_1, w_2, while the next problem is defined exactly as in Bitcoin, i.e., $H(H(x||w_1)||w_2)$, the witnesses are valid if $H(x||w_1) < T$ holds, compared to $H(H(x||w_1)||w_2) < T$. This swap allows the randomness of the outer hash witness to be freely selected by a uniform distribution. In turn, this gives us the ability to argue that (i) due to the randomness extraction property of the hash, the inner hash value is computationally indistinguishable from uniform and hence the solving run-time of the ISP can be simulated independently of the problem statement; (ii) again due to the randomness extraction property, the outer hash value is computationally indistinguishable from uniform, and (iii) witness malleability can be shown in a straightforward manner by choosing another witness for the outer hash at random. Moreover, regarding the hard-ISP property, we can take advantage of the iterative hardness of Bitcoin's ISP construction and the fact that Construction 2 is closely related to it. The main idea is that if there exists an attacker against our construction, then we can use it to break the iterative hardness property (Definition 6) by using the inner hash witnesses in Construction 2 as an outer hash witnesses in Construction 1. Putting everything together results in the following:

Lemmas 5 and 6 (Informal). *Assume the existence of a hash function family that is CRE and iteratively hard for appropriate parameters. Then, there exists a secure ISP problem.*

Finally, using the above results we are able to obtain Theorem 3.

4.3 Blockchain Protocol Description

Next, we are going to describe our new protocol. Our protocol, $\Pi_{\mathsf{PL}}^{\mathsf{new}}$, uses as building blocks three cryptographic primitives: An ISP $\mathcal{I} = (M, V, S)$, a collision-resistant hash function family $\mathcal{H}$, and a robust NIZK protocol $\Pi_{\mathsf{NIZK}} = (q, \mathsf{P}, \mathsf{V}, \mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2), \mathsf{E})$ for the language[10]

$$L = \{(\Lambda[X, W, R], x, x')|\exists w \in W : (x, w) \in R \wedge S(x, w) == x'\}$$

where $\Lambda[X, W, R]$ is an ISP instance of $\mathcal{I}$. Π_{NIZK} also supports labels, which we denote as a superscript on P and V. The initialization of these primitives happens through the CRS all parties share at the start of the execution, which contains: An instance description $\Lambda[X, W, R]$, a statement $x_{\mathtt{Gen}}$, the description of a hash function $H : \{0, 1\}^* \to \{0, 1\}^\lambda$ and the NIZK reference string Ω, each randomly sampled from $I_\lambda, X, \mathcal{H}, \{0, 1\}^{q(\lambda)}$, respectively. Moreover, as in [31], our protocol is parameterized by the chain validation predicate $\mathrm{V}(\cdot)$, the chain reading function $\mathrm{R}(\cdot)$, and the input contribution function $\mathrm{I}(\cdot)$ to capture higher-level applications, e.g., Bitcoin.

Next, we introduce some notation used in the description of our protocol. We use the terms block and chain to refer to tuples of the form $\langle s, m, x, \pi\rangle \in \{0, 1\}^\lambda \times \{0, 1\}^* \times X \times \{0, 1\}^{\mathsf{poly}(\lambda)}$, and sequences of such tuples, respectively. The rightmost (resp., leftmost) block of chain $\mathcal{C}$ is denoted by $\mathrm{head}(\mathcal{C})$ (resp., $\mathrm{tail}(\mathcal{C})$). Each block contains the hash of the previous block s, a message m, the next problem x to be solved, and a NIZK proof π. We denote by $B_{\mathtt{Gen}} = \langle 0^\lambda, 0^\lambda, x_{\mathtt{Gen}}, 0^\lambda\rangle$ a special block called the *genesis block*; note that $x_{\mathtt{Gen}}$ is part of the CRS. A chain $\mathcal{C} = (\langle s_i, m_i, x_i, \pi_i\rangle)_{i\in[k]}$ is valid if: (i) The first block of $\mathcal{C}$ is equal to $B_{\mathtt{Gen}}$; (ii) the contents of the chain $\mathbf{m}_{\mathcal{C}} = (m_1, \ldots, m_k)$ are valid according to the chain validation predicate V, i.e., $\mathrm{V}(\mathbf{m}_{\mathcal{C}})$ is true; (iii) $s_{i+1} = H(s_i||m_i||x_i||i)$[11] for all $i \in [k]$, and (iv) $\mathsf{V}^{s_{i+1}}((\Lambda, x_{i-1}, x_i), \pi_i)$ is true for all $i \in [k] \setminus \{1\}$; see Algorithm 1. We call $H(s_i||m_i||x_i||i)$ the *hash of block* B_i and denote it by $H(B_i)$, and define $H(\mathcal{C}) \triangleq H(\mathrm{head}(\mathcal{C}))$. We will consider two valid blocks or chains as equal, if all their parts match, except possibly for the NIZK proofs.

We proceed to describe the main function of the protocol, presented in Algorithm 4. At each round, each party chooses the longest valid chain among the ones it has received (Algorithm 2) and tries to extend it by computing a new witness. If it succeeds, it diffuses the new block to the network. In more detail, each party will run the solver M on the problem x defined in the last block $\langle s, m, x, \pi\rangle$ of the chosen chain $\mathcal{C}$. If it succeeds on finding a witness w, it will then compute a NIZK proof that it knows a witness w such that $(x, w) \in R$ and $S(x, w) = x'$, for some $x' \in X$. The proof should also have a label $H(H(\mathit{head}(\mathcal{C}))||m'||x'||(|\mathcal{C}|+1))$, where m' is the output of the input contribution function $\mathrm{I}(\cdot)$, i.e., the message encoded in the block; see Algorithm 3. Then, the party diffuses the extended

[10] We assume that both V and S are efficiently computable. Hence, $L \in \mathcal{NP}$.

[11] We include a fixed length (λ-bit) encoding of the height of the block in the hash on purpose. This way, the contents of the hash chain form a suffix-free code [9], which in turn implies collision resistance. See Lemma 1.

