Iterated Search Problems and Blockchain Security under Falsifiable Assumptions

Juan A. Garay
Texas A&M University
garay@cse.tamu.edu

Aggelos Kiayias[*]
University of Edinburgh
& IOHK
akiayias@inf.ed.ac.uk

Giorgos Panagiotakos
University of Edinburgh
giorgos.pan@ed.ac.uk

June 25, 2019

Abstract

We put forth a new class of search problems, *iterated search problems* (ISP), and study their relation to the design of secure blockchain protocols. We prove that (i) the Bitcoin blockchain protocol implies a hard ISP problem, but ISP hardness is not by itself sufficient to prove its security, and (ii) a suitably enhanced class of ISPs is sufficient to imply, via construction, a secure blockchain protocol in the common reference string (CRS) model. We then put forth a specific proposal for an enhanced ISP based on an underlying cryptographic hash function. The resulting blockchain protocol's security reduces to the ISP hardness of the hash-based scheme and to a computational randomness extraction property of the hash function. As a corollary, we obtain a blockchain protocol secure in the standard model under falsifiable assumptions; in contrast, all previous blockchain protocols were shown secure in the random oracle model.

[*]Research partly supported by EU Project No.780477, PRIVILEDGE.

Contents

1 Introduction

Blockchain protocols, introduced by Nakamoto [40], are seen as a prominent application of the "proof of work" (PoW) concept to the area of consensus protocol design. PoWs were introduced in the work of Dwork and Naor [25] initially as a spam protection mechanism, and subsequently found applications in other domains such as Sybil attack resilience [24] and denial of service protection [35, 4], prior to their application to the domain of distributed consensus hinted at early on by Aspnes *et al.* [3].

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, 12]), there has been no definitive analysis of the primitive in the context of blockchain protocol security. 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 ("Byzantine")—a hallmark of the peer-to-peer setting where Bitcoin is designed for. Despite the fact that the Bitcoin blockchain has been analyzed formally [28, 44, 29, 5], the required PoW properties have not been identified and the analysis has been carried out in the random oracle (RO) model [9]. The same is true for a wide variety of other protocols in the space, including [2, 36, 30].

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 instance, Canetti *et al.* [17] 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 [42] 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 [34] showed that the Fiat-Shamir heuristic [27] does not necessarily imply a secure digital signature, which is in contrast with the result by Pointcheval and Stern [45] 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). Further, the formalization of non-idealized, concrete hash function assumptions sufficient to prove security of Bitcoin and related protocols has been identified as an important open question [16].

Towards identifying "good" cryptographic assumptions, Naor [41] introduced a framework for classifying cryptographic hardness assumptions and identified the notion of a *falsifiable assumption* which was further studied by Gentry and Wichs [33]. In the latter formulation, a falsifiable assumption is one that can be expressed as a game between an adversary and a challenger, and where the challenger can efficiently determine (and output) when the adversary wins the game. As a main result they show that succinct non-interactive arguments (SNARGs), which exist in the RO model, are impossible to construct in the standard model under any falsifiable assumption. The above highlights the main open question that motivates our work:

> *Is it possible to prove the security of blockchain protocols in the standard model under falsifiable assumptions? Furthermore, can these assumptions be simply expressed with respect to a given concrete hash function?*

Our results. In this paper we answer the above question in the positive, as follows. First, we put forth a new class of search problems, which we call *iterated search problems* (ISP). An instance description of an iterated search problem 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. Importantly, an ISP problem is equipped with a *successor* algorithm that given a statement x and a witness w, can produce a successor problem statement x', and a *solving* algorithm that given an initial problem statement x can find a sequence of witnesses, each corresponding to the next statement defined by algorithm S on input the previous statement and witness, starting from x. At the same time, if the solving algorithm takes t steps to solve k instances iteratively, no alternative algorithm can substantially speed this solving process up and produce k iterative solutions with non-negligible probability. This is the *iterated hardness* property of the ISP. We observe that it is easy to describe candidates for an ISP that are plausibly iteratively hard by employing a cryptographic hash function. Moreover, the iterated hardness property of the hash-based ISP is a falsifiable assumption.

Next, we prove that iterated hardness is necessary to implement a transaction ledger in the blockchain setting, focusing on the Bitcoin blockchain protocol. We achieve this by considering the natural ISP problem implied by any implementation of the Bitcoin blockchain. The ISP problem is defined by the blockchain structure itself: The instance is the hash of the previous block, the witness is the block content together with the nonce that determines that the PoW has been solved, and the successor function is the hash operation that creates the hash chain from which the chain structure of the blockchain is inherited. We prove that any successful attack against the iterated hardness of this ISP, results in a successful attack against the Bitcoin blockchain.

We then note that a related but distinct notion of hardness, *sequential* (i.e., non-parallelizable) iterated hardness, implies the existence of a hard ISP. This notion has been considered as early as [46], mainly in the domains of timed-release cryptography [15] and protocol fairness [32], and recently formalized in [13] under the term iterated sequential functions. In addition, a number of candidate hard problems have been proposed, including squaring a group element of a composite moduli [46], hashing, and computing the modular square root of an element on a prime order group [39]. Nevertheless, and importantily, we prove that the blockchain protocol which is based on the ISP that results from the underlying iterated sequential problem is *insecure*. The fundamental issue is that it does not allow for parallelization, which, as our negative result shows, is crucial for proving the security of any (Bitcoin-like) blockchain protocol. This also highlights that, even though necessary, iterated hardness is by itself insufficient to prove the security of a blockchain protocol.

Motivated by the above we identify a set of additional properties that epitomize what we call an *enhanced* ISP, from which we can provably derive the security of a blockchain protocol. The enhanced properties are as follows. First, 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 to the output of the successor over a random witness produced by the solving algorithm.

Armed with the above definitions we show a novel blockchain protocol whose security can be reduced to the hardness of the underlying enhanced ISP. 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. This is achieved by taking advantage of zero-knowledge proofs with efficient simulation and the ability of the reduction to efficiently extract a sequence of iterated witnesses despite the fact that the attacker may not produce consecutive blocks.

We perform our analysis in the static setting with synchronous rounds as in [28], 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. It is straightforward to extend our results to the Δ-synchronous setting of [44]. Further, we leave as an open question the extension of our results to the dynamic setting of [29] as well as matching the 50% threshold on adversarial computational power of the Bitcoin blockchain which can be shown in the RO model.

Finally, we put forth a new candidate construction for a hash-based ISP, for which it is possible to derive *all* the extra properties of the enhanced ISP from a simple property relating to the randomness extraction capability of the underlying cryptographic hash function. Moreover, we prove that the hardness of this new hash-based ISP can be derived from the hardness of the ISP that is implied by the actual Bitcoin blockchain implementation. Putting everything together we prove our main theorem which establishes the existence of a secure blockchain protocol whose security can be reduced to simple falsifiable assumptions regarding the underlying hash function—namely, collision resistance, computational randomness extraction and Bitcoin-ISP hardness.

Regarding related work, the notion of "correlation intractability" (CI) [16] bears some ressemblance to our ISP notion. There are, however, two main differences. First, CI only bounds the success probability in solving a single challenge, while our notion talks about (and fundamentally requires) multiple instances. Second, while CI talks about any sparse relation, our definition talks about a single non-sparse relation. Exploring further connections between the two notions is an interesting direction. Another work focusing on sufficient conditions for the consensus problem in the permissionless setting is [31], which introduced the concept of "signatures of work" (SoW) as the basic underlying assumption. The only known implementation of SoWs is in the RO model, hence it is unknown (and an interesting open question) whether SoWs can be realised under simpler falsifiable assumptions like the ones we consider here.

Organization of the paper. The basic computational model, definitions and cryptographic building blocks used by our constructions are presented in Section 2. The formulation of iterated search problems, as well as proofs of necessity and by-itself insufficiency of the iterated hardness property for proving the security of blockchain protocols are presented in Section 3. The conditions that make an enhanced ISP and the provably secure blockchain protocol based on it, together with a candidate enhanced ISP construction, are presented in Section 4. Other complementary material is presented in the appendix.

2 Preliminaries

In this section we introduce basic notation and definitions that we 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. For a set X, $x \overset{\$}{\leftarrow} X$ 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 $\{0,1\}^m$. We denote the statistical distance between two random variables X, Z with range U by $\Delta[X, Y]$, i.e.,

$\Delta[X,Z] = \frac{1}{2}\sum_{v\in U} |\Pr[X=v] - \Pr[Z=v]|$. For $\epsilon > 0$, we say that X, Y are *ϵ-close* when $\Delta(X,Y) \le \epsilon$. $\approx$ and $\overset{c}{\approx}$ denote statistical and computational indistinguishability, respectively. We let λ denote the security parameter.

Protocol execution and security models. In this paper we will follow the concrete approach [7, 10, 32, 11] to security evaluation rather than the asymptotic one. 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. [11]; 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.* [32]; refer to [32] 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 $\Pr[\mathsf{Steps}_M(1^\lambda, x) \le t(\lambda)] = 1$.

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 [28], an abstraction was proposed for the analysis of the Bitcoin protocol. Later, in [31], it was extended to accommodate a standard-model analysis of PoW-based blockchain protocols. We adopt the latter model for our analysis, and overview the basics here.

The execution of a protocol Π is driven by an "environment" program $\mathcal{Z}$ that may spawn multiple instances running the protocol Π . The programs in question can be thought of as "interactive RAMs" communicating through registers in a well-defined manner, with instances and their spawning at the discretion of a control program which is also an IRAM and is denoted by C. In particular, the control program C forces the environment to perform a "round robin" participant execution sequence for a fixed set of parties.

Specifically, the execution driven by $\mathcal{Z}$ is defined with respect to a protocol Π, an adversary $\mathcal{A}$ (also an IRAM) and a set of parties $P_1, ..., P_n$; these are hardcoded in the control program C. The protocol Π is defined in a "hybrid" setting and has access to one ideal communication functionality, called the *diffusion channel* (see below). It is used as subroutine by the programs involved in the execution (the IRAMs of Π and $\mathcal{A}$) and is accessible to all parties once they are spawned.

Initially, the environment $\mathcal{Z}$ is restricted by C to spawn the adversary $\mathcal{A}$. Each time the adversary is activated, it may communicate with C via messages of the form (Corrupt, P_i). The control program C will register party P_i as corrupted, only provided that the environment has previously given an input of the form (Corrupt, P_i) to $\mathcal{A}$ and that the number of corrupted parties is less or equal t, a bound that is also hardcoded in C. The first party to be spawned running protocol Π is restricted by C to be party P_1. After a party P_i is activated, the environment is restricted to activate party P_{i+1} , except when P_n is activated in which case the next party to be activated is always the adversary $\mathcal{A}$. Note that when a corrupted party P_i is activated the adversary $\mathcal{A}$ is activated instead. In this work we consider only static adversaries.[1]

[1] We note that all our results also hold against an adaptive adversary, if we allow erasures.

In addition, we assume the existence of a *common reference string* (CRS) that becomes available to all parties at the start of the execution. The CRS is sampled in a trusted manner from a known efficiently samplable distribution.

Next, we describe how different parties communicate. Initially, the diffusion functionality sets the variable round to be 1. It also maintains a Receive() string defined for each party P_i. A party is allowed at any moment to fetch the contents of 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; note that m is allowed to be empty. At any moment, the adversary $\mathcal{A}$ is allowed to receive the contents of all messages for the round and 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() strings and includes any messages that were diffused by the parties in the current round but not contributed by the adversary to the Receive() strings. The variable round is then incremented.

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 $t_{\mathcal{A}}$;[2]
- 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.

Cryptographic primitives and building blocks. We will make use of the following cryptographic primitives: Cryptographic hash functions, (computational) randomness extractors [43, 19], robust non-interactive zero-knowledge (NIZK) [47], and iterated sequential functions [13]. Except for the last notion, which is fairly new, refer to Appendix A for the formal security definitions.

Definition 3 ([13])**.** $f : X \to Y$ is a *(t, ϵ)-sequential function* for $\lambda = O(\log(|X|))$, if the following conditions hold:

1. There exists an algorithm that for all $x \in X$ evaluates f in parallel time t using $\mathsf{poly}(\log(t), \lambda)$ processors.
2. For all $\mathcal{A}$ that run in parallel time strictly less than $(1 - \epsilon) \cdot t$ with $\mathsf{poly}(t, \lambda)$ processors,

$$\Pr[y_{\mathcal{A}} = f(x) | y_{\mathcal{A}} \leftarrow \mathcal{A}(\lambda, x), x \leftarrow X] < \mathsf{negl}(\lambda).$$

Definition 4 ([13])**.** Let $g : X \to X$ be a function which satisfies (t, ϵ)-sequentiality. A function $f : \mathbb{N} \times X \to X$ defined as $f(k, x) = g^{(k)}(x) = (g \circ g \circ \ldots \circ g)(x)$ is called an *iterated sequential function* (with round function g), if for all $k = 2^{o(\lambda)}$, the function $h : X \to X$ such that $h(x) = f(k, x)$ is (kt, ϵ)-sequential.