Algorithm 1. The validate procedure, parameterized by $B_{\texttt{Gen}}$, the hash function $H(\cdot)$, the *chain validation predicate* $V(\cdot)$, and the verification algorithm V of Π_{NIZK}. The input is $\mathcal{C}$.

```
1: function validate($\mathcal{C}$)
2:     $b \leftarrow \mathrm{V}(\mathbf{m}_{\mathcal{C}}) \wedge (\mathsf{tail}(\mathcal{C}) = B_{\texttt{Gen}})$        ▷ $\mathbf{m}_{\mathcal{C}}$ describes the contents of chain $\mathcal{C}$.
3:     if $b = \mathrm{True}$ then        ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
4:         $s' \leftarrow H(B_{\texttt{Gen}})$        ▷ Compute the hash of the genesis block.
5:         $x' \leftarrow x_{\texttt{Gen}}$
6:         $\mathcal{C} \leftarrow \mathcal{C}^{1\rceil}$        ▷ Remove the genesis from $\mathcal{C}$
7:         while $(\mathcal{C} \neq \epsilon \wedge b = \mathrm{True})$ do
8:             $\langle s, m, x, \pi \rangle \leftarrow \mathsf{tail}(\mathcal{C})$
9:             $s'' \leftarrow H(\mathsf{tail}(\mathcal{C}))$
10:            if $(s = s' \wedge \mathsf{V}^{s''}(\Omega, (\Lambda, x', x), \pi))$ then
11:                $s' \leftarrow s''$        ▷ Retain hash value
12:                $x' \leftarrow x$
13:                $\mathcal{C} \leftarrow \mathcal{C}^{1\rceil}$        ▷ Remove the tail from $\mathcal{C}$
14:            else
15:                $b \leftarrow \mathrm{False}$
16:    return $(b)$
```

chain to the network. Finally, if the party is queried by the environment, it outputs $\mathrm{R}(\mathcal{C})$, where $\mathcal{C}$ is the chain selected by the party; the chain reading function $\mathrm{R}(\cdot)$ interprets $\mathcal{C}$ differently depending on the higher-level application running on top of the Bitcoin backbone protocol. We assume that all honest parties take the same number of steps t_H per round.

Algorithm 2. The function that finds the "best" chain, parameterized by function $\max(\cdot)$. The input is $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$.

```
1: function maxvalid($\mathcal{C}_1, \ldots, \mathcal{C}_k$)
2:     $temp \leftarrow \varepsilon$
3:     for $i = 1$ to $k$ do
4:         if validate($\mathcal{C}_i$) then
5:             $temp \leftarrow \max(\mathcal{C}, temp)$
6:     return $temp$
```

In order to turn the above protocol into a protocol realizing a public transaction ledger, we define functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ exactly as in [31]. For completeness we give these definitions in Table 1. We denote the new public ledger protocol by $\Pi_{\mathsf{PL}}^{\mathsf{new}}$.

4.4 ISP Security Properties

Next, we present a set of ISP properties sufficient to prove our protocol secure. Later in Sect. 4.6 we show how to instantiate them.

Algorithm 3. The *proof of work* function is parameterized by the hash function $H(\cdot)$, and the proving algorithm P of Π_{NIZK}. The input is $(m', \mathcal{C})$.

```
function $\mathsf{pow}(m', \mathcal{C})$
    $\langle s, m, x, \pi \rangle \leftarrow \mathrm{head}(\mathcal{C})$
    $w \leftarrow M(x)$                                  ▷ Run the honest solving algorithm of the ISP.
    if $w \neq \bot$ then
        $x' \leftarrow S(x, w)$                          ▷ Compute the next problem to be solved.
        $s' \leftarrow H(s||m||x|||\mathcal{C}|)$        ▷ Compute the hash of the last block.
        $s'' \leftarrow H(s'||m'||x'|||\mathcal{C}| + 1)$  ▷ Compute the hash of the new block.
        $\pi' \leftarrow \mathsf{P}^{s''}(\Omega, (\Lambda, x, x'), w)$  ▷ Compute the NIZK proof.
        $B \leftarrow \langle s', m', x', \pi' \rangle$
    $\mathcal{C} \leftarrow \mathcal{C}B$                ▷ Extend chain
    return $\mathcal{C}$
```

Algorithm 4. The Bitcoin backbone protocol, parameterized by the *input contribution function* $\mathrm{I}(\cdot)$ and the *chain reading function* $\mathrm{R}(\cdot)$.

```
$\mathcal{C} \leftarrow B_{\mathsf{Gen}}$                ▷ Initialize $\mathcal{C}$ to the genesis block.
$st \leftarrow \varepsilon$
$round \leftarrow 0$
while TRUE do
    $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}$, any chain $\mathcal{C}'$ found in RECEIVE())
    $\langle st, m \rangle \leftarrow \mathrm{I}(st, \tilde{\mathcal{C}}, round, \text{INPUT}(), \text{RECEIVE}())$   ▷ Determine the $m$-value.
    $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(m, \tilde{\mathcal{C}})$
    if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ then
        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
        DIFFUSE($\mathcal{C}$)
    $round \leftarrow round + 1$
    if INPUT() contains READ then
        write $\mathrm{R}(\mathbf{m}_{\mathcal{C}})$ to OUTPUT()
```

In the same spirit as in Boneh *et al.* [14]'s definition of an iterated sequential function, we can define the notion of a *hard* iterated search problem. Our definition is parameterized by t, δ and k_0, all functions of λ which we omit for brevity. Unlike the former definition, we take in account the *total* number of steps instead of only the sequential ones, and we require the error probability to be negligible after at least k_0 witnesses have been found instead of one. In that sense, our notion relaxes the strict convergence criterion of [14]. Finally, note that the adversary is allowed some precomputation time.

Definition 8. *An ISP $\mathcal{I} = (V, M, S)$ is (t, δ, k_0)-*hard *iff it holds that*

– *For $\lambda \in \mathbb{N}$ and for all polynomially large $k \geq k_0$:*

$$\Pr_{\substack{\Lambda[X,W,R] \leftarrow I_\lambda;\\ x \leftarrow X}}\left[\begin{array}{l}(w_i)_{i \in [k]} \leftarrow M(x, k) : (x, (w_i)_i) \in R \\ \wedge\, \mathsf{Steps}_M(x, k) \leq k \cdot t\end{array}\right] \geq 1 - \mathsf{negl}(\lambda), \quad \textit{and}$$

Table 1. *The instantiation of functions* $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ *for protocol* $\Pi_{\mathsf{PL}}^{\mathsf{new}}(\mathcal{I})$.

Content validation pre-dicate $\mathrm{V}(\cdot)$	$\mathrm{V}(\cdot)$ is true if its input $\langle m_1, \ldots, m_\ell \rangle$ is a valid ledger, i.e., it is in $\mathcal{L}$
Chain reading function $\mathrm{R}(\cdot)$	$\mathrm{R}(\cdot)$ returns the contents of the chain if they constitute a valid ledger, otherwise it is undefined
Input contribution function $\mathrm{I}(\cdot)$	$\mathrm{I}(\cdot)$ returns the largest subsequence of transactions in the input and receive registers that constitute a valid ledger, with respect to the contents of the chain $\mathcal{C}$ the party already has, preceded by a neutral random transaction

– *for any PPT RAM* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\lambda \in \mathbb{N}$, *and all polynomially large* $k \geq k_0$, *it holds that*

$$\Pr_{\substack{\Lambda[X,W,R] \leftarrow I_\lambda;\\ x \leftarrow X}} \left[\begin{array}{l} st \leftarrow \mathcal{A}_1(1^\lambda, \Lambda); (w_i)_{i \in [k]} \leftarrow \mathcal{A}_2(1^\lambda, st, x) : \\ (x, (w_i)_i) \in R \wedge Steps_{\mathcal{A}_2}(st, x) < (1-\delta)k \cdot t \end{array} \right] \leq \mathsf{negl}(\lambda).$$

The next property, has to do with establishing an upper bound t on the the running time of the verification algorithm V. Intuitively, the product $\theta \cdot t$ should be a lot smaller than the number of steps $t_{\mathcal{H}}$ per round available to honest parties, to avoid resource depletion attacks.

Definition 9. *An ISP* $\mathcal{I} = (V, M, S)$ *is* t-verifiable *iff algorithm* V *takes time at most* t *(on all inputs).*

In general, attacking an honest solver amounts to finding a certain set of inputs over which the honest solving algorithm fails to produce witnesses sufficiently fast. In order to combat this attack, we introduce the following property: We say that an ISP $\mathcal{I}$ is (t, α)-*successful* when the probability that M[12] computes a witness in t steps is at least α.

Definition 10. *An ISP* $\mathcal{I} = (V, M, S)$ *is* (t, α)-successful *iff for* $\lambda \in \mathbb{N}$, $\Lambda[X, W, R] \in I_\lambda$, *and for all* $x \in X$ *it holds that:* $\Pr[\mathsf{Steps}_M(x) < t] \geq \alpha$.

The iterated hardness property (Definition 8) does not give any guarantees regarding composition. For blockchain protocols, however, this is necessary as many parties concurrently try to solve the same ISP. To address this issue, we introduce the next property that ensures that learning how long it took for a witness to be computed or what the next problem defined by such witness is, does not leak any information that could help the adversary find a witness himself. More formally, there exists an efficient simulator whose output is computationally indistinguishable from the distribution of the time it takes to compute a witness w for some statement x and the next statement $S(x, w)$. Note that, crucially, the simulator does not depend on the instance description Λ or the problem statement x, and that we consider a non-uniform distinguisher.

[12] For brevity, we use $M(x)$ instead of $M(x, 1)$ in this section.

Definition 11. *An ISP $\mathcal{I} = (V, M, S)$ is t-*next-problem simulatable *iff there exists a t-bounded RAM Ψ such that for any PPT RAM D, any $\lambda \in \mathbb{N}$, any $z \in \{0,1\}^{poly(\lambda)}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $x \in X$, it holds that*

$$|\Pr[D(1^\lambda, z, \Lambda, x, (S(x, M(x)), \mathsf{Steps}_M(x))) = 1] - \Pr[D(1^\lambda, z, \Lambda, x, \Psi(1^\lambda)) = 1]| \leq \mathsf{negl}(\lambda).$$

The next property has to do with a party's ability to "cheaply" compute witnesses for a statement, if it already knows one. This will be important to ensure that even if the adversary has external help to produce some of the witnesses needed by the hard ISP experiment, as is the case for blockchain protocols, still the overall process remains hard with respect to the number of consecutive blocks the adversary actually produced. We call this ISP property *witness malleability.*

Definition 12. *An ISP $\mathcal{I} = (V, M, S)$ is t-*witness malleable *iff there exists a t-bounded RAM Φ such that for any PPT RAM D, any $\lambda \in \mathbb{N}$, any $z \in \{0,1\}^{poly(\lambda)}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $(x, w) \in R$, it holds that $(x, \Phi(x, w)) \in R$, and*

$$|\Pr[D(1^\lambda, z, \Lambda, x, w, S(x, \Phi(x, w))) = 1] - \Pr[D(1^\lambda, z, \Lambda, x, w, S(x, M(x))) = 1]| \leq \mathsf{negl}(\lambda).$$

Finally, we call an ISP that satisfies all the above properties *secure.*

Definition 13. *An ISP $\mathcal{I} = (V, M, S)$ is $(t_{\mathsf{ver}}, t_{\mathsf{succ}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-*secure *iff it is t_{ver}-verifiable, $(t_{\mathsf{succ}}, \alpha)$-successful, t_{nps}-next-problem simulatable, t_{mal}-witness malleable, and $(t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-hard.*

An ISP scheme with trivial parameters is of limited use in a distributed environment; for example, if $\delta_{\mathsf{hard}} \ll 1$ or $t_{\mathsf{hard}} \ll t_{\mathsf{ver}}$. Hence, next we describe the parameters' ranges that make for a non-trivial secure ISP. First off, and ignoring negligible terms, one can show that $\alpha \leq \frac{t_{\mathsf{succ}}}{(1-\delta_{\mathsf{hard}})t_{\mathsf{hard}}}$ (see Lemma 4). On the other hand, the successful property always holds for $\alpha = 0$. Therefore, for a non-trivial ISP scheme it should hold that α is close to $\frac{t_{\mathsf{succ}}}{(1-\delta_{\mathsf{hard}})t_{\mathsf{hard}}}$. To avoid denial of service attacks, $\theta \cdot t_{\mathsf{ver}}$ must be sufficiently small compared to t_{hard}, the running time of the solving algorithm M. Furthermore, t_{mal} should be a lot smaller than t_{hard}, otherwise M can be used as a trivial simulator. We note, that the security of the protocol that we presented earlier relies on the fact that a secure ISP scheme with favorable parameters exists, mainly reflected in Assumption 2 (Sect.4.5).

4.5 Security of the ISP-based Blockchain Protocol

In this subsection we prove that $\Pi^{\mathsf{new}}_{\mathsf{PL}}$ implements a robust public transaction ledger (cf. Definition 3), assuming the underlying ISP $\mathcal{I}$ is secure.