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

Robust public transaction ledgers. Our work is concerned with necessary and sufficient conditions to implement a transaction ledger. Next, we give the definition introduced in [28]. Note that the Liveness definition we adopt is a slight strengthening of the one in [28], introduced in [44].

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 in the Bitcoin protocol. It is possible for the adversary to create two transactions that are conflicting; valid ledgers must not contain conflicting transaction. 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.

Definition 5. 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:

- **Persistence:** Parameterized by $k \in \mathbb{N}$ (the "depth" parameter), 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 (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 the next round on.
- **Liveness:** Parameterized by $u, k \in \mathbb{N}$ (the "wait time" and "depth" parameters, resp.), for every u consecutive rounds there exists an honest party who will report as stable the transactions given as input to an honest party (possibly the same) at a round in this interval, that are either (i) issued by Txgen, or (ii) are neutral.

3 Necessary Conditions for PoW-based Blockchain Protocols

In this section we study the necessary conditions for the security of PoW-based blockchain protocols, focusing on (an abstraction of) Bitcoin. Our aim will be to distill what are the necessary properties the underlying moderately hard problem should satisfy. In the Bitcoin protocol this problem is related to a real-world hash function, namely, SHA-256. We abstract the problem in Section 3.1, where we introduce the notion of *hard iterated search problems*. Then, in Section 3.2 we describe a generalized Bitcoin-like protocol based on such problems, and show that both the existence of such problems is necessary to prove the security of Bitcoin, and yet it is not sufficient.

3.1 Iterated search problems

Next, we introduce a class of problems expressive enough to abstract Bitcoin's underlying computational problem. The straightforward requirements are the ability to find a witness for a problem statement, and to verify that a witness is correct, matching the block mining and block verification procedures, respectively. In addition, we require 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, whose name we borrow 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, we will later argue that iterated sequential functions are *not* the correct abstractions for the underlying computational problem, as they allow for an attack against the protocol. We proceed to give a formal definition.

An *iterated search problem* (ISP) $\mathcal{I}$ specifies a collection $(I_\lambda)_{\lambda\in\mathbb{N}}$ of distributions.[3] For every value of the 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 $(I_\lambda)_\lambda$ have polynomial-time samplers. The following are the algorithms provided by an ISP, 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[4] 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. As an example, we present Bitcoin's underlying computational problem captured as an ISP.

Construction 1. Let T be a protocol parameter that has to do with how hard it is to solve a problem instance.[5] 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 $W = \{0,1\}^* \times \{0,1\}^\lambda$;
- $R = \{(x, w) | H(H(x||m)||ctr) < T, \text{ for } w = m||ctr\}$;
- $V(x, w)$ checks whether $H(H(x||m)||ctr) < T$, for $w = m||ctr$;
- $S(x, w) = H(H(x||m)||ctr)$, 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 $M(x', 1)$ on the statement x' produced by $M(x, k-1)$. The output consists of all witnesses found.

To ease the presentation, we recursively extend the definitions of S and R to sequences of witnesses as follows:

- $S(x, (w_i)_{i\in[k]}) = \begin{cases} x, & k = 0; \\ S(S(x, (w_i)_{i\in[k-1]}), w_k), & k \geq 2. \end{cases}$
- $(x, (w_i)_{i\in[k]}) \in R \overset{\triangle}{\Leftrightarrow} \bigwedge_{i=1}^{k} (S(x, (w_j)_{j\in[i-1]}), w_i) \in R.$

We require that for all $\lambda \in \mathbb{N}$, all $\Lambda[X, W, R] \in I_\lambda$, and for all $x \in X$, there exists $w \in W$ such that $(x, w) \in R$. Moreover, we assume that M is *correct*, i.e., for $(w_i)_{i\in[k]} \leftarrow M(x, k)$, it holds that $(x, (w_i)_{i\in[k]}) \in R$ with overwhelming probability in λ.

Finally, in the same spirit of Boneh *et al.* [13]'s definition of an iterated sequential function (cf. Definition 3), we 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,

[3] Here we follow the notation used to define *subset membership problems* in [18]. We remark that no other connection exists between the two papers.

[4] 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 [29], which we do not study here, where the next block's difficulty depends on the whole chain.

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

our notion relaxes the strict convergence criterion of [13]. Finally, we note that is possible to define the error probabilities concretely. For simplicity, we chose not to do so, as the current formulation still suffices to capture the qualitative aspect of the primitive.

Definition 6. An ISP $\mathcal{I} = (V, M, S)$ is (t, δ, k_0)-*hard* iff it holds that:

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

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

- for any PPT RAM $\mathcal{A}$, for sufficiently large $\lambda \in \mathbb{N}$ and for all polynomially large $k \geq k_0$:

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

Remark 1. While in this section we focus on the hardness of an ISP, later we are going to require more properties for it to be useful, including fast verification and the ability to parallelize the solver. If we did not require any extra properties, we could design trivially hard ISPs, for example based on the hardness of outputting long sequences of 1 in the output register.

3.2 An ISP-based Bitcoin protocol

Next, we describe an adaptation of the Bitcoin public ledger protocol [28] with respect to an ISP $\mathcal{I} = (M, V, S)$; we call the resulting protocol $\Pi^{\mathsf{isp}}_{\mathsf{PL}}(\mathcal{I})$. We give a high-level overview of the protocol, mostly focusing on the points where it differs from that in [28]. We refer the reader to [28] for a detailed exposition.

At the start of the protocol, all parties have access to the CRS which contains a randomly sampled instance description $\Lambda[X, W, R]$ and a randomly sampled statement $x_{\texttt{Gen}} \in X$. We use the terms block and chain to refer to tuples of the form $\langle x, w \rangle \in X \times W$, and sequences of such tuples, respectively. A block $B = \langle x, w \rangle$ is *valid* if $(x, w) \in R$; a chain $\mathcal{C} = (\langle x_i, w_i \rangle)_{i \in [k]}$ is *valid* if (i) $x_1 = x_{\texttt{Gen}}$, (ii) the i-th block is valid, and (iii) $x_{i+1} = S(x_i, w_i)$, for all $i \in [k]$.

The protocol proceeds as follows: Each party tries to "extend" its chain of blocks (initially just $x_{\texttt{Gen}}$) using M. In case it receives a valid chain $\mathcal{C}$ that is longer than the chain that the party tries to extend, it adopts the longer chain, and runs M on the new problem defined by this chain. If M finishes executing before receiving a new chain, the party diffuses the newly found chain to the network, and stops for this round. We assume that all honest parties take the same number of steps t_H per round. If the allowed steps do not suffice for M to finish in a specific round, the program is interrupted until the next round. Finally, if queried by the environment, the parties output the contents of the chain they have selected, excluding the last k blocks. The contents of any chain are determined by the chain reading function $\mathrm{R}(\cdot)$, which takes as input a chain and outputs a transaction ledger (see Remark 2).

Remark 2. In the ISP definition, we do not explicitly model the ability of the solver M to encode information in the witnesses it computes. The reason for this is to better highlight the hardness properties of the ISP, instead of its use as an encoding mechanism. However, in order for Bitcoin to implement a transaction ledger, it is necessary that M implements such a mechanism. Otherwise, honest parties will not be able to add to the chain the transactions given to them by the environment, and the Liveness property would not hold. Hence, to keep the ISP definition uncluttered, and still be able to argue about the necessary hardness condition it should satisfy, we opt for a weak encoding property for $\mathcal{I}$: We assume that M encodes a random transaction on the witness it outputs. More formally, we assume that for any input $x \in X$, the contents of block $\langle x, M(x, 1) \rangle$ extracted by $R(\cdot)$, are distributed uniformly on the transaction space. This corresponds to an execution of the Bitcoin backbone protocol where honest parties only generate random transactions locally, e.g., coinbase transactions. This change is w.l.o.g., as the necessary conditions derived for this restricted scenario should also hold in the general case where more complex scenarios regarding transactions are allowed.

3.3 Iterated hardness is necessary

We now analyze the necessary conditions that Bitcoin's underlying ISP should satisfy in order for Bitcoin to implement a ledger. In particular, we show that the ISP used in Bitcoin must be hard (Definition 6). In more detail, we prove that no adversary that is suitably computationally bounded, should be able to find witnesses at a rate much higher than that of the honest parties. Other, somewhat secondary properties are necessary to state our subsequent result. We start with those.

The first property has to do with the event that two different chains define the same next problem statement to be solved. If such an event happens, it can lead to situations where a chain contains a cycle, which the protocol cannot handle.

Definition 7. An ISP $\mathcal{I} = (V, M, S)$ is *collision resistant* iff for all PPT $\mathcal{A}$ and sufficiently large $\lambda \in \mathbb{N}$, it holds that

$$\Pr_{\substack{\Lambda[X,W,R]\leftarrow I_\lambda;\\ x\leftarrow X}}\left[\begin{array}{l}((w_i)_i,(w'_j)_j)\leftarrow \mathcal{A}(1^\lambda,\Lambda,x): S(x,(w_i)_i)=S(x,(w'_j)_j)\\ \wedge\, (x,(w_i)_i)\in R \wedge (x,(w'_j)_j)\in R\end{array}\right] \le \mathsf{negl}(\lambda).$$

Additionally, we assume that there exists an upper bound t_{ver} on the time it takes to evaluate the verification algorithm V, such that $t_{\mathsf{ver}} \cdot \theta$ is insignificant compared to $t_\mathcal{H}$, where θ is the upper bound on the messages send per round and $t_\mathcal{H}$ is the number of steps each honest party takes per round. Intuitively, the time to verify that a witness is valid, must be a lot smaller than the rate at which the adversary can send messages to honest parties. Otherwise, the adversary can launch a denial of service attack by spamming parties with "fake" witnesses, making them spend most of their computing power on verifying them.

For the rest of this section we will assume that $\mathcal{I}$ encodes random transactions, is collision resistant, and $t_{\mathsf{ver}} \cdot \theta$ is insignificant compared to $t_\mathcal{H}$. We are now ready to state the necessary hardness condition regarding ISPs. The main idea is that any adversary whose computational power is a constant fraction of that of the honest parties, if $\mathcal{I}$ is not a hard-ISP for appropriate parameters, can extend her private chain faster than the rate at which honest parties extend their own chains. Hence, it can create a long enough fork and break Persistence.

Theorem 8. *Let $n, t, t_\mathcal{H}, t_\mathcal{A}$ such that $t_\mathcal{A} = c \cdot (n-t)t_\mathcal{H}$, for some $c \in (\frac{1}{n-t}, 1)$, and $\mathcal{I} = (V, M, S)$ be an ISP. If $\Pi^{\mathsf{isp}}_{\mathsf{PL}}(\mathcal{I})$ satisfies Persistence and Liveness with parameters u, k, then there exists an ISP $\mathcal{I}'$ that is $(\frac{u}{k} \cdot (n-t)t_\mathcal{H}, 1 - \frac{c}{2u}, 2k+2)$-hard.*

Proof. Let M' be a RAM, that takes the as input (x', k'), and performs the same computation as the $n-t$ honest parties in the Bitcoin execution, i.e., it uses x' as x_{Gen} and runs the Bitcoin algorithm with no adversarial interference until a chain of length k' is computed. Then, it outputs the witnesses of the chain as the hard-ISP experiment dictates. We will show that $\mathcal{I}' = (V, M', S)$ is a hard ISP.

Let $\delta = 1 - \frac{c}{2u}$, and, for the sake of contradiction, assume that $\mathcal{I}'$ is not $(\frac{u}{k} \cdot (n-t)t_\mathcal{H}, \delta, k)$-hard. First, we will show that the first part of Definition 6 is satisfied. Assume that the adversary in the Bitcoin execution stays inactive. Due to Liveness, every u rounds honest parties add a new transaction k blocks deep in some chain. Repeating the same argument $\frac{m}{k}$ times, and since honest parties extend always the longest chain, it follows that after $\frac{m}{k} \cdot u$ rounds honest parties will have added m new blocks to the blockchain with overwhelming probability on λ. Equivalently, M' computes m blocks in

$m \cdot \frac{u}{k} \cdot (n-t)t_{\mathcal{H}}$ steps ($\frac{m}{k} \cdot u$ rounds), for any polynomially large $m \geq 2k+2$. Hence, the first part of the hardness definition is satisfied, and thus the second part should not hold. This implies that there exists an adversary $\mathcal{A}$ that for infinitely many λ and $k_1 \geq 2k+2$ computes more blocks that the hardness definition allows, with non-negligible probability. We are going to use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ that creates a long enough chain privately, and breaks Persistence.

$\mathcal{A}'$ takes as input the genesis block, and runs $\mathcal{A}$ on input $(\Lambda, x_{\texttt{Gen}})$. If it computes k_1 blocks before the honest parties, it waits until some honest party computes a chain of length at least $k_1 - k$, and sends its own chain to a different party. We proceed to analyze the probability that $\mathcal{A}'$ breaks Persistence.