Security Proof of the ISP-based Protocol. We proceed to the main part of the protocol analysis. The first assumption we are going to make is that the underlying ISP $\mathcal{I}$ is secure, and that the runtimes of the procedures of the NIZK system are upper bounded.

Table 2. *The parameters in our analysis: $\lambda, n, t, t_{\mathcal{H}}, t_{\mathcal{A}}, t'_{\mathcal{H}}, t'_{\mathcal{A}}, \theta, k_{hard}$ are in $\mathbb{N}$, $\alpha, f, \gamma, \beta, \delta, \delta_{\mathsf{Steps}}, \delta_{\mathsf{ISP}}$ are in $\mathbb{R}$.*

λ :	security parameter
n :	number of parties
t :	number of parties corrupted
$t_{\mathcal{H}}$:	number of steps per round per honest party
$t_{\mathcal{A}}$:	total number of adversarial steps per round
$t'_{\mathcal{H}}$:	lower bound on number of steps running M per round per honest party
$t'_{\mathcal{A}}$:	round simulation cost, excluding honest calls to M
θ :	upper bound on the number of messages sent by the adversary per round
β :	upper bound on the rate at which the adversary computes witnesses per step
α :	probability that M outputs a witness after $t'_{\mathcal{H}}$ steps
f :	probability that at least one party computes a block in a round
γ :	probability that exactly one party computes a block in a round
δ :	upper bound on the total block generation rate
δ_{Steps} :	honest parties' advantage on number of steps
δ_{ISP} :	adversary's advantage on ISP witnesses computation rate
k_{hard} :	convergence parameter of ISP hardness

Assumption 1 (ISP Assumption). *For parameters $t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}$, $\delta_{\mathsf{hard}}, k_{\mathsf{hard}}, t_{\mathsf{P}}$, $t_{\mathsf{V}}, t_{\mathsf{S}}$, and t_{E} we assume that:*

– *ISP $\mathcal{I}$ is $(t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-secure;*[13]
– *running the prover (resp., verifier, simulator, extractor) of Π_{NIZK} takes t_{P} (resp. $t_{\mathsf{V}}, t_{\mathsf{S}}, t_{\mathsf{E}}$) steps.*

Next, we introduce some additional notation necessary to formalize our second assumption that has to do with the computational power of the honest parties and the adversary. For brevity, and to better connect our analysis to previous work [31,36,49], we denote by $\beta = ((1-\delta_{\mathsf{hard}})\cdot t_{\mathsf{hard}})^{-1}$, the upper bound on the rate at which the adversary can compute witnesses in the iterated hardness game. We introduce two variables, $t'_{\mathcal{H}}$ and $t'_{\mathcal{A}}$, that have to do with the effectiveness of honest parties and the adversary in producing witnesses for $\mathcal{I}$. $t'_{\mathcal{H}}$ is a lower bound on the number of steps each honest party takes per round running M. It holds that in any round at least $n-t$ parties will run M for at least $t'_{\mathcal{H}}$ steps. $t'_{\mathcal{A}}$ denotes the maximum time needed by a RAM machine to simulate the adversary, the environment and the honest parties in one round of the protocol execution, without taking into account calls made to M by the latter, and with the addition of one invocation of the NIZK extractor. They amount to:

$$t'_{\mathcal{A}} = t_{\mathcal{A}} + \theta \cdot t_{\mathsf{V}} + t_{\mathsf{E}} + n(t_{\mathsf{bb}} + t_{\mathsf{nps}} + t_{\mathsf{mal}} + t_{\mathsf{S}}) \quad \text{and} \quad t'_{\mathcal{H}} = t_{\mathcal{H}} - t_{\mathsf{bb}} - \theta t_{\mathsf{V}} - t_{\mathsf{P}},$$

where t_{bb} (bb for backbone) is an upper bound on the number of steps needed to run the code of an honest party in one round besides the calls to $M, \mathsf{P}, \mathsf{V}$ (Table 2).

[13] $t'_{\mathcal{H}}$ is related to our model and we formally define it in the next paragraph.

We are now ready to state our main computational assumption regarding the honest parties and the adversary. Besides assuming that the total number of steps the honest parties take per round exceed those of the adversary, and that the total block generation rate is bounded, we have to additionally assume that the efficiency of the solving algorithm M used by honest parties is comparable to that of the adversary; i.e, as explained earlier, α should be comparable to $\beta t'_{\mathcal{H}}$, otherwise the adversary will be able to compute long chains of blocks fast and break the security of the protocol. The observation we just made, corresponds to the first condition in our formalization, which we present next. To avoid confusion, we cast most of our analysis based on the δ parameter. Furthermore, note that under optimal conditions – i.e., δ_{ISP} close to 0 and $t_{\mathsf{P}}, t_{\mathsf{V}}, t_{\mathsf{E}}, t_{\mathsf{S}}, t_{\mathsf{nps}}, t_{\mathsf{mal}}$ a lot smaller than $t_{\mathcal{H}}$ – our assumption allows for an adversary that controls up to 1/3 of the total computational power available (vs. 1/2 in the RO model).

Assumption 2. *There exist $\delta_{\mathsf{ISP}}, \delta_{\mathsf{Steps}}$ and $\delta \in (0,1)$, such that for sufficiently large $\lambda \in \mathbb{N}$:*

- $\alpha \geq (1-\delta_{\mathsf{ISP}})\beta t'_{\mathcal{H}} > \mathsf{negl}(\lambda)$ *(ISP generation gap)*
- $(n-t)t'_{\mathcal{H}}(1-\delta_{\mathsf{Steps}}) \geq 2 \cdot t'_{\mathcal{A}}$ *(steps gap)*
- $\frac{\delta_{\mathsf{Steps}}-\delta_{\mathsf{ISP}}}{2} \geq \delta > \beta(t'_{\mathcal{A}} + nt_{\mathcal{H}})$ *(bounded block generation rate)*

Next, we focus on structural properties of blockchains in our protocol. We follow a similar approach to [36] based on a collisions resistant hash function. Observe that the hash structure of any blockchain in our protocol is similar to the Merkle-Damgard transform [24], defined as:

$$\mathrm{MD}(IV, (x_i)_{i\in[m]}) : z = IV;\ \text{for } i = 1 \text{ to } m \text{ do } z = H(z||x_i);\ \text{return } z,$$

where H is the hash function described in the CRS, and IV is set to $B_{\mathtt{Gen}}$. Based on this observation, as in [36], we can show that no efficient adversary can find distinct chains with the same hash value, as this would result to finding a collision on the underlying hash function. Due to space limitations we point to the full version of the paper for the proof.

Lemma 1. *Let $\mathcal{H}$ be a collision-resistant hash function family. The probability that any PPT RAM $\mathcal{A}$, given $B_{\mathtt{Gen}}$, can find two distinct valid chains $\mathcal{C}_1, \mathcal{C}_2$ such that $H(\mathcal{C}_1) = H(\mathcal{C}_2)$, is negligible in λ.*

Lemma 1 implies that insertion and copy properties[14] of [31], that have to do with the way blocks are connected, do not occur with overwhelming probability in λ.

Definition 14. *An* insertion *occurs when, given a chain $\mathcal{C}$ with two consecutive blocks B and B_0, a block B^* created after B_0 is such that B, B^*, B_0 form three consecutive blocks of a valid chain. A* copy *occurs if the same block exists in two different positions.*

[14] A third property, called "prediction," also introduced in [31], is not needed in our proof as it is captured by the fact that the ISP is hard even in the presence of adversarial precomputation.

Corollary 1. *Let $\mathcal{H}$ be a collision-resistant hash function family. Then, for any PPT $\mathcal{A}, \mathcal{Z}$ no insertions or copies occur in $\text{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$ with probability* $1 - \mathsf{negl}(\lambda)$.

We proceed to the main part of the analysis. First, we introduce some useful notation. For each round j, we define the Boolean random variables X_j and Y_j as follows. Let $X_j = 1$ if and only if j was a *successful round*, i.e., at least one honest party computed a witness at round j, and let $Y_j = 1$ if and only if j was a *uniquely successful round*, i.e., *exactly one honest party* computed a witness at round j. With respect to a set of rounds R, let $X(R) = \sum_{j \in R} X_j$ and define $Y(R)$ similarly.

Moreover, with respect to some block B computed by an honest party P at some round r, let $Z^P_r(R)$ denote the maximum number of distinct blocks diffused by the adversary during R that have B as their ancestor and lie on the same chain; note that honest parties compute at most one block per round. If P is corrupted or did not compute any block at r, let $Z^P_r(R) = 0$. We extend the definition of random variable $X(R)$ to $X^P_r(R)$ similarly.

An important part of our analysis will be to establish lower and upper bounds for these random variables. First, in Lemma 3 we will show that the rate at which the adversary produces witnesses is upper bounded by $\beta \cdot t'_{\mathcal{A}}$. Then, in Lemma 4 we prove that the expected rate of successful and uniquely successful rounds is lower bounded by f and γ, respectively, both defined below:

$$f = 1 - (1-\alpha)^{n-t} \text{ and } \gamma = (n-t) \cdot \alpha \cdot (1 - \beta t_{\mathcal{H}})^{n-1}$$

Finally, for our analysis to go through, γ should be twice as big as $\beta \cdot t'_{\mathcal{A}}$. As we demonstrate next, this follows from the fact that in Assumption 2 the honest parties take at least double the steps the adversary takes per round.

Lemma 2. *Assume an ISP that complies with Assumptions 1 and 2. It holds that $\gamma \geq 2(1+\delta)\beta t'_{\mathcal{A}}$.*

Proof. For γ it holds that:

$$\begin{aligned}\gamma =& (n-t) \cdot \alpha \cdot (1-\beta t_{\mathcal{H}})^{n-1} \geq (n-t) \cdot \alpha \cdot (1 - \beta t_{\mathcal{H}} n) \\ \geq& (n-t) \cdot (1-\delta_{\mathsf{ISP}}) \cdot \beta t'_{\mathcal{H}} \cdot (1-\delta) \geq \frac{(1-\delta_{\mathsf{ISP}})(1-\delta)}{(1-\delta_{\mathsf{Steps}})} \cdot 2 \cdot \beta t'_{\mathcal{A}} \geq 2(1+\delta)\beta t'_{\mathcal{A}}\end{aligned}$$

where we have first used Bernouli's inequality, and then the three conditions from Assumption 2. The last inequality follows from the fact that $\frac{\delta_{\mathsf{Steps}} - \delta_{\mathsf{ISP}}}{2} \geq \delta$. □

As promised, we prove next that the adversary cannot mine blocks extending a single chain, with rate and probability better than that of breaking the iterative hardness property. Due to space limitations we only give a proof sketch, and point to the full version of the paper for the proof.

Lemma 3. *For any set of consecutive rounds R, where $|R| \geq k_{\mathsf{hard}}/\beta t'_{\mathcal{A}}$, for any party P, and any round $i \in R$, the probability that $Z^P_i(R) \geq \beta t'_{\mathcal{A}}|R|$ is* $\mathsf{negl}(\lambda)$.

(Proof Sketch.). W.l.o.g., let i be the first round of $R = \{i' | i \leq i' < i + s\}$, and let E be the event where in $\textsc{view}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$ party P at round i mined a block B, and the adversary mined at least $\beta t'_{\mathcal{A}} s$ blocks until round $i+s$ that extend B and are part of a single chain. For the sake of contradiction, assume that the lemma does not hold, and thus $\Pr[E]$ is non-negligible. Using $\mathcal{A}$, we will construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that breaks the iterative hardness (Definition 8) of $\mathcal{I}$ with non-negligible probability.

$\mathcal{A}'$ is going to run internally $\mathcal{A}$ and $\mathcal{Z}$, while at the same time simulating the work honest parties do using the NIZK proof simulator. Moreover, $\mathcal{A}'$ is also going to use the witness malleability property, to trick $\mathcal{A}$ to produce blocks in a sequence, instead of interleaved adversarial and (simulated) honest blocks. Finally, using the NIZK extractor, $\mathcal{A}'$ is going to extract the witnesses from the adversarial blocks, and win the iterative hardness game. By a hybrid argument, we can show that the view of $\mathcal{A}, \mathcal{Z}$ is indistinguishable both in the real and the simulated run, and thus the probability that E happens will be the same in both cases, i.e., non-negligible.

We can do exactly the same reduction without simulating honest parties' work. Then, the total running time of the second stage of $\mathcal{A}'$ is $s \cdot (t'_{\mathcal{A}} + nt'_{\mathcal{H}})$-bounded. Hence, we can derive the following bound on the longest chain that can be produced by both honest and malicious parties during a certain number of rounds.