First, we show that $\mathcal{A}'$ computes k_1 blocks before the honest parties. It holds that

$$(1-\delta) \cdot k_1 \cdot \frac{u}{k} \cdot (n-t)t_{\mathcal{H}} \leq \frac{c}{2u} k_1 \frac{u}{k} \frac{t_{\mathcal{A}}}{c} \leq t_{\mathcal{A}} \cdot \frac{k_1}{2k} \leq t_{\mathcal{A}} \cdot (\frac{k_1}{k} - 1) < t_{\mathcal{A}} \cdot \lfloor \frac{k_1}{k} \rfloor$$

This implies that $\mathcal{A}'$ is going to take $\lfloor \frac{k_1}{k} \rfloor$ rounds to compute k_1 blocks with non-negligible probability. For the moment assume that with overwhelming probability no honest party computes more than k blocks in any single round; we will prove this at the end. It follows that it takes honest parties at least $\lfloor \frac{k_1}{k} \rfloor$ rounds to compute a chain of length k_1 with overwhelming probability. Hence, with non-negligible probability $\mathcal{A}'$ will compute a chain of length k_1 before the honest parties.

Next, we argue that the chain that $\mathcal{A}'$ diffuses to the network, is going to break Persistence. We have already shown that due to Liveness, at some round an honest party is going to compute and transmit a chain of size at least $k_1 - k \geq k+2$, consisting only of honestly mined blocks. As discussed before, when the adversary sees that, it immediately transmits its own longer chain to a different party; if all honest parties extend their chains, it picks a random one. Now note that necessarily the adversarial chain is the longest one transmitted at this round. Otherwise, some honest party must have computed more than k blocks in that round, which happens only with negligible probability. Moreover, since the transactions in the honest blocks are distributed uniformly, and the output of the adversary does not depend on them, with overwhelming probability the transactions in the stable honest blocks are going to be different from those in the chain crafted by the adversary. Therefore, two honest parties will have conflicting views, and Persistence does not hold with parameter k with non-negligible probability.

Finally, we argue that with overwhelming probability no honest party computes more than k blocks in any single round. Let E_1 be the event that there exists party P and round r such that P at r computes more than k blocks, and its chain is as long as the longest chain transmitted by any honest party at this round. For the sake of contradiction, assume this event occurs with non-negligible probability ϵ. We will describe an adversary $\mathcal{A}$ that breaks Persistence if this is the case. $\mathcal{A}$ picks a round r' and a party P' at random, and at round r' simulates exactly the behavior of P' at that round (note that it only needs $t_{\mathcal{H}}$ steps to do that, which we assume it has, i.e., $c \geq (n-t)^{-1}$). If it computes more than k blocks, it picks a random round and a random party and transmits its chain to it then.

We next proceed to analyze the probability that $\mathcal{A}$ breaks Persistence. Let $E_2^{P,r}$ be the event that party P at round r computes more than k blocks, and its chain is as long as the longest chain transmitted by any honest party at this round, and note that $\bigvee_{(P,r) \in \mathcal{P} \times p(\lambda)} E_2^{P,r} = E_1$, where $\mathcal{P}$ is the set of honest parties, and $p(\lambda)$ is a polynomial upper bound on the number of rounds. Such a bound always exists, since the execution is driven by $\mathcal{Z}$, which is a PPT RAM. Also, let $E_3^{P,r}$ be the event that $\mathcal{A}$ has a chain at least as long as P at round r, and E_4 be the event that $\mathcal{A}$ at round r' has a chain that is longer than all parties. We are going to show that $\Pr[E_4]$ is non-negligible. It holds that

$$\begin{aligned}
\Pr[E_4] &= \Pr_{(P',r')\leftarrow\mathcal{P}\times p(\lambda)}[E_2^{P',r'} \wedge E_3^{P',r'}] && \text{(total probability)}\\
&= \sum_{(P',r')\in\mathcal{P}\times p(\lambda)} \Pr[E_2^{P,r} \wedge E_3^{P,r} \wedge (P,r) = (P',r')] && \text{(independence)}\\
&= \sum_{(P',r')\in\mathcal{P}\times p(\lambda)} \Pr[E_2^{P,r} \wedge E_3^{P,r}] \Pr[(P,r) = (P',r')] && \text{(symmetry)}\\
&\geq \sum_{(P',r')\in\mathcal{P}\times p(\lambda)} \frac{\Pr[E_2^{P,r}]}{2} \Pr[(P,r) = (P',r')] && \text{(union bound)}\\
&\geq \frac{1}{2n\cdot p(\lambda)} \Pr[\bigvee_{(P,r)\in\mathcal{P}\times p(\lambda)} E_2^{P,r}] = \frac{\Pr[E_1]}{2n\cdot p(\lambda)} \geq \frac{\epsilon}{2n\cdot p(\lambda)} > \mathsf{negl}(\lambda)
\end{aligned}$$

We can extend this argument to show that with non-negligible probability $\mathcal{A}$ will send its chain at a party that has a smaller chain, and some other party at the next round will compute a chain that will create a fork of size more than k, and break Persistence. Hence, $\mathcal{A}$ breaks Persistence with non-negligible probability. □

Note that as c approaches 1, the ISP should be secure against stronger adversaries. This is in line with our intuition: The better the security of the resulting protocol is, the harder the requirements from the ISP are.

Next, we show that the existence of iterated sequential functions (cf. Definition 3) that can be evaluated using a constant number of processors, implies that of hard ISPs, for appropriate parameters. Note, that if the standard solver of the assumed (t,δ)-ISF requires very small parallelization, i.e. $c \approx 1$, the resulting ISP is $(t,\delta,1)$-hard.

Lemma 9. *If there exist a $(\hat{t},\delta)$-iterated sequential function, which can be evaluated in time $\hat{t} \in poly(\lambda)$ with $c = O(1)$ parallel processors, then there exists a $(c\cdot\hat{t}, 1-\frac{1-\delta}{c}, 1)$-hard ISP.*

Proof. Let $f_\lambda : X \to X$ be a $(\hat{t},\delta)$-iterated sequential function, for some $\lambda \in \mathbb{N}$. Let $\mathcal{I}$ be the following ISP: $R(x,w) = \{(x, f_\lambda(x)) | x \in X\}$, S is equal to the identity function, $I_\lambda = \{[X, X, R]\}$, and M is the standard solver of f_λ. By the ISF definition, M on input x computes $f(x)$ in $\hat{t}$ sequential steps with parallelization c, i.e, $c\hat{t}$ steps in total. On the other hand, for any $k \geq 1$, the adversary takes $(1-\delta)k\hat{t} = (1-(1-\frac{1-\delta}{c}))kc\hat{t}$ sequential steps to compute $f^k(x)$. Hence, $\mathcal{I}$ is a $(c'\cdot\hat{t}, 1-\frac{1-\delta}{c}, 1)$-hard ISP. □

With foresight, we conclude this subsection by stating a stronger hardness property that allows the adversary some precomputation time. This will be useful in the next section, where we present and prove secure a protocol based on an enhanced ISP. We note, that a more convoluted version of this property, where the problem statement x is sampled from a distribution that has to do with the solving algorithm M, can be proven necessary in a similar way as Theorem 8; our results in Section 4 apply for both definitions.

Definition 10. An ISP $\mathcal{I} = (V, M, S)$ is (t,δ,k_0)*-hard against precomputation* iff $\mathcal{I}$ is (t,δ,k_0)-hard (Definition 6) and for any PPT RAM $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, for sufficient large $\lambda \in \mathbb{N}$, and all polynomially large $k \geq k_0$, it holds that

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

Remark 3. Our analysis, with minor differences, also applies to other well-known PoW-based blockchain protocols. For example, in the GHOST protocol [48] parties choose their chain by successively picking the "heaviest" subtree of blocks, starting from the genesis block. By similar arguments as above, it is implied that it is necessary for the ISP to be hard with respect to an upper bound on the rate at which the honest parties generate blocks (cf. the chain growth rate in Bitcoin). However, while this may sound as an improvement, it opens up the space for other attacks against the ISP; for example, if computing a new witness for a specific statement becomes easy after computing the first one, the adversary can create a large enough "tree" of blocks and break Persistence essentially at the cost of computing a single witness. This is not the case for Bitcoin, that can withstand such an attack.

3.4 Iterated sequential functions are not sufficient

We showed above that iterated sequential functions imply the existence of hard ISPs. In this section, we explore the question of whether Bitcoin can be solely based on such functions, and show that this is not the case. This also implies that assuming only that the underlying problem of Bitcoin is a hard ISP, is not sufficient to prove Bitcoin secure.

The first problem we encounter when trying to base Bitcoin on the ISP implied by an ISF, is that ISFs are functions, i.e., for a single problem statement there exists only one possible witness. This in turn means that it is impossible to encode information in the witness, which as argued in the previous section, is necessary to prove security. This aspect, however, is not the only problem. Assume that an *iterated sequential relation* (ISR), the generalization of ISF to relations with similar security guarantees, exists, and that it satisfies all the necessary conditions described in the previous section—i.e., collision resistance, fast verification and the ability to encode a random message to the witness. Even then, we can show that if we base our protocol to an ISR, it will be insecure. The main reason being that the solving algorithm must be *parallelizable*—exactly the opposite of what sequential functions or relations guarantee.

Informally, the solving algorithm M must be parallelizable in the following sense: Running multiple instances of M in parallel should result in finding a witness *before* a single invocation of M that runs for about the same number of steps in total, with non-negligible probability. This corresponds to the fact that in Bitcoin the computation of blocks is distributed among several parties which work independently, while the adversary is modeled as a single RAM. If M is not parallelizable, the adversary can run a single instance of M for less steps than the honest parties in total, and extend its chain as fast as them. This in turn means that it can create a deep enough fork and break the security of the protocol. This is exactly the case for (ideal) perfectly secure ISRs: An adversary that has the same power as approximately *one* honest party, needs exactly the same time as all honest parties to run M and find a solution, no matter their number.

Next, we show that under the assumption that all machines in our computational model take sequential steps with approximately the same rate, and have some small amount of parallelization available, (ideal) perfectly secure ISRs cannot be used to prove Bitcoin secure. Our assumption about the adversary is realistic, since it is possible that it has dedicated hardware that runs faster than the average modern processor. Formally, similarly to [14], we slightly modify our model and assume that the machines of honest parties in our execution are parallel RAMs that have c processors, for $c = O(1)$, while the adversary is free to choose the number of processors its machine has, as long as the rate of sequential steps $\mathsf{seq}_{\mathcal{A}}$ it takes is close to that of each honest party $\mathsf{seq}_{\mathcal{H}}$, i.e, $\mathsf{seq}_{\mathcal{A}} \leq (1+\epsilon) \cdot \mathsf{seq}_{\mathcal{H}}$, for some $\epsilon \in [0,1)$. The total number of steps each party takes per round is the product of the number of processor it uses times the number of sequential steps it takes. We call this the (c,ϵ)-*sequential model.*

Theorem 11. *Let R be an $(\hat{t},0)$-iterated sequential relation which can be evaluated in time $\hat{t} \in poly(\lambda)$ with $c = O(1)$ parallel processors, and $\mathcal{I}$ be the implied ISR instance. For $t_{\mathcal{H}}, t_{\mathcal{A}}$ and any polynomial*

$p(\cdot)$, such that $t_{\mathcal{A}} = \frac{t_{\mathcal{H}}}{1+\frac{1}{p(\lambda)}}$, $\Pi_{\mathsf{PL}}^{\mathsf{isp}}(\mathcal{I})$ does not satisfy Persistence for any polynomially large k in the $(c, \frac{1}{p(\lambda)})$-sequential model.

Proof. Let M be the standard solver for R, and assume that Persistence holds for some k. We will describe an adversary $\mathcal{A}$ that uses M to break Persistence with non-negligible probability. By the sequentiality property of the ISR, it is implied that the honest parties compute chains of length k', after at least $k'\hat{t}$ sequential steps with overwhelming probability. On the other hand, the adversary, by running a single instance of M iteratively, can also create a chain of length k' in $k'\hat{t}$ sequential steps.

We will assume now that $\mathcal{A}$ takes $(1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}}$ sequential steps per round with parallelization c. It can do this, since $c(1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}} = \frac{t_{\mathcal{H}}}{(1+\frac{1}{p(\lambda)})} = t_{\mathcal{A}}$. It follows that $\mathcal{A}$ needs $\lceil \frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{A}}} \rceil$ rounds to compute the blocks, while the honest parties need $\frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{H}}}$. It holds that for $k' \geq \max\{k, \frac{1}{\hat{t}} \cdot (1+p(\lambda)) \cdot \mathsf{seq}_{\mathcal{H}}\}$:

$$\begin{aligned}\lceil \frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{A}}} \rceil < \frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{A}}} + 1 \leq \frac{k'\hat{t}}{(1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}}} + 1 \leq \frac{k'\hat{t} + (1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}}}{(1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}}} \\ \leq \frac{k'\hat{t} + k'\hat{t} \cdot \frac{1}{p(\lambda)}}{(1+\frac{1}{p(\lambda)})\mathsf{seq}_{\mathcal{H}}} \leq \frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{H}}} \leq \lceil \frac{k'\hat{t}}{\mathsf{seq}_{\mathcal{H}}} \rceil\end{aligned}$$

Hence, the adversary with overwhelming probability computes a chain of k' blocks faster than the honest parties for some polynomially large k'. Note, that the adversary can also tell exactly when the honest parties are going to compute a chain of the same length, and thus break Persistence for any k as in Theorem 8. The theorem follows. □

4 Sufficient Conditions and a Provably Secure ISP-based Blockchain

We first give a complete overview of the technical results of this section regarding the implementation of a transaction ledger based on an ISP.