Corollary 2. *For any set of consecutive rounds R, where $|R| \geq k_{\mathsf{hard}}/\beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}})$, for any party P, and any round $i \in R$, the probability that $Z^P_i(R) + X^P_i(R) \geq \beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}}) \cdot |R|$ is $\mathsf{negl}(\lambda)$.*

Next, we prove lower bounds on the rate of successful and uniquely successful rounds. In our proof we are going to take advantage of the next-problem simulatable property of $\mathcal{I}$ and the zero-knowledge property of the robust NIZK we are using. The main idea is to first use these two properties and similar arguments as in Lemma 3 to construct an "ideal" execution where: (i) honest parties' behavior is efficiently simulated using Ψ, and (ii) is computationally indistinguishable from the "real" execution. Then, since the outputs of different invocations of the runtime simulator $\Psi(1^\lambda)$ are independent, it will be much easier to establish lower bounds for $X(\cdot)$ and $Y(\cdot)$ in the ideal execution. Finally, due to the fact that the two executions are computationally indistinguishable, and the execution properties we examine can be efficiently checked, it will follow that the same bounds should also hold for the real execution with negligible difference in probability. Due to space limitations we point to the full version of the paper for the proof.

Lemma 4. *For any set of consecutive rounds R, with $|R| \geq \lambda/\gamma\delta^2$, the following two events occur with negligible probability in λ:*

- *the number of uniquely successful rounds in R is less or equal to $(1-\frac{\delta}{4})\gamma \cdot |R|$;*
- *the number of successful rounds in R is less or equal to $(1-\frac{\delta}{4})f \cdot |R|$.*

Following the strategy of [31], we are now ready to define the set of *typical executions* for this setting.

Definition 15 (Typical execution). *An execution is* typical *if and only if* $\lambda \geq 9/\delta$ *and for any set* R *of consecutive rounds with* $|R| \geq \max\{4k_{\mathsf{hard}}, \lambda\}/\gamma\delta^2$*, the following hold:*

1. $Y(R) > (1 - \frac{\delta}{4})\gamma|R|$ *and* $X(R) > (1 - \frac{\delta}{4})f|R|$*;*
2. *for any party* P*, any round* $i \in R$*:* $Z_i^P(R) < \frac{\gamma}{2(1+\delta)} \cdot |R|$ *and* $Z_i^P(R) + X_i^P(R) < \beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}}) \cdot |R|$ *; and*
3. *no insertions and no copies occurred.*

Theorem 1. *An execution is typical with probability* $1 - \mathsf{negl}(\lambda)$*.*

Having established that typical rounds happen with overwhelming probability, the rest of the proof follows closely that of [31]. The only difference is that to prove the corresponding common-prefix lemma, although we can match blocks mined in uniquely successful rounds to adversarial blocks in one of the two chains that constitute the fork, the typicality of the execution only provides a bound on the maximum number of blocks in a single chain. Hence, only half of the blocks matched must outnumber the uniquely successful rounds in this interval, which is also the reason why our proof only works with an adversary controlling up to 1/3 of the parties. Due to space limitations we point to the full version of the paper for the details.

Next, we state our theorem. Note that both Consistency and Liveness depend on the convergence parameter k_{hard} of $\mathcal{I}$.

Theorem 2. *Assuming the existence of a collision-resistant hash function family, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and a secure ISP problem* $\mathcal{I}$ *that comply with Assumptions 1 and 2, protocol* $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ *implements a robust public transaction ledger with parameters* $k = \max\{4k_{\mathsf{hard}}, \lambda\}/\gamma\delta$ *and* $u = 2k/(1 - \frac{\delta}{4})f$*, except with negligible probability in* λ*.*

4.6 Realizing ISPs from Non-idealized Hash Functions

Next, we present a secure ISP problem assuming the existence of a hash function that satisfies both the computational extraction and iterated hardness properties presented in Sect. 3.

Construction 2. *Let* $\mathcal{H}$ *be a hash function family as in Definitions 5 and 6. Let* $T \in \{0,1\}^\lambda$ *be a hardness parameter. An instance of a secure ISP is as follows:*

- I_λ *is the uniform distribution over* $K(\lambda)$*, i.e.,* $\Lambda = \{k\}$*;*
- $X = \{0,1\}^\lambda, W = \{0,1\}^{2(d-1)\lambda}$*;*
- $R = \{(x, w) | H_k(x||w_1) < T$ *for* $w = w_1||w_2\}$*;*
- $M(x, 1)$ *iteratively samples* w_1 *from* $\mathcal{U}_{(d-1)\lambda}$*, and tests whether* $H_k(x||w_1) < T$*, until it finds a solution. It then samples a uniformly random* w_2 *from* $\mathcal{U}_{(d-1)\lambda}$*, and outputs* $w_1||w_2$*.*
- $S(x, w) = H_k(H_k(x||w_1)||w_2)$*, for* $w = w_1||w_2$*.*

Construction 2 is similar to Bitcoin's ISP construction (see Sect. 4.1, Construction 1), with the following differences:

1. In our construction $H_k(x||w_1)$ is required to be smaller than the hardness parameter T, while in Bitcoin $H_k(H_k(x||w_1)||w_2)$ is expected to be small, where w_1 is the hash of some message. This change allows a party who already knows a witness (w_1, w_2) for some statement, to produce a new one by changing w_2 arbitrarily.
2. Each time M tests a new possible witness, w_1 is sampled randomly, instead of just being increased by one, as in Bitcoin. This will help us later on to argue that each test succeeds with probability proportional to T.

Obviously, if used in "native" Bitcoin this construction is totally insecure, as by the time an honest party publishes a block, anyone can compute another valid block with minimal effort. However, it is good enough for our new protocol, where the witnesses are not exposed, and thus only a party who knows a witness can generate new witnesses for free. Next, we argue the security of the construction.

Assuming $\mathcal{H}$ is a computational randomness extractor is sufficient for the security properties that make up a secure ISP, besides hardness, to be satisfied. First, the fact that $H_k(x||w_1)$ is computationally indistinguishable from uniform, for any $x \in X$, implies that the runtime and the output of M are computationally indistinguishable from a process that sampled repeatedly a uniform value from $\{0,1\}^\lambda$ until it finds one that is smaller than T. This in turn implies that the runtime distribution of M is indistinguishable from the geometric distribution with parameter $T/2^\lambda$, and thus the successful ISP property is satisfied. Further, since w_2 is also chosen uniformly at random, we can show that a simulator that samples a random value from $\mathcal{U}_\lambda$ and the geometric distribution, satisfies the next-problem simulatability property. Finally, by resampling a new w_2 uniformly at random, an admissible witness is produced, and the witness malleability property follows. Thus, we are able to state the following lemma. Due to space limitations we point to the full version of the paper for the proof.