As a first intermediate step, in Section 4.1, we list a set of additional properties that epitomize what we call an *enhanced* ISP, from which subsequently we show that we can provably derive the security of a blockchain protocol. The enhanced properties are as follows. First, 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 to the output of the successor over a random witness produced by the solving algorithm.

Armed with the above definitions we show a novel blockchain protocol whose security can be reduced to the hardness of the underlying enhanced ISP. 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. This is achieved by taking advantage of zero-knowledge proof simulation and the ability of the reduction to extract a sequence of iterated witnesses despite the fact that the attacker may not produce consecutive blocks.

At a first glance, the fact that we use ZK proofs for a language that is moderately hard may seem counterintuitive, due to the fact that a trivial simulator and extractor exist for the zero-knowledge and soundness properties, since computing a new witness for a given statement takes polynomial time. Not in our case, as 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. Furthermore, the construction requires the robustness of the NIZK in the sense of [47], which in turn can be inferred by one-way trapdoor permutations and a dense public-key cryptosystem (see Appendix A). The theorem we prove is as follows.

Theorem 38 (Informal). *Assuming the existence of a collision-resistant hash function family, a one-way trapdoor permutation, a dense cryptosystem and an enhanced ISP problem with appropriate parameters, there exists a protocol that implements a transaction ledger.*

We note that we perform our analysis in the static setting with synchronous rounds as in [28], 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. It is straightforward to extend our results to the Δ-synchronous setting of [44].

Finally, in Section 4.3, we show how the enhanced-ISP properties, besides hardness, can be instantiated via a suitable computational randomness extractor. In more detail, we put forth a new candidate ISP construction (Construction 2) that is based on a hash function that is also a computational randomness extractor. The main characteristic of this new ISP 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. 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. Based on these observations, we can prove that the enhanced properties introduced in the beginning of the section are satisfied. Furthermore, we prove that the hardness of this new ISP can be derived from the hardness of the Bitcoin ISP construction. Putting everything together results in the following:

Theorem 42 (Informal). *Assuming the existence of a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and that Construction 1 is a hard-ISP, based on a collision resistant hash family $\mathcal{H}$ that is a computational randomness extractor, with appropriate parameters, there is a protocol that implements a robust public transaction ledger.*

4.1 Enhanced ISPs

As we showed in the previous section, the existence of hard ISPs is necessary, but not sufficient to prove Bitcoin secure. Here we first define an enhanced, "blockchain-friendly" notion of security for ISPs, encompassing hardness, which we then show to be sufficient to implement a provably secure blockchain via a new protocol. We conclude the section presenting a concrete ISP proposal that satisfies the extra security properties and is plausibly hard.

First, we present a set of extra ISP properties, in addition to hardness, that are specific to the use of an ISP in blockchain applications. We note that the properties' specifics are not necessary for the description of our protocol, hence the eager reader can directly proceed to Section 4.2.

The first property, has to do with establishing an upper bound t on the the running time of the verification algorithm V, for the reasons explained in Section 3.

Definition 12. 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[6] computes a witness in t steps is at least α.

Definition 13. An ISP $\mathcal{I} = (V, M, S)$ is (t, α)-*successful* iff for sufficiently large $\lambda \in \mathbb{N}$ for all $\Lambda[X, W, R] \in I_\lambda$, and for all $x \in X$ it holds that:

$$\Pr\left[\,\mathsf{Steps}_M(x) < t\,\right] \geq \alpha.$$

The iterated hardness property, as formulated in the previous section, does not give any guarantees regarding composition. In the Bitcoin setting, 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 running time and 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.

Definition 14. 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, for sufficiently large $\lambda \in \mathbb{N}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $x \in X$ it holds that

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

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 blocks the adversary actually produced. We call this ISP property *witness malleability.*

Definition 15. 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, for sufficiently large $\lambda \in \mathbb{N}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $(x, w) \in R$ it holds that $(x, \Phi(x, w)) \in R$, and

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

Finally, we call a hard ISP that satisfies all the above properties *enhanced.*

Definition 16. An ISP $\mathcal{I} = (V, M, S)$ is $(t_{\mathsf{ver}}, t_{\mathsf{succ}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-*enhanced* iff it is correct, 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 against precomputation.

Ranges of parameters. 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, here we describe the parameters' ranges that make for a non-trivial enhanced 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 25). 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 $\frac{t_{\mathsf{succ}}}{(1-\delta_{\mathsf{hard}})t_{\mathsf{hard}}}$. Moreover, in order to avoid attacks as the one against sequentially secure primitives, described in Section 3.4, and to ensure that our scheme is parallelizable, we will require α to be smaller than 1.

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

Next, and as already discussed in Section 3, 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. Finally, 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 present later will rely on the fact that an enhanced ISP scheme with favorable parameters exists, mainly reflected in Assumption 1 (Section 4.2.3).

Remark 4. Gentry and Wichs [33] define as falsifiable the cryptographic assumptions that can be expressed as a game between an efficient challenger and an adversary. We note that all assumptions that constitute an enhanced ISP 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 most of the introduced properties, we quantify over all instance descriptions Λ and statements x, which is not immediate to express in the framework of [33]. We could instead first provide as input to the adversary randomly sampled Λ and x, and then have him provide us a statement x' and a sequence of witnesses $(w_i)_i$ such that $S(x, (w_i)_i) = x'$, for which the properties should hold. For simplicity, here we choose the former version of the definitions. However, we note that the proof techniques we use later can be easily adapted to handle the latter.

4.2 The provably secure ISP-based protocol

The main challenge that our protocol has to overcome, is that while its security is based on iterated hardness (Definition 10), it operates in a setting where the adversary can also take advantage of the work of honest parties. This includes, the adversary seeing the information leaked by the honestly produced blocks, as well as, honest parties directly working on the chain it is extending. In contrast, the hard ISP experiment does not give any guarantees about these cases; the adversary does not receive any externally computed witnesses.

Towards this end, blocks in our protocol, instead of exposing the relevant witness computed, 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 counter-intuitive, due to the fact that a trivial simulator and extractor 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 1). Note that in practice this can be achieved by making the ISP hard enough, which however will affect the performance of the resulting ledger being implemented.

In addition, to further reduce the security guarantees required by the ISP, in our protocol the hash chain structure of blocks is decoupled from the underlying computational problem. Finally, the protocol adopts the longest-chain selection rule, which as we will see later allows it to operate even if the witnesses of the ISP are malleable (cf. Remark 3).

4.2.1 Protocol description

Next, we are going to describe our new protocol. Our protocol, $\Pi^{\mathsf{new}}_{\mathsf{PL}}$, uses as building blocks three cryptographic primitives: An enhanced 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[7]

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

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

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 [28], our protocol is parameterized by functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ that capture higher-level applications (such as Bitcoin).

Next, we introduce some notation needed to understand the description of the algorithms (similarly to Section 3.2). 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 head($\mathcal{C}$) (resp., 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*. 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) $s_{i+1} = H(s_i, m_i, x_i, i)$[8] 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 $H(\mathcal{C}) \triangleq H(\text{head}(\mathcal{C}))$. We will consider two valid blocks or chains as equal, if all their parts match, except possibly the NIZK proofs.

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

```
function validate($\mathcal{C}$)
    $b \leftarrow \mathrm{V}(\mathbf{m}_{\mathcal{C}}) \wedge (\text{tail}(\mathcal{C}) = B_{\mathtt{Gen}})$        ▷ $\mathbf{m}_{\mathcal{C}}$ describes the contents of chain $\mathcal{C}$.
    if $b = \text{True}$ then        ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
        $s' \leftarrow H(B_{\mathtt{Gen}})$        ▷ Compute the hash of the genesis block.
        $x' \leftarrow x_{\mathtt{Gen}}$
        $\mathcal{C} \leftarrow \mathcal{C}^{1\rceil}$        ▷ Remove the genesis from $\mathcal{C}$
        while $(\mathcal{C} \neq \epsilon \wedge b = \text{True})$ do
            $\langle s, m, x, \pi\rangle \leftarrow \text{tail}(\mathcal{C})$
            $s'' \leftarrow H(\text{tail}(\mathcal{C}))$
            if $(s = s' \wedge \mathsf{V}^{s''}(\Omega, (\Lambda, x', x), \pi))$ then
                $s' \leftarrow s''$        ▷ Retain hash value
                $x' \leftarrow x$
                $\mathcal{C} \leftarrow \mathcal{C}^{1\rceil}$        ▷ Remove the tail from $\mathcal{C}$
            else
                $b \leftarrow \text{False}$
            end if
        end while
    end if
    return $(b)$
end function
```

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

[8] 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 [8], which in turn implies collision resistance. See Lemma 20.

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 $H(H(head(\mathcal{C})), m', x', |\mathcal{C}|+1)$, where m' is the output of the input contribution function $\mathrm{I}(\cdot)$; see Algorithm 3. Then, the party diffuses the extended 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 backbone protocol. The main function of the protocol is presented in Algorithm 4. 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 $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$.

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

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 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 $w \neq \perp$ 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.
        $B \leftarrow \langle s', m', x', \pi' \rangle$
    end if
    $\mathcal{C} \leftarrow \mathcal{C}B$                                           ▷ Extend chain
    return $\mathcal{C}$
end function
```

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 [28]. For completeness we give these definitions in Table 1. We denote the new public ledger protocol by $\Pi_{\mathsf{PL}}^{\mathsf{new}}$.

4.2.2 Security properties of the blockchain

Next, we describe a number of desired basic properties for the blockchain introduced in [28, 37, 44]. (The informed reader can safely skip this section and proceed to the security analysis.) At a high level, the first property, called *common prefix*, has to do with the existence, as well as persistence in time, of a common prefix of blocks among the chains of honest players. Here we will consider a stronger variant

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())
    $\langle st, m \rangle \leftarrow \mathrm{I}(st, \tilde{\mathcal{C}}, round$, INPUT(), 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
        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
        BROADCAST($\mathcal{C}$)
    end if
    $round \leftarrow round + 1$
    if INPUT() contains READ then
        write $\mathrm{R}(\mathbf{m}_{\mathcal{C}})$ to OUTPUT()
    end if
end while
```

Content validation predicate $\mathrm{V}(\cdot)$	$\mathrm{V}(\cdot)$ 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 transaction tx_0, where tx_0 is a neutral random transaction.

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

of the property, presented in [44], which allows for the black-box proof of application-level properties (such as the *persistence* of transactions entered in a public transaction ledger built on top of the Bitcoin backbone). We will use $\mathcal{C} \preceq \mathcal{C}'$ to denote that some chain $\mathcal{C}$ is a prefix of some other chain $\mathcal{C}'$, and $\mathcal{C}^{\lceil k}$ to denote the chain resulting from removing the last k blocks of $\mathcal{C}$. We will call a block honest, if it was diffused for the first time in the execution by some honest party, and adversarial otherwise.

Definition 17 ((Strong) Common Prefix)**.** The *strong common prefix* property Q_{cp} with parameter $k \in \mathbb{N}$ states that the chains $\mathcal{C}_1, \mathcal{C}_2$ reported by two, not necessarily distinct honest parties P_1, P_2, at rounds r_1, r_2 in $\textsc{view}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, with $r_1 \leq r_2$, satisfy $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.

The next property relates to the proportion of honest blocks in any portion of some honest player's chain.

Definition 18 (Chain Quality)**.** The *chain quality* property Q_{cq} with parameters $\mu \in \mathbb{R}$ and $k \in \mathbb{N}$ states that for any honest party P with chain $\mathcal{C}$ in $\textsc{view}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that for any k consecutive blocks of $\mathcal{C}$ the ratio of adversarial blocks is at most μ.

Further, in the derivations in [28] an important lemma was established relating to the rate at

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
θ :	upper bound on the number of messages sent by the adversary per round
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
β :	upper bound on the rate at which the adversary computes witnesses per step
δ :	advantage from the computing power assumption
k :	number of blocks for the common-prefix property
k_{hard} :	convergence parameter of ISP hardness
ℓ :	number of blocks for the chain-quality property

Table 2: *The parameters in our analysis:* $\lambda, n, t, t_{\mathcal{H}}, t_{\mathcal{A}}, \theta, k, k_{hard}, \ell$ *are in* $\mathbb{N}$, f, γ, β, δ *are in* $(0, 1)$.

which the chains of honest players were increasing as the Bitcoin backbone protocol was run. This was explicitly considered in [37] as a property under the name *chain growth.*

Definition 19 (Chain Growth)**.** The *chain growth* property Q_{cg} with parameters $\tau \in \mathbb{R}$ (the "chain speed" coefficient) and $s, r_0 \in \mathbb{N}$ states that for any round $r > r_0$, where honest party P has chain $\mathcal{C}_1$ at round r and chain $\mathcal{C}_2$ at round $r + s$ in $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that $|\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau \cdot s$.

4.2.3 Security analysis

In this section we prove that $\Pi^{\mathsf{new}}_{\mathsf{PL}}$ implements a robust public transaction ledger (cf. Definition 5), assuming the underlying ISP $\mathcal{I}$ is $(t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-enhanced; $t'_{\mathcal{H}}$ is related to our model and we formally define it in the next paragraph. Further, we assume that running the prover (resp., verifier, simulator, extractor) of Π_{NIZK} takes t_{P} (resp. $t_{\mathsf{V}}, t_{\mathsf{S}}, t_{\mathsf{E}}$) steps. As we have already hinted, we are going to make concrete assumptions about the proof-system parameters. Next, we introduce some additional notation, and the computational assumption under which we prove security.

For brevity, and to better connect our analysis to previous work [28, 44], we denote by $\beta = \frac{1}{(1-\delta_{\mathsf{hard}}) \cdot t_{\mathsf{hard}}}$, the upper bound on the rate at which the adversary can compute witnesses in the iterated hardness game. Next we introduce two variables, $t'_{\mathcal{H}}$ and $t'_{\mathcal{A}}$, that have to do with the effectiveness of honest parties and any adversary against our protocol 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 thus 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 execution of the protocol, without taking into account calls made to M by the latter, and also run 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[9], besides the calls to $M, \mathsf{P}, \mathsf{V}$.

We are now ready to state our main computational assumption regarding the honest parties and the adversary. Previous analysis of PoW-based blockchain protocols in the RO model [28, 44] assumed

[9] Note that t_{bb} depends on the running time of three external functions: $\mathrm{V}(\cdot), \mathrm{I}(\cdot)$ and $R(\cdot)$. For example, in Bitcoin these functions include the verification of digital signatures, which would require doing modular exponentiations. In any case, t_{bb} is at least linear in λ.

two things: First, that the total number of steps the honest parties take per round exceed those of the adversary, and second, that the total block generation rate is bounded. These are enough in an idealized setting, as the efficiency of the honest PoW algorithm is the same as that of the adversary. In contrast, in our case 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 $\beta t'_{\mathcal{H}}$. Next, we formalize this assumption. The first condition corresponds to the observation we just made, while the other two correspond to adaptations of the older assumptions. 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 1 (**Computational Power Assumption.**). Let $\delta_{\mathsf{ISP}}, \delta_{\mathsf{Steps}}$ and $\delta \in (0,1)$. It holds that

- $\alpha \geq (1-\delta_{\mathsf{ISP}})\beta t'_{\mathcal{H}}$ (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 the hash functions used by the protocol and the necessary security assumptions to avoid cycles in the blockchains. Observe that the hash structure of any blockchain in our protocol is similar to the Merkle-Damgard transform [20], defined as:

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

Based on this observation, we can show that no efficient adversary can find distinct chains with the same hash value.

Lemma 20. *For any PPT RAM $\mathcal{A}$, the probability that $\mathcal{A}$ 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.*

Proof. To show that the adversary cannot find distinct chains with the same hash, we are going to take advantage of the following property of the MD transform: For any non-empty valid chain $\mathcal{C} = B_1, \ldots, B_k$, where $B_i = \langle s_i, m_i, x_i, \pi_i \rangle$, it holds that for any $j \in [k]$, $H(\mathrm{head}(\mathcal{C})) = \mathrm{MD}(H(B_j), ((m_i, x_i, i))_{i\in\{j+1,\ldots,k\}})$ Let $\mathcal{C}_1 = B_{\mathtt{Gen}}, B_1, \ldots, B_{|\mathcal{C}_1|}$, $\mathcal{C}_2 = B_{\mathtt{Gen}}, B'_1, \ldots, B'_{|\mathcal{C}_2|}$, $z = ((m_i, x_i, i))_{i\in[|\mathcal{C}_1|]}$ and $z' = ((m'_i, x'_i, i))_{i\in[|\mathcal{C}_2|]}$. For the sake of contradiction, assume that the lemma does not hold and there exists an adversary $\mathcal{A}$ that can find valid chains $\mathcal{C}_1$, $\mathcal{C}_2$ such that $H(\mathcal{C}_1) = H(\mathcal{C}_2)$, with non-negligible probability. By our observation above, this implies that $\mathrm{MD}(H(B_{\mathtt{Gen}}), z) = \mathrm{MD}(H(B_{\mathtt{Gen}}), z')$.

We will construct an adversary $\mathcal{A}'$ that breaks the collision resistance of H with non-negligible probability. We have two cases. In the first case, $|\mathcal{C}_1| \neq |\mathcal{C}_2|$. Then, since the height of the chain is part of the hash of blocks $B_{|\mathcal{C}_1|}, B'_{|\mathcal{C}_2|}$, and $H(B_{|\mathcal{C}_1|}) = H(\mathrm{head}(\mathcal{C}_1)) = H(\mathrm{head}(\mathcal{C}_2)) = H(B_{|\mathcal{C}_2|})$, it follows that a collision in H has been found. In the second case, where $|\mathcal{C}_1| = |\mathcal{C}_2|$, following the classical inductive argument for the MD transform, it can be shown that either $\mathcal{C}_1$ and $\mathcal{C}_2$ are equal, or there exists $\ell \in [|\mathcal{C}_1|]$, such that $\mathrm{MD}(H(B_{\mathtt{Gen}}), ((m_i, x_i, i))_{i\in[\ell]}) = \mathrm{MD}(H(B_{\mathtt{Gen}}), ((m'_i, x'_i, i))_{i\in[\ell]})$ and $(m_\ell, x_\ell) \neq (m'_\ell, x'_\ell)$, which implies again that a collision has been found for H. Since the probability that a collision can be found for H is negligible, the lemma follows. □

The following two properties, introduced in [28], regarding the way blocks are connected are implied by Lemma 20.[10]

[10] A third property, called "prediction," also introduced in [28], 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.

Definition 21. 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.

Corollary 22. *Let $\mathcal{H}$ be a collision-resistant hash functions family. Then, no insertions and no copies occur with probability* $1 - \mathsf{negl}(\lambda)$.

Next, we prove that the adversary cannot mine blocks extending a single chain, with rate and probability better than that of breaking the iterative hardness property. 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 $Z_B(R)$ 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. Define $X_B(R)$ similarly.

Lemma 23. *For any set of consecutive rounds R, where $|R| \geq \frac{k_{\mathsf{hard}}}{\beta t'_{\mathcal{A}}}$, and for any party P, the probability that P mined some honest block B at some round $i \in R$ and $Z_B(R) \geq \beta t'_{\mathcal{A}} |R|$, is* $\mathsf{negl}(\lambda)$.

Proof. 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 $\text{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}}, \mathcal{A}, \mathcal{Z}}$ 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 against precomputation (Definition 10) 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 will 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.

Next, we describe the behavior of $\mathcal{A}'$ in more detail. We are going to describe the two stages of $\mathcal{A}'$ separately, i.e. before and after obtaining x. First, $\mathcal{A}'_1(\Lambda)$ sets $(\Lambda, x_{\mathtt{Gen}}, H, \Omega)$ as the common input for $\mathcal{A}$ and $\mathcal{Z}$, where Ω has been generated using S_1 and the rest of the input using the default samplers, and stores the NIZK trapdoor tk. Then, it perfectly simulates honest parties up to round $i - 1$ and at the same time runs $\mathcal{A}$ and $\mathcal{Z}$ in a black-box way. Finally, it outputs the contents of the registers of $\mathcal{A}$ and $\mathcal{Z}$ and the NIZK trapdoor tk, as variable st. It can do all this, since in the iterated hardness against precomputation experiment it has polynomial time on λ on his disposal. Note, that up until this point in the eyes of $\mathcal{A}$ and $\mathcal{Z}$ the simulated execution is perfectly indistinguishable compared to the real one.

For the second stage, $\mathcal{A}'_2(st, x)$, is first going to use st to reset $\mathcal{A}$ and $\mathcal{Z}$ to the same state that they were. We assume that this can be done efficiently, e.g., by having $\mathcal{A}$ and $\mathcal{Z}$ read from the registers where st is stored whenever they perform some operation on their registers. It will also continue to simulate honest parties, this time in a more efficient way.

$\mathcal{A}'_2$ takes as input a problem statement x sampled from X, as in Definition 10. It should somehow introduce x to the simulated protocol execution, without the adversary noticing any difference that could help him distinguish from the real execution. Let $B_0 = \langle s_0, m_0, x_0, \pi_0 \rangle$ be the head of chain $\mathcal{C}$ that party P is extending at round i, and m_1 the block input it produced for this round using the input contribution function I$(\cdot)$. $\mathcal{A}_2'$ is first going to run M on input x for the amount of steps available to P. If it is not successful, it is going to perfectly simulate the rest of the execution as the protocol dictates. Otherwise, if it is successful and produces some witnesses w, it will broadcast the following block:

$$B_1 = \langle H(B_0), m_1, S(x,w), \mathsf{S}_2^{H(H(B_0),m_1,S(x,w),|\mathcal{C}|+1)}(\Omega, (\Lambda, x_0, S(x,w)), tk)\rangle$$

where the last component is a simulated NIZK proof for the statement $(x_0, S(x,w))$. Note, that $\mathcal{A}_2'$ does not know any witness for this statement, and it is possible that no such witness exists. Later, we will argue that the output of the simulator on this input should be indistinguishable from the output on the statement $(x_0, S(x_0, M(x_0)))$. Also, note that due to the next-problem simulatability property, $\mathcal{A}_2$ will not be able to tell the difference of P running M on x_0 or x at this round.

$\mathcal{A}_2'$ will follow a more complex strategy to simulate the rest of the honest parties invocations. For each honest party, it will run the next-problem simulator $\Psi(1^\lambda)$ and check if the numbers of steps output is less than the number of steps available on this invocation. If they are not, $\mathcal{A}_2'$ will proceed by just updating the state of this party for the round. Otherwise, it will simulate its behavior when being successful, as follows: Let block $B^* = \langle s^*, m^*, x^*, \pi^*\rangle$ be the head for the chain $\mathcal{C}^*$ the honest party was trying to extend with message m'' in this round. Let $B_j = \langle s_j, m_j, x_j, \pi_j\rangle$ be the adversarial block that descends B_1 and maximizes the number of adversarial blocks between itself and B_1. Let $B' = \langle s', m', x', \pi'\rangle$ be the parent of B_j. If no such adversarial block exists, assume that $B_j = B_1$ and $B' = \langle \emptyset, x, w, \emptyset\rangle$. $\mathcal{A}_2'$ first runs the NIZK extractor $\mathsf{E}^{H(B_j)}(\Omega, ((\Lambda, x', x_j), \pi_j), tk)$ to obtain a witness w' for x'. Then, it runs $\Phi(x', w')$ and obtains a new witness w'' for x'; let $x'' = S(x', w'')$. Finally, it is going to make $\mathcal{A}_2$ believe that the block it has computed extends B^*, instead of B', by simulating a NIZK proof as follows: $\pi'' = \mathsf{S}_2^{H(H(B^*),m'',x'',|\mathcal{C}^*|+1)}(\Omega, (\Lambda, x^*, S(x', w'')), tk)$. The new block that $\mathcal{A}_2'$ is going to diffuse is $\langle H(B^*), m'', x'', \pi''\rangle$. We point to Figure 1 for an example of the procedure described above.

In the following claim we argue that the view H_{sim} of the adversary in the simulated run we just described is computationally indistinguishable from its view H_0 in $\text{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$.

Claim 1. $H_{sim} \overset{c}{\approx} H_0$.[11]

Proof. We start by describing a sequence of hybrids:

- Hybrid H_0: The view of the adversary in $\text{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$.
- Hybrid H_1: Same as H_0, with the only difference being replacing honest parties' calls to P by calls to S_2, and Ω being generated by S_1.
- Hybrid $H_{1,i}$ to $H_{n,s}$: In hybrid $H_{u,v}$, we replace the next statement and the NIZK in the block produced by party u at round v if successful, with a possibly wrong statement and proof computed as described in the proof above.

By the zero knowledge property of the NIZK proof system it easily follows that H_0 is indistinguishable from H_1; H_0 corresponds to the real execution, while H_1 to the simulated one.