Lemma 5. *If $\mathcal{H}$ is a CRE hash family (Definition 5), then Construction 2 is $O(\lambda)$-next-problem simulatable, $O(\lambda)$-witness malleable, and $(t, \mathcal{C}_{T/2^\lambda}(O(t)))$-successful for any $t \in poly(\lambda)$, where $\mathcal{C}_{T/2^\lambda}$ is the cumulative geometric distribution with parameter $T/2^\lambda$.*

Regarding the hard-ISP property, we are going to take advantage of the iterative hardness of Bitcoin's ISP construction and the fact that Construction 2 is closely related to it. The main idea is that if there exists an attacker against our construction, then we can use it to break the iterative hardness property (Definition 6). In more detail, given as input a statement x, the iterated hardness attacker runs the attacker of our construction with input $H(x||w)$, where w is sampled at random. It is easy to see that if $((w_1, w'_1), \ldots, (w_m, w'_m))$ are the witnesses it is going to produce, then $((w, w_1), (w'_1, w_2), \ldots, (w'_{m-1}, w_m))$ are valid witnesses for Construction 1, and also against the iterative hardness property. The following lemma highlights this relation. Due to space limitations we point to the full version of the paper for the proof.

Lemma 6. *Assume Construction 2 is based on a hash family $\mathcal{H}$ that is CRE and t-iteratively hard. Then, for some polynomial $k_0(\cdot)$, any $\sigma \in (0,1)$ and $t' = \frac{2^\lambda}{(1-\sigma)T}$, Construction 2 is $(t', 1-t'/t, k_0)$-hard.*

Due to Theorem 2 and the previous two lemmas, we can implement a ledger assuming the existence of a robust NIZK, a hash family that is collision-resistant, another hash function family that is both CRE and iteratively hard for appropriate parameters, and that the adversary controls less than a third of the total computational power. The following theorem holds.[15] Due to space limitations we point to the full version of the paper for the proof.

Theorem 3. *Assuming the existence of collision-resistant hash functions, a hash function family that is CRE and t_{hard}-iteratively hard, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and that for some $\delta_{\mathsf{Steps}} \in (0,1)$, sufficiently large $\lambda \in \mathbb{N}$, and T equal to $\lfloor 2^\lambda \cdot \min\{\frac{\ln((1-\delta^2_{\mathsf{Steps}}/4)^{-1})}{t'_{\mathcal{H}}}, \frac{\delta_{\mathsf{Steps}}/4}{(t'_{\mathcal{A}}+nt'_{\mathcal{H}})(1+\delta_{\mathsf{Steps}}/2)}\}\rfloor$ it holds that :*

- $t_{\mathsf{hard}} \geq (1+\delta_{\mathsf{Steps}}/2)^{-1} \cdot \frac{2^\lambda}{T}$; *and*
- $2 \cdot t'_{\mathcal{A}} \leq (1-\delta_{\mathsf{Steps}}) \cdot (n-t)t'_{\mathcal{H}}$

protocol $\Pi^{\mathsf{new}}_{\mathsf{PL}}$ based on Construction 2 implements a robust public transaction ledger, except with negligible probability in λ.

A Cryptographic Primitives and Building Blocks

In this section we provide formal definitions for additional cryptographic primitives used throughout the paper.

Randomness Extractors. We make use of the notion of *weak* computational randomness extractors, as formalized in [22].

Definition 16. *An extractor is a family of functions $\mathsf{Ext} = \{\mathsf{Ext}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{d(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$, where $n(\cdot), d(\cdot)$ and $m(\cdot)$ are polynomials. The extractor is called weak $k(\cdot)$-computational if Ext_λ is PPT, and for all efficiently samplable probability ensembles $\{X_\lambda\}_\lambda$ with min-entropy $k(\lambda)$:*

$$(\mathsf{Ext}_\lambda(X_\lambda, U_{d(\lambda)}))_{\lambda\in\mathbb{N}} \overset{c}{\approx} (U_{m(\lambda)})_{\lambda\in\mathbb{N}}$$

where computational indistinguishability is defined w.r.t. a non-uniform distinguisher.

Robust Non-interactive Zero-Knowledge. We make use of the following composable notion of non-interactive zero-knowledge, introduced in [52].

[15] For simplicity, we assume that the cost in computational steps of evaluating H, and the hidden constant in the successful property of Lemma 5 are both 1. The theorem can be easily generalized for arbitrary costs.

Definition 17. *Given an NP relation R, let $L = \{x : \exists w \text{ s.t. } R(x,w) = 1\}$. $\Pi = (q, \mathsf{P}, \mathsf{V}, \mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2), \mathsf{E})$ is a* robust NIZK argument *for L, if $\mathsf{P}, \mathsf{V}, \mathsf{S}, \mathsf{E} \in PPT$ and $q(\cdot)$ is a polynomial such that the following conditions hold:*

1. *Completeness. For all $x \in L$ of length λ, all w such that $R(x,w) = 1$, and all $\Omega \in \{0,1\}^{q(\lambda)}$, $\mathsf{V}(\Omega, x, \mathsf{P}(\Omega, w, x))] = 1$.*
2. *Multi-theorem zero-knowledge. For all PPT adversaries $\mathcal{A}$, we have that $\textsc{Real}(\lambda) \approx \textsc{Sim}(\lambda)$, where*
$$\textsc{Real}(\lambda) = \{\Omega \leftarrow \{0,1\}^{q(\lambda)}; out \leftarrow \mathcal{A}^{\mathsf{P}(\Omega,\cdot,\cdot)}(\Omega); \textit{Output out}\},$$
$$\textsc{Sim}(\lambda) = \{(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); out \leftarrow \mathcal{A}^{\mathsf{S}_2'(\Omega,\cdot,\cdot,tk)}(\Omega); \textit{Output out}\},$$
and $\mathsf{S}_2'(\Omega, x, w, tk) \overset{def}{=} \mathsf{S}_2(\Omega, x, tk)$ if $(x,w) \in R$, and outputs `failure` *if $(x,w) \notin R$.*
3. *Extractability. There exists a PPT algorithm E such that, for all PPT $\mathcal{A}$,*
$$\Pr\left[\begin{array}{l}(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); (x,\pi) \leftarrow \mathcal{A}^{\mathsf{S}_2(\Omega,\cdot,tk)}(\Omega); w \leftarrow \mathsf{E}(\Omega,(x,\pi),tk) : \\ R(x,w) \neq 1 \wedge (x,\pi) \notin \mathcal{Q} \wedge \mathsf{V}(\Omega,x,\pi) = 1\end{array}\right] \leq \mathsf{negl}(\lambda)$$
where $\mathcal{Q}$ contains the successful pairs (x_i, π_i) that $\mathcal{A}$ has queried to S_2.