Next, we will argue that $H_{u-1,v}$ is indistinguishable from $H_{u,v}$, for some $u \in [n], v \in R$ (let $H_{0,i} = H_1$), by contradiction . Assume $H_{u-1,v} \overset{c}{\not\approx} H_{u,v}$. There are three cases. In the first case, party u at round v is not successful. Obviously, in this case nothing changes in the view of the adversary, and the two executions are indistinguishable. In the second case, $u = P$, $v = i$ and P is successful. In $H_{P,i}$, instead of $S(x_0, M(x_0))$, the next problem will be $S(x, M(x))$, and the NIZK will be changed accordingly. Assuming that the two hybrids are distinguishable, by an averaging argument, we can

[11] For brevity, we abuse notation here and use the computational indistinguishability relation to sequences of random variables, instead of random variables ensembles. The related random variable ensembles can be easily deduced.

construct a PPT distinguisher D that distinguishes $(x_0, S(X, M(X)))$ from $(x_0, S(x_0, M(x_0))$, for some x_0. This is a contradiction, since by the next-problem simulatability property it follows that:

$$(x_0, S(x_0, M(x_0))) \overset{c}{\approx} (x_0, \Psi(1^\lambda)) \overset{c}{\approx} (x_0, S(X, M(X)))$$

where the last part follows from the fact that $\Psi(1^\lambda)$ does not depend on the problem statement. For our final case, assume that either $u \neq P$ or $v \neq i$ and u is successful. Similarly, we can arrive to a contradiction due to the witness malleability property, by the fact that for any $x, x' \in X, w' \in W$ where $(x', w') \in R$ it holds that:

$$(x, S(x, M(x))) \overset{c}{\approx} (x, \Psi(1^\lambda)) \overset{c}{\approx} (x, S(x', M(x'))) \overset{c}{\approx} (x, S(x', \Phi(x', w')))$$

If the second and third distributions are distinguishable, then we can construct a distinguisher for the next-problem simulatable property, while if the third and the fourth distributions are distinguishable, we can construct a distinguisher for the witness malleability property.

Finally, note that the running time of simulated parties, is computationally indistinguishable from the running time parties take when running M, again from the next-problem simulatable property. The claim follows by the fact that $H_{n,s}$ is the same as H_{sim}. $\dashv$

Since $\mathcal{A}$ and $\mathcal{Z}$ cannot distinguish between the real execution and the one we described above, E will occur with non-negligible probability in H_{sim}, i.e. $\mathcal{A}$ will compute at least $\beta t'_{\mathcal{A}} s$ blocks starting from round i and up to round $i + s$ that descend B_1 and lie on the same chain. By the way honest blocks are constructed, $\mathcal{A}'_2$ knows the witnesses of the honest blocks in this chain, and using the NIZK extractor it can extract the witnesses of the adversarial ones. Now, note that each adversarial block includes a witness to the problem statement defined by the previous block, while at the same time each subsequent honest block defines a problem statement that lies in a sequence starting from x and followed by at least as many witnesses as on the previous block. It follows that $\mathcal{A}'_2$ can extract a sequence of valid witnesses of length at least $\beta t'_{\mathcal{A}} s + 1$, where the plus one comes from the witness computed by P at round i, and win in the iterative hardness game with non-negligible probability while taking at most

$$t_{\mathcal{H}} + s \cdot (t_{\mathcal{A}} + \theta \cdot t_{\mathsf{V}} + t_{\mathsf{E}}) + s \cdot n(t_{\mathsf{bb}} + t_{nps} + t_{mal} + t_{\mathsf{S}}) \leq s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}}$$

steps. Hence, $\mathcal{A}'_2$ has computed $\beta t'_{\mathcal{A}} s + 1 \geq \beta(s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}}) \geq k_{\mathsf{hard}}$ blocks in $s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}} = (1 - \delta_{\mathsf{hard}}) t_{\mathsf{hard}} \cdot \beta(s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}})$ steps with non-negligible probability. This is a contradiction to our initial assumption that $\mathcal{I}$ is a $(t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-hard ISP. □

Note that 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 24. *For any set of consecutive rounds R, where $|R| \geq \frac{k_{\mathsf{hard}}}{\beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}})}$, and for any party P, the probability that P mined some honest block B during the first round of R and $Z_B(R) + X_B(R) > \beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}}) \cdot |R|$, is $\mathsf{negl}(\lambda)$.*

Next, we prove lower bounds on the rate of successful and uniquely successful rounds. Our proof crucially depends on the next-problem simulatable property of $\mathcal{I}$. Specifically, the fact that the runtime simulator $\Psi(1^\lambda)$ does not depend on the problem statement, implies that the steps honest parties take running M at each round will be indistinguishable from a set of i.i.d random variables following the distribution defined by $\Psi(1^\lambda)$. Hence, for some set of rounds the sum of the Bernoulli random variables

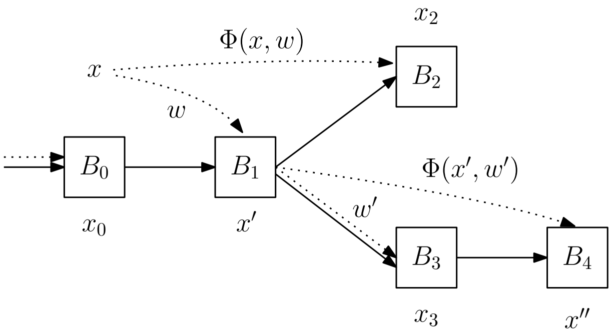

Figure 1: *A possible scenario according to Lemma 23. The blocks have been generated in order B_0, B_1, B_2, B_3, B_4, with B_3 being the only adversarial block. The cases where a valid witness is either known or can be extracted, and a NIZK proof has either been computed or simulated for the depicted transitions, correspond to the dotted and normal arrows, respectively.*

of the event that a round is successful or uniquely successful, which only depends on the running time of M, will necessarily have good concentration properties. In turn, this implies that we can lower-bound the rates of successful and uniquely successful rounds with good probability.

Lemma 25. *Let $\gamma = (n-t)\cdot\alpha(1-\beta t_{\mathcal{H}})^{n-1}$, $f = (1-(1-\alpha)^{n-t})$. For any set of consecutive rounds R, with $|R| \geq \frac{\lambda}{\gamma\delta^2}$, the following two events occur with negligible probability in λ:*

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

Proof. Our proof strategy will be as follows: first we will prove the lemma assuming that the running time of different invocation of M from honest parties, follows the distribution implied by the simulator Ψ in the next-problem simulatable property. Note, that the property dictates that this distribution is computationally indistinguishable from the real one. Then, we will show, that the result also holds for the real execution.

For some fixed execution we will denote by the array $\mathbf{T}_{R\times n} = (t_{i,j}) \in \mathbb{N}^{|R|\times n}$ the number of steps each honest party takes running M, for each round in the set R. It holds that at most t elements of each column are zero, i.e. corrupted, and the rest are lower bounded by $t'_{\mathcal{H}}$ and upper bounded by $t_{\mathcal{H}}$. W.l.o.g let $R = \{1,\ldots,s\}$.

We first introduce some additional notation to help in our analysis. Consider the following random variables:

- $P_{i,j} = 1$ if $\Psi(1^\lambda) \leq t_{i,j}$, and 0 otherwise. Ψ is invoked with independent random coins, for each i,j.
- $Y_i = 1$ if $\sum_{j=1}^{n} P_{i,j} = 1$, and 0 otherwise.
- $X_i = 1$ if $\sum_{j=1}^{n} P_{i,j} \geq 1$, and 0 otherwise.
- $Y = \sum_{i\in[s]} Y_i$, $X = \sum_{i\in[s]} X_i$.

Note that the output of Ψ neither depends on the input of M or the problem description Λ. Hence, the random variables in $\{P_{i,j}\}_{(i,j)\in[s]\times[n]}$ are mutually independent. Which further implies that the random variables in $\{Y_i\}_{i\in[s]}$ and $\{X_i\}_{i\in[s]}$ are mutually independent, respectively.

Next, we derive lower and upper bounds for $\Psi(1^\lambda)$. First, from the Successful property it follows that $\Pr[\Psi(1^\lambda) \leq t'_{\mathcal{H}}] \geq \alpha - \mathsf{negl}(\lambda)$. Otherwise, we can construct a distinguisher for $\Psi(1^\lambda)$ and $\mathsf{Steps}_M(x)$, for any Λ, x, by checking whether the input to the distinguisher is smaller than $t'_{\mathcal{H}}$. This violates the next-problem simulatable property. Similarly, we can upper bound $\Pr[\Psi(1^\lambda) \leq t_{\mathcal{H}}]$.

Claim 2. $\Pr[\Psi(1^\lambda) \leq t_{\mathcal{H}}] \leq t_{\mathcal{H}}\beta + \mathsf{negl}(\lambda)$.

Proof. For the sake of contradiction, assume that the difference $\Pr[\Psi(1^\lambda) \leq t_\mathcal{H}] - t_\mathcal{H}\beta$ is non-negligible. First, we will argue that there exists an $x \in X$, such that $\Pr[\mathsf{Steps}_M(x)] \leq t_\mathcal{H}\beta$. For the sake of contradiction, assume that for all $x \in X$, $\Pr[\mathsf{Steps}_M(x)] > t_\mathcal{H}\beta$. By the iterated hardness property, we have that for $k \geq k_0$, any k/β-bounded adversary will compute k or more witnesses with negligible probability in λ (assume we pick a k that is a multiple of β). This implies that the expected number of blocks any such adversary computes is at most k. Let an adversary that is based on M work as follows: on some initial input x, it runs M for at most $t_\mathcal{H}$ steps. If, it succeeds on producing a witness, it computes the next problem, and runs M again with the new input. If not, it runs M on the initial input. By our assumption and the linearity of expectation, the expected number of blocks our adversary will mine on k/β steps, is greater than

$$t_\mathcal{H}\beta \cdot k/\beta t_\mathcal{H} = k.$$

This is a contradiction. Hence, there exists exists an $x_0 \in X$, such that $\Pr[\mathsf{Steps}_M(x_0)] \leq t_\mathcal{H}\beta$. This in turn implies that we can construct a distinguisher for $\Psi(1^\lambda)$ and $\mathsf{Steps}_M(x_0)$, by checking whether the input of the distinguisher is less or equal to $t_\mathcal{H}$. This is a contradiction to the next-problem simulatable property. Therefore, the claim follows. ⊣

By the above two bounds, and assuming party j at round i is not corrupted, it follows that

$$\alpha - \mathsf{negl}(\lambda) \leq \Pr[P_{i,j} = 1] \leq t_\mathcal{H}\beta + \mathsf{negl}(\lambda).$$

For simplicity, in the subsequent calculations, we ignore the negligible terms. Next, we lower-bound the expected value of random variables $(Y_i)_i$ and $(X_i)_i$.

Claim 3. *It holds that for any $i \in R : \mathbb{E}[Y_i] \geq \gamma$.*

Proof.

$$\begin{aligned}
\mathbb{E}[Y_i] =& \Pr[Y_i = 1] = \Pr[\sum_{j\in[n]} P_{i,j} = 1] \\
=& \sum_{j\in[n]} \Pr[P_{i,j} = 1] \cdot \prod_{m\in[n]\setminus\{j\}} \Pr[P_{i,m} = 0] \\
\geq& \sum_{j\in[n]} \alpha \prod_{m\in[n]\setminus\{j\}} (1 - \Pr[P_{i,m} = 1]) \\
\geq& (n-t)\alpha(1 - t_\mathcal{H}\beta)^{n-1} = \gamma
\end{aligned}$$

where in the second inequality we have used the fact the respective random variables are independent. ⊣

Claim 4. *It holds that for any $i \in R : \mathbb{E}[X_i] \geq f$.*

Proof.

$$\begin{aligned}
\mathbb{E}[X_i] =& \Pr[X_i = 1] = \Pr[\sum_{j\in[n]} P_{i,j} \geq 1] \\
=& 1 - \Pr[\sum_{j\in[n]} P_{i,j} = 0] \\
=& 1 - \prod_{j\in[n]} \Pr[P_{i,j} = 0] \\
\geq& 1 - (1-\alpha)^{n-t} = f
\end{aligned}$$

⊣

By the linearity of expectation we have that $\mathbb{E}[Y(R)] \geq \gamma|R|$ and $\mathbb{E}[X(R)] \geq f|R|$. Since the random variables in $\{Y_i\}_i$ are mutually independent, and $\delta \in (0,1)$, by an application of the Chernoff Bound, we have that:

$$\Pr[Y(R) \leq (1-\frac{\delta}{4})\gamma|R|] \leq \Pr[Y(R) \leq (1-\frac{\delta}{4})\mathbb{E}[Y(R)]] \leq e^{-\Omega(\delta^2\gamma|R|)}.$$

Similarly, $\Pr[X(R) \leq (1-\frac{\delta}{4})f|R|] \leq e^{-\Omega(\delta^2 f|R|)}$.

Finally, using similar arguments as in Lemma 23, it follows that the view of the adversary in the real execution, and its view on an execution where the honest parties are simulated using the simulators Ψ and S of the next-problem simulatable property and the NIZK, respectively, are computationally indistinguishable. For the latter execution, we have already established that the conditions of the lemma hold. Hence, since these conditions can be checked in polynomial time, it follows that they should also hold for the real execution, with negligible difference in probability. The lemma follows. □

In addition, we show that γ is sufficiently bigger than two times $\beta \cdot t'_{\mathcal{A}}$.

Lemma 26. $\gamma \geq 2(1+\delta)\beta t'_{\mathcal{A}}$.

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 1 (the Computational Power Assumption). The last inequality follows from the fact that $\frac{\delta_{\mathsf{Steps}}-\delta_{\mathsf{ISP}}}{2} \geq \delta$. □

Following the strategy of [28], we are now ready to define the set of *typical executions* for this setting.

Definition 27 (Typical execution). An execution is *typical* if and only if for any set R of consecutive rounds with $|R| \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$, the following hold:

1. $Y(R) > (1-\frac{\delta}{4})\gamma|R|$ and $X(R) > (1-\frac{\delta}{4})f|R|$;
2. for any block B mined by an honest party at some round in R, $Z_B(R) < \frac{\gamma}{2(1+\delta)}\cdot|R|$ and $Z_B(R) + X_B(R) < \beta(t'_{\mathcal{A}}+nt'_{\mathcal{H}})\cdot|R|$; and
3. no insertions and no copies occurred.

Theorem 28. *An execution of* $\Pi^{\mathsf{new}}_{\mathsf{PL}}(\mathcal{I})$ *is typical with probability* $1-\mathsf{negl}(\lambda)$.

Proof. In order for an execution to not be typical one of the three points of Definition 27 must not hold. For any set of rounds R, such that $|R| \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$, it holds that point 1 of Definition 27 is implied by Lemma 25 and point 2 is implied by Lemma 23 and Corollary 24. Lastly, point 3 is implied by Corollary 22. Note, that w.l.o.g., for Lemma 23 we assume an adversary with power $t'_{\mathcal{A}} = \frac{\gamma}{\beta\cdot 2(1+\delta)}$, and thus $\frac{k_{\mathsf{hard}}}{\beta t'_{\mathcal{A}}} \leq \frac{4k_{\mathsf{hard}}}{\gamma\delta^2}$. Moreover, for Corollary 24 we can easily show that $\frac{k_{\mathsf{hard}}}{\beta(t'_{\mathcal{A}}+nt'_{\mathcal{H}})} \leq \frac{k_{\mathsf{hard}}}{\gamma}$. Finally, we can bound the probability that an execution is not typical by applying the union bound on the negation of these events over all sets of consecutive rounds of appropriate size. By the fact that the probability of each of these events is negligible in λ, and that the number of the events is polynomial in λ, the theorem follows. □

Next, we prove that the rate at which the adversary generates blocks in any big enough round interval, is at most half the rate of uniquely successful rounds. This relation is going to be at the center of the security proof we are going to develop next.

Lemma 29. *For any set of rounds R in a typical execution, where $|R| \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$, and for any block B mined by an honest party during R, it holds that $Z_B(R) < (1-\frac{\delta}{4})\frac{Y(R)}{2}$.*

Proof. For any big enough set of rounds R it holds that:

$$Z_B(R) < \frac{\gamma}{2(1+\delta)}|R| \leq (1-\frac{\delta}{4})(1-\frac{\delta}{4})\frac{\gamma}{2} \leq (1-\frac{\delta}{4})\frac{Y(R)}{2}$$

where, the first and the last inequality follow from the assumption that the execution is typical. □

Finally, we can use the machinery built in [28] to prove the Common Prefix, Chain Quality and Chain Growth properties, with only minor changes.

Higher level properties. Next, we describe the changes one has to do after proving the typical execution theorem with respect to the analysis of [28], in order to prove the security of the protocol in our model. We only give brief proof sketches of lemmas and theorems from [28] that are exactly the same for our own setting.

Lemma 30 (Chain Growth Lemma)**.** *Suppose that at round r an honest party has a chain of length ℓ. Then, by round $s \geq r$, every honest party has adopted a chain of length at least $\ell + \sum_{i=r}^{s-1} X_i$.*

Proof. The main idea of the proof of this lemma is that, after each successful round at least one honest party will have received a chain that is at least one block longer than the chain it had, and all parties pick only chains that are longer than the ones they had. □

Theorem 31 (Chain Growth)**.** *In a typical execution the chain-growth property holds with parameters $\tau = (1-\frac{\delta}{4})f$ and $s \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$.*

Proof. Let R be any set of at least s consecutive rounds. Then, since the execution is typical: $X(R) \geq (1-\frac{\delta}{4})f \cdot |R| \geq \tau \cdot |R|$. By Lemma 30, each honest player's chain will have grown by that amount of blocks at the end of this round interval. Hence, the chain growth property follows. □

Lemma 32. *Let B be some honest block in a typical execution. Any sequence of $k \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$ consecutive blocks in some chain $\mathcal{C}$, where the first block in the sequence directly descends from B, have been computed in at least k/δ rounds, starting from the round that B was computed.*

Proof. For some $k \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$, assume there is a set of rounds R' , such that $|R'| < k/\delta$, and more than k blocks that descend from block B have been computed. Then, there is a set of rounds R, where $|R| \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$, such that $X_B(R) + Z_B(R) \geq k \geq |R|\delta > |R|\beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}})$, where the last inequality follows from Assumption 1. This contradicts the typicality of the execution, hence the lemma follows. □

Lemma 33 (Common Prefix Lemma)**.** *Assume a typical execution and consider two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $len(\mathcal{C}_2) \geq len(\mathcal{C}_1)$. If $\mathcal{C}_1$ is adopted by an honest party at round r, and $\mathcal{C}_2$ is either adopted by an honest party or diffused at round r, then $\mathcal{C}_1^{\lceil k} \leq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \leq \mathcal{C}_1$, for $k \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$.*

Proof. The proof in [28] shows that for every block mined at a uniquely successful round, there exists an adversarial block in one of the two chains. This in turn implies that one of the two chain has a number of adversarial blocks that is at least as big as half the number of uniquely successful rounds. Using the previous lemma the proof proceeds as in [28], reaching a contradiction with Lemma 29. Note, that all adversarial blocks in the matching between uniquely successful rounds and adversarial blocks are descendants of the last honest block in the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. □

Theorem 34 (Common Prefix)**.** *In a typical execution the common-prefix property holds with parameter* $k \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$.

Proof. The main idea of the proof is that if there exists a deep enough fork between two chains, then the previously proved lemma cannot hold. Hence, the theorem follows. □

Theorem 35 (Chain Quality)**.** *In a typical execution the chain-quality property holds with parameter* $\mu < 1 - \delta/4$ *and* $\ell \geq \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$.

Proof. The main idea of the proof is the following: a large enough number of consecutive blocks will have been mined in a set rounds that satisfies the properties of Definition 27. Hence, the number of blocks that belong to the adversary will be upper bounded, and all other blocks will have been mined by honest parties. □

Finally, the Persistence and Liveness properties follow from the three basic properties, albeit with different parameters than in [28].

Lemma 36 (Persistence)**.** *It holds that* $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ *with* $k = \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$ *satisfies Persistence with overwhelming probability in* λ.

Proof. The main idea is that if persistence is violated, then the common-prefix property will also be violated. Hence, if the execution is typical the lemma follows. □

Lemma 37 (Liveness)**.** *It holds that* $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ *with* $k = \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$ *and* $u = \frac{2k}{(1-\frac{\delta}{4})f}$ *rounds satisfies Liveness with overwhelming probability in* λ.

Proof. The main idea here is that after u rounds at least $2k$ successful rounds will have occurred in a typical execution. Thus, by the chain growth lemma the chain of each honest party will have grown by $2k$ blocks, and by the chain quality property at least one of these blocks that is deep enough in the chain is honest. □

Hence, protocol $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ implements a robust transaction ledger. Note that both Persistence and Liveness depend on the convergence parameter k_{hard} of $\mathcal{I}$.

Theorem 38. *Assuming the existence of a collision-resistant hash function family, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and an enhanced ISP problem* $\mathcal{I}$ *that comply with Assumption 1, protocol* $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ *implements a robust public transaction ledger with parameters* $k = \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$ *and* $u = \frac{2k}{(1-\frac{\delta}{4})f}$*, except with negligible probability in* λ.

4.3 An Enhanced ISP Construction

We now present an ISP problem that is plausibly hard against precomputation (Definition 10), and satisfies *all* other properties of an enhanced ISP (Definition 16). Our construction is based on a hash function that is both collision resistant and a computational randomness extractor. Similar assumptions have been made in the analysis of HMAC regarding the underlying compression function [22, 38]. Next, we give a formal definition of the properties our hash function should satisfy.

Definition 39. For some $c, d \in \mathbb{N}^+$, where $c < d$, let $\mathcal{H} = \{\{H_k : \{0,1\}^{d\lambda} \to \{0,1\}^\lambda\}_{k \in K(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a hash-function family. $\mathcal{H}$ is a *computational collision-resistant randomness extracting* (C-CRRE) hash family if

- $\mathcal{H}$ is collision resistant, and
- for sufficiently large $\lambda \in \mathbb{N}$, for each $k \in K(\lambda)$, $E(x, i) \overset{\triangle}{=} H_k(x||i)$ is a computational $(c\lambda - \lambda, \mathsf{negl}(\lambda))$-randomness extractor, where $|x| = c\lambda$.

A related concept, proposed by Dodis in [21], is that of a (statistical) collision-resistant randomness extractor, which can be constructed assuming the existence of claw-free permutations. For this construction, in order for collision resistance to hold, it is required that the seed of the extractor is fixed. On the other hand, randomness extraction is only guaranteed for sources which are independent of the seed. This is problematic in our case, since the seed is part of the description of the hash function and should be public. Thus, we cannot ensure that the source distribution will be independent of the seed, and cannot use this construction in our setting.

We are now ready to describe our ISP construction.

Construction 2. Let $\mathcal{H}$ be a hash function family as in Definition 39. Let $T \in \{0,1\}^\lambda$ be a hardness parameter. An instance of an enhanced ISP is as follows:

- I_λ is the uniform distribution over $K(\lambda)$, i.e., $\Lambda = \{k\}$;
- $X = \{0,1\}^\lambda, W = \{0,1\}^{2(d-1)\lambda}$;
- $R = \{(x, w) | H_k(x||w_1) < T \text{ for } w = w_1||w_2\}$;
- $M(x, 1)$ 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 $w_1||w_2$.
- $S(x, w) = H_k(H_k(x||w_1)||w_2)$.

Construction 2 is similar to Bitcoin's PoW construction (see Section 3.1, 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 some 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 an enhanced 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 distribution of M is indistinguishable from the geometric distibution with parameter $T/2^\lambda$. This implies the successful

ISP property. 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 w_2 uniformly at random, the witness malleability property follows. We are thus able to state:

Lemma 40. *If $\mathcal{H}$ is a C-CRRE hash family, then Construction 2 is $O(\lambda)$-next-problem simulatable, $O(\lambda)$-witness malleable, and $(t, \mathcal{C}_{T/2^\lambda}(O(t)) - \mathsf{negl}(\lambda))$-successful for any $t \in poly(\lambda)$, where $\mathcal{C}_{T/2^\lambda}$ is the cumulative geometric distribution with parameter $T/2^\lambda$.*

Proof. We start by defining some notation that we will use in the proof. Let c, d be as in Definition 39. For any sufficiently large $\lambda \in \mathbb{N}$, fix some $k \in K(\lambda)$ and $x \in X$. Let random variable $Z = H_k(x||\mathcal{U}_{c\lambda}||\mathcal{U}_{(d-c-1)\lambda})$. By our assumption, H_k is a $(c\lambda, \mathsf{negl}(\lambda))$-extractor. Moreover, $x||\mathcal{U}_{c\lambda}$ has $c\lambda$ bits of min-entropy. It follows that $Z \overset{c}{\approx} \mathcal{U}_\lambda$. Now, assume instead that x is sampled from some distribution $\hat{\mathcal{X}}$, as it will be the case in an actual execution, and let $\hat{Z}$ be the random variable produced by first sampling x from $\hat{\mathcal{X}}$, and then applying $H_k(x||\mathcal{U}_{c\lambda}||\mathcal{U}_{(d-c)\lambda})$. By an averaging argument it follows that there exists some x such that $\hat{Z} \overset{c}{\approx} Z$, which in turn implies that $\hat{Z} \overset{c}{\approx} \mathcal{U}_\lambda$.