As in [28], we also require that the proof system supports labels. That is, algorithms $\mathsf{P}, \mathsf{V}, \mathsf{S}, \mathsf{E}$ take as input a label ϕ, and the completeness, zero-knowledge and extractability properties are updated accordingly. This can be achieved by adding the label ϕ to the statement x. In particular, we write $\mathsf{P}^\phi(\Omega, x, w)$ and $\mathsf{V}^\phi(\Omega, x, \pi)$ for the prover and the verifier, and $\mathsf{S}_2^\phi(\Omega, x, tk)$ and $\mathsf{E}^\phi(\Omega, (x,\pi), tk)$ for the simulator and the extractor.

Theorem 4. ([52]). *Assuming trapdoor permutations and a dense cryptosystem exist, robust NIZK arguments exist for all languages in $\mathcal{NP}$.*

B Iterated Hardness is Necessary

In this section, we demonstrate that an attack against iterated hardness implies an attack against the Bitcoin protocol, assuming the underlying hash function is collision-resistant and CRE (Definition 5). We phrase our attack against an abstraction of the Bitcoin protocol which appeared in [31], from which it is straightforward to extract a version of the protocol for our model. The main idea of the attack, is that if the hash function is CRE and not iteratively hard for appropriate parameters, then while honest parties' chains will grow at a fixed rate due to the CRE property, Bitcoin protocol's adversary can use the iterated hardness adversary to quickly produce a longer chain and break consistency. Due to space limitations we point to the full version of the paper for the proof.

Theorem 5. *Let $n, t, t_\mathcal{H}, t_\mathcal{A}$ such that $t_\mathcal{A} = c \cdot (n-t)t_\mathcal{H}$, for some $c \in (0,1)$. If $\mathcal{H}$ is collision-resistant and CRE, and the Bitcoin protocol from [31] satisfies Consistency with parameter k, then $\mathcal{H}$ is $\frac{c}{2} \cdot \frac{(n-t)t_\mathcal{H}}{(1-T/2^\lambda)^{(n-t)t_\mathcal{H}}}$-iteratively hard, for any polynomial k.*

As expected, as the computational power of the adversary decreases, the iteratively hard hash function needs to be less secure.

References

1. Alwen, J., Tackmann, B.: Moderately hard functions: definition, instantiations, and applications. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 493–526. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_17
2. Andrychowicz, M., Dziembowski, S.: PoW-based distributed cryptography with no trusted setup. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 379–399. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_19
3. Aspnes, J., Jackson, C., Krishnamurthy, A.: Exposing computationally-challenged Byzantine impostors. Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer Science, July 2005
4. Back, A.: Hashcash-a denial of service counter-measure (2002)
5. 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
6. Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Proofs of work from worst-case assumptions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 789–819. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_26
7. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. In: 37th Annual Symposium on Foundations of Computer Science, FOCS 1996, Burlington, Vermont, USA, 14–16 October 1996, pp. 514–523 (1996)
8. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997, pp. 394–403 (1997)
9. Bellare, M., Jaeger, J., Len, J.: Better than advertised: improved collision-resistance guarantees for md-based hash functions. In: CCS 2017, pp. 891–906. ACM, New York (2017)
10. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993 (1993)
11. Bellare, M., Rogaway, P.: The exact security of digital signatures-how to sign with RSA and Rabin. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_34
12. Bernstein, D.J., Lange, T.: Non-uniform cracks in the concrete: the power of free precomputation. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 321–340. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_17
13. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: Sudan, M. (ed.) Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, 14–16 January 2016, pp. 345–356. ACM (2016)
14. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25

15. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_15
16. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
17. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14–17 October 2001, Las Vegas, Nevada, USA, pp. 136–145. IEEE Computer Society (2001)
18. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 91–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_4
19. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_2
20. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004)
21. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
22. Dachman-Soled, D., Gennaro, R., Krawczyk, H., Malkin, T.: Computational extractors and pseudorandomness. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 383–403. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_22
23. Damgård, I.B.: Collision free hash functions and public key signature schemes. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 203–216. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-39118-5_19
24. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
25. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_30
26. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_24
27. Dwork, C., Naor, M.: Pricing via processing or combatting Junk Mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_10
28. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 465–488. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_20
29. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
30. Garay, J., Kiayias, A.: SoK: a consensus taxonomy in the blockchain era. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 284–318. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_13

31. 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
32. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. IACR Cryptol. ePrint Arch. **2014**, 765 (2014)
33. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_10
34. Garay, J.A., Kiayias, A., Leonardos, N., Panagiotakos, G.: Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 465–495. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_16
35. Garay, J.A., Kiayias, A., Panagiotakos, G.: Blockchains from non-idealized hash functions. Cryptology ePrint Archive, Report 2019/315 (2019). https://eprint.iacr.org/2019/315
36. Garay, J.A., Kiayias, A., Panagiotakos, G.: Consensus from signatures of work. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 319–344. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_14
37. Garay, J.A., MacKenzie, P., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. J. Cryptol. **24** (2011)
38. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) STOC 2011. ACM (2011)
39. Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, New York (2006)
40. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: Foundations of Computer Science (FOCS 2003). IEEE Computer Society (2003)
41. Juels, A., Brainard, J.G.: Client puzzles: a cryptographic countermeasure against connection depletion attacks. In: NDSS 1999. The Internet Society (1999)
42. Katz, J., Miller, A., Shi, E.: Pseudonymous secure computation from time-lock puzzles. IACR Cryptology ePrint Archive 2014:857 (2014)
43. Krawczyk, H.: Cryptographic extraction and key derivation: the HKDF scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_34
44. Lenstra, A.K., Wesolowski, B.: A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366 (2015). https://eprint.iacr.org/2015/366
45. Maurer, U. (ed.): EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9
46. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). http://bitcoin.org/bitcoin.pdf
47. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_8
48. Nisan, N., Zuckerman, D.: Randomness is linear in space. J. Comput. Syst. Sci. **52**(1), 43–52 (1996)
49. 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

50. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_33
51. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA (1996)
52. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_33