We next argue about the distribution of the running time of M. Algorithm M on input x iteratively samples a uniformly random $w_1||w_1'$ from $\mathcal{U}_{c\lambda} \times \mathcal{U}_{(d-c-1)\lambda}$, and tests whether $H_k(x||w_1||w_1') < T$, until it finds a solution. For a moment, assume that M instead tested whether a value sampled from $\mathcal{U}_\lambda$ is smaller than T. Then, its running time would be distributed according to the geometric distribution $\mathcal{G}_p$ with parameter $p = T/2^\lambda$. Since $Z \overset{c}{\approx} \mathcal{U}_\lambda$, and by an application of a hybrid argument, the distribution of $\mathsf{Steps}_M(x)$ should be computationally indistinguishable from $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$, where c_1 is a constant related to the cost of sampling a random value for each test and evaluating H, and c_2 to the cost of sampling w_2. The hybrid argument proceeds by replacing a computation of $H(x||\mathcal{U}_{(d-1)\lambda}) < T$ at some step of M, with $\mathcal{U}_\lambda < T$. If between any two hybrids the distribution of the runtime of M is not indistinguishable, then we can easily construct a distinguisher for $H(x||\mathcal{U}_{(d-1)\lambda})$ and $\mathcal{U}_\lambda$. Hence, $\mathsf{Steps}_M(x)$ should be computationally indistinguishable from $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$. It follows that M must be $(t, \mathcal{C}_{T/2^\lambda}(O(t)) - \mathsf{negl}(\lambda))$-successful, for any $t \in poly(\lambda)$.

Next, note that M, after finding a small hash, hashes again the result with a fresh randomly sampled string w_2. For some input x of algorithm M, let $\hat{\mathcal{X}}$ be the distribution of the small hash. Since $\hat{Z} \overset{c}{\approx} \mathcal{U}_\lambda$, it follows that $H_k(\hat{\mathcal{X}}, \mathcal{U}_{(d-1)\lambda})$ will be computationally indistinguishable from U_λ. Hence, for the simulator Ψ that outputs a randomly sampled pair from $\mathcal{U}_\lambda$ and $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$, M satisfies the next-problem simulatability property. Note, that using the inverse transform technique, we can sample from the geometric distribution (truncated to 2^λ) in $O(\lambda)$ steps.

Finally, the witness malleability property holds if we assume that $\Phi(x, (w_1, w_2))$ returns the witness (w_1, w_2'), where w_2' is sampled uniformly at random. Again, $S(x, \Phi(x, (w_1, w_2)))$ will be indistinguishable from $\mathcal{U}_\lambda$.

□

Regarding the hard-ISP property, to our knowledge no reductions exist (yet) of iterated hardness to weaker assumptions in the standard model, even for the more extensively studied iterated sequential functions mentioned earlier. We argue the plausibility of our construction being hard against precomputation (Definition 10), by the fact that Construction 2 is based on Bitcoin's ISP construction (see Section 3.1 for a simplified version), for which no attacks are known. The main idea is that if there exists an attacker against our Construction, then we can use it to break the hardness of Construction 1. In more detail, given as input a statement x, the 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. The following lemma highlights this relation:

Lemma 41. *If Construction 1 is based on a C-CRRE hash family $\mathcal{H}$ and is (t, δ, k)-hard against precomputation, then Construction 2, also based on $\mathcal{H}$, is (t, δ, k)-hard against precomputation.*

Proof. As mentioned, Construction 2 is a mirror image of Construction 1, in the sense that the first hash, instead of the second, is required to be smaller than T, and the other one can have an arbitrary value. We make the following simplifying assumptions: $\mathcal{H}$ is the same in both constructions, M in Construction 1 chooses *ctr* at random[12] as in Construction 2, and the size of W in Construction 1 is adapted accordingly. For the sake of contradiction, assume that Construction 2 is not (t, δ, k)-hard against precomputation. Using similar arguments as in the analysis of Lemma 40, the distribution of the runtime of the solvers of both ISPs is identical. Hence, it has to be the case that there exists an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that for infinitely many λ and some $m \geq k$ breaks the hardness of Construction 2. Using $\mathcal{A}$, we are going to construct an attacker $\mathcal{A}'$ that breaks the hardness of Construction 1.

Let $\mathcal{A}'$ work as follows: First, $\mathcal{A}'_1$ runs $\mathcal{A}_1$ and forwards variable *st* to $\mathcal{A}'_2$. Then, $\mathcal{A}'_2$ on input *st* and a randomly sampled problem statement x, runs $\mathcal{A}_2$ on input $H(x||w)$, where w is sampled at random. If $\mathcal{A}_2$ succeeds, it outputs witnesses $((w_1, w'_1), \dots, (w_m, w'_m))$. Then, $\mathcal{A}'_2$ outputs $((w, w_1), \dots, (w'_{m-1}, w_m))$. Note, that in that case (w, w_1) is a witness for x, for the game $\mathcal{A}'$ is playing, since $H(H(x||w)||w_1) < T$. Moreover, it should hold that $H(H(H(H(x||w)||w_1)||w'_1)||w_2) < T$. In turn, this implies that $((w, w_1), (w'_1, w_2))$ is a valid sequence of witnesses for $\mathcal{A}'$. Similarly, it follows that $((w, w_1), \dots, (w'_{m-1}, w_m))$ is a valid sequence of m witnesses for the game $\mathcal{A}'$ is playing. Hence, whenever $\mathcal{A}$ wins, $\mathcal{A}'$ also wins.

We proceed to analyze the winning probability of $\mathcal{A}'$. We have already argued that whenever $\mathcal{A}$ wins, $\mathcal{A}'$ also wins. Moreover, we have assumed that $\mathcal{A}$ succeeds in producing $m \geq k$ witnesses with non-negligible probability. Due to the randomness extraction property of $\mathcal{H}$, the distribution of $H(x, w)$ will be computationally indistinguishable from the uniform distribution over $\{0, 1\}^\lambda$. Hence, the probability that $\mathcal{A}$ wins is negligibly close to the probability that it wins on a uniformly random input, and thus $\mathcal{A}'$ wins also with non-negligible probability. This is a contradiction, and the lemma follows.

□

In the same spirit, we require $\mathcal{H}$ to be collision resistant, as otherwise it is possible that an adversary who finds collisions can break the hard-ISP property by creating cycles.[13] Due to Theorem 38 and the previous two lemmas, the following theorem holds:

Theorem 42. *Assuming the existence of a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and that Construction 1 is a hard-ISP against precomputation based on a C-CRRE hash family $\mathcal{H}$, that comply with Assumption 1, protocol $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ based on Construction 2 implements a robust public transaction ledger with parameters $k = \frac{\max\{4k_{\mathsf{hard}}, \lambda\}}{\gamma\delta}$ and $u = \frac{2k}{(1-\frac{\delta}{4})f}$, except with negligible probability in λ.*

References

[1] J. Alwen and B. Tackmann. Moderately hard functions: Definition, instantiations, and applications. In *Theory of Cryptography TCC*, 2017.

[2] M. Andrychowicz and S. Dziembowski. Distributed cryptography based on the proofs of work. Cryptology ePrint Archive, Report 2014/796, 2014. `http://eprint.iacr.org/`.

[12] We can avoid this simplification by further assuming that $\mathcal{H}$ is a statistical extractor and also a PRF, similarly to [38]. We leave this for the full version of the paper.

[13] Note that requiring $\mathcal{H}$ to be an extractor is sufficient to guarantee the collision resistance of the cascade construction [23].

[3] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged Byzantine impostors. Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer Science, July 2005.

[4] A. Back. Hashcash–a denial of service counter-measure, 2002.

[5] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Advances in Cryptology - CRYPTO*, 2017.

[6] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of work from worst-case assumptions. In *Advances in Cryptology - CRYPTO*, 2018.

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Foundations of Computer Science*. IEEE, 1997.

[8] M. Bellare, J. Jaeger, and J. Len. Better than advertised: Improved collision-resistance guarantees for md-based hash functions. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 891–906, New York, NY, USA, 2017. ACM.

[9] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

[10] M. Bellare and P. Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 399–416. Springer, 1996.

[11] D. J. Bernstein and T. Lange. Non-uniform cracks in the concrete: the power of free precomputation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340. Springer, 2013.

[12] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In M. Sudan, editor, *Proceedings of the 2016 ACM ITCS*, pages 345–356. ACM, 2016.

[13] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO 2018.*

[14] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Annual International Cryptology Conference*, pages 757–788. Springer, 2018.

[15] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, Berlin, Heidelberg, 2000.

[16] R. Canetti, Y. Chen, L. Reyzin, and R. D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 91–122. Springer, 2018.

[17] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[18] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, Berlin, Heidelberg, 2002.

[19] D. Dachman-Soled, R. Gennaro, H. Krawczyk, and T. Malkin. Computational extractors and pseudorandomness. In R. Cramer, editor, *Theory of Cryptography*, pages 383–403, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] I. B. Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

[21] Y. Dodis. On extractors, error-correction and hiding all partial information. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security, 2005.*, pages 74–79. IEEE, 2005.

[22] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the cbc, cascade and hmac modes. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 494–510, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[23] Y. Dodis and P. Puniya. Getting the best out of existing hash functions; or what if we are stuck with sha? In S. M. Bellovin, R. Gennaro, A. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security*, pages 156–173, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[24] J. R. Douceur. The sybil attack. In P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, editors, *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[25] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[26] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. Continuous non-malleable codes. In *Theory of Cryptography - TCC*, 2014.

[27] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*. Springer, 1986.

[28] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, 2015.

[29] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol with chains of variable difficulty. *IACR Cryptology ePrint Archive*, 2016:1048, 2016.

[30] J. A. Garay, A. Kiayias, N. Leonardos, and G. Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, pages 465–495, 2018.

[31] J. A. Garay, A. Kiayias, and G. Panagiotakos. Consensus from signatures of work. Cryptology ePrint Archive, Report 2017/775, 2017. `https://eprint.iacr.org/2017/775`.

[32] J. A. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. *Journal of cryptology*, 24(4):615–658, 2011.

[33] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In L. Fortnow and S. P. Vadhan, editors, *Symposium on Theory of Computing, STOC 2011*, pages 99–108. ACM, 2011.

[34] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. In *Foundations of Computer Science (FOCS 2003)*. IEEE Computer Society, 2003.

[35] A. Juels and J. G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*. The Internet Society, 1999.

[36] J. Katz, A. Miller, and E. Shi. Pseudonymous secure computation from time-lock puzzles. *IACR Cryptology ePrint Archive*, 2014:857, 2014.

[37] A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. Technical report, IACR: Cryptology ePrint Archive, 2015.

[38] H. Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 631–648, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[39] A. K. Lenstra and B. Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015. `https://eprint.iacr.org/2015/366`.

[40] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[41] M. Naor. On cryptographic assumptions and challenges. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.

[42] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[43] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, Feb. 1996.

[44] R. Pass, L. Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. Cryptology ePrint Archive, Report 2016/454, 2016. `http://eprint.iacr.org/2016/454`.

[45] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[46] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.

[47] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. CRYPTO '01, London, UK, UK, 2001.

[48] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. `http://eprint.iacr.org/`.

A Cryptographic primitives and building blocks (cont'd)

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

Collision resistance. We make use of the following notion of security for cryptographic hash functions:

Definition 43. Let $\mathcal{H} = \{\{H_k : M(\lambda) \to Y(\lambda)\}_{k \in K(\lambda)}\}_{\lambda \in \mathbb{N}}$ 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 $\mathcal{H}$,

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

Randomness extractors. We also make use of the notion of randomness extractors, introduced in [43]. A random variable X is a k-source if $H_\infty(X) \geq k$, i.e., if $\Pr[X = x] \leq 2^{-k}$.

Definition 44. A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-*extractor* if for every k-source X on $\{0,1\}^n$, $\Delta[\mathsf{Ext}(X, U_d), U_m] \leq \epsilon$.

The definition of a (k, t, ϵ)-*computational* randomness extractor [19] is exactly the same, substituting statistical distance with (t, ϵ)-computational indistinguishability, where t refers to the runtime of the distinguisher and ϵ to its probability of success.

Robust non-interactive zero-knowledge. In our construction (Section 4) we make use of the following notion, introduced in [47].

Definition 45. Given an NP relation R, let $\mathcal{L} = \{x : \exists w \text{ s.t. } R(x, w) = 1\}$. $\Pi = (q, P, V, S = (S_1, S_2), E))$ is a *robust NIZK argument* for $\mathcal{L}$, if $P, V, S, E \in PPT$ and $q(\cdot)$ is a polynomial such that the following conditions hold:

1. Completeness. For all $x \in \mathcal{L}$ 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

$$\textsc{Real}(\lambda) = \{\Omega \leftarrow \{0,1\}^{q(\lambda)}; out \leftarrow \mathcal{A}^{\mathsf{P}(\Omega,\cdot,\cdot)}(\Omega); \text{Output } out\},$$
$$\textsc{Sim}(\lambda) = \{(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); out \leftarrow \mathcal{A}^{\mathsf{S}_2'(\Omega,\cdot,\cdot,tk)}(\Omega); \text{Output } out\},$$

   and $\mathsf{S}_2'(\Omega, x, w, tk) \overset{\triangle}{=} \mathsf{S}_2(\Omega, x, tk)$ 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 [26], we also require that the proof system supports labels. That is, algorithms $\mathsf{P}, \mathsf{V}, \mathsf{S}, \mathsf{E}$ take as input a public 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 46 ([47])**.** *Assuming trapdoor permutations and a dense cryptosystem exist, robust NIZK arguments exist for all languages in $\mathcal{NP}$.*