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

March 29, 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) any blockchain protocol implies a hard ISP problem, i.e., ISP hardness is necessary for secure blockchain protocols—but not sufficient by itself, 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 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 Horizon 2020 project PANORAMIX, No. 653497.

Contents

1 Introduction

Blockchain protocols, introduced by Nakamoto [33], 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 [19] initially as a spam protection mechanism, and subsequently found applications in other domains such as Sybil attack resilience [18] and denial of service protection [29, 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 PoW in the context of an application 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, 11]), 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 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 [22, 37, 23, 5], the required PoW properties have not been identified and the analysis has been carried out in the random oracle (RO) model [8]. The same is true for a wide variety of other protocols in the space, including [2, 30, 24].

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.* [15] prove 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 [35] proves that efficient non-committing encryption has no instantiation in the standard model but a straightforward implementation in the RO model, while Goldwasser and Kalai [28] show that the Fiat-Shamir heuristic [21] does not necessarily imply a secure digital signature which is in contrast to the result by Pointcheval and Stern [38] 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).

Naor [34] introduced a framework for classifying cryptographic hardness assumptions and identified the concept of a *falsifiable assumption* which was further studied by Gentry and Wichs [27]. 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 one of the main open questions that motivate our work:

> *Is it possible to prove the security of blockchain protocols in the standard model under falsifiable assumptions?*

Our results. 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, solving instances in an iterative fashion as produced by the successor algorithm whenever a new witness is found. 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 in the blockchain setting. We achieve this by considering the natural ISP problem implied by any implementation of a blockchain protocol. The ISP problem is defined by the blockchain structure itself: The problem 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 attacker against the iterated hardness of this ISP, results in a successful attack against the blockchain protocol. At the same time, we prove that iterated hardness of this ISP is by itself insufficient to prove the security of the blockchain protocol: Indeed, there could be ISP problems that are hard but not parallelizable in a non-interactive fashion, something that we observe can lead to an attack against the underlying blockchain protocol.

Motivated by the above, we list a set of additional properties[1] that epitomize what we call an *enhanced* ISP from which we can provably derive the security of a blockchain protocol. Then we show how these additional properties can be instantiated via a suitable randomness extractor. 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 has an *almost independent run-time* when the number of steps of the solving algorithm is almost independent across different problem statements. 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: (i) A novel blockchain protocol whose security can be reduced to the hardness of the underlying enhanced ISP, and (ii) that in the case of our hash-based ISP, it is possible to derive *all* the extra properties of the enhanced ISP from a simple

[1] Some of the security properties, as well as the computational model defined in Section 3.1, were first introduced in [25]. The current characterization of the underlying primitive in [25] is as a signature of work; refer to previous versions for the PoW formulation.

property relating to the randomness extraction capability of the underlying cryptographic hash function. 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.

We perform our analysis in the static setting with synchronous rounds as in [22], 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 [37]. Further, we leave as an open question the extension of our results to the dynamic setting of [23] as well as matching the 50% threshold on adversarial computational power of the Bitcoin blockchain which can be shown in the RO model.

We note that a related but distinct notion of hardness, *sequential*, i.e., non-parallelizable iterated hardness, has been considered as early as [39], mainly in the domains of timed-release cryptography [13] and protocol fairness [26], and recently formalized in [12]. In addition, a number of candidate hard problems have been proposed, including squaring a group element of a composite moduli [39], hashing, and computing the modular square root of an element on a prime order group [32]. However, as we prove, this type of hardness is incompatible with PoW-based blockchains, and this is the reason we put forth the notion of hard ISPs. The distinction is that ISPs allow paralelization, which is crucial for proving the security of a (Bitcoin-like) blockchain protocol.

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.

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, $x \xleftarrow{\$} X$ 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 $\mathcal{U}$. 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 $\Delta(X, Y) \leq \epsilon$. $\approx$ 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, 9, 26, 10] 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. [10]; 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.* [26]; refer to [26] for details on using interactive RAMs in a UC-like framework, as well as to Section 3.1. Given a RAM M, we will denote by $\mathsf{Steps}_M(x)$ the random variable that corresponds to the number of steps of M given input x. We will say that M is t-bounded if it holds that $\Pr[\mathsf{Steps}_M(x) \leq t(|x|)] = 1$.

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.

Cryptographic primitives and building blocks. We will make use of the following cryptographic primitives: cryptographic hash functions, randomness extractors [36], robust non-interactive zero-knowledge (NIZK) [40], and iterated sequential functions.

Iterated sequential functions. We recite the hardness definition introduced in [12]:

Definition 1 ([12])**.** $f : X \to Y$ 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 2 ([12])**.** Let $g : X \to X$ be a function which satisfies (t, ϵ)-sequentiality. A function $f : \mathbb{N} \times X \to X$ defined as $f(k, x) = g^{(k)}(x) = g \circ g \circ \ldots \circ g$ 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.

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

Definition 3. 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 $\mathcal{H}$,

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

Randomness extractors. We also make use of the notion of randomness extractors, introduced in [36]. A random variable X is a k-source if $H_\infty(X) \geq k$, i.e., if $\Pr[X = x] \leq 2^{-k}$.

Definition 4. 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 difference $k + d - m$ is called the *entropy loss* of the extractor.

Robust non-interactive zero-knowledge. In our construction (Section 4) we will make use of the following notion, introduced in [40].

Definition 5. 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
$$\textsc{Real}(\lambda) = \{\Omega \leftarrow \{0,1\}^{q(\lambda)}; \mathit{out} \leftarrow \mathcal{A}^{\mathsf{P}(\Omega,\cdot,\cdot)}(\Omega); \text{Output } \mathit{out}\},$$
$$\textsc{Sim}(\lambda) = \{(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); \mathit{out} \leftarrow \mathcal{A}^{\mathsf{S}_2'(\Omega,\cdot,\cdot,tk)}(\Omega); \text{Output } \mathit{out}\},$$
and $\mathsf{S}_2'(\Omega, x, w, tk) \stackrel{\triangle}{=} \mathsf{S}_2(\Omega, x, tk)$ 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 [20], 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 6 ([40]). *Assuming trapdoor permutations and a dense cryptosystem exist, robust NIZK arguments exist for all languages in NP.*

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. To analyze the problem, we first present an appropriate security model, introduced in [25], where a standard model analysis can be carried out. Our focus will be on distilling 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 3.2, where we introduce the notion of *hard iterated search problems*. Then, in Section 3.3 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 The Bitcoin backbone model

In [22], an abstraction was proposed for the analysis of the Bitcoin protocol. Here we overview the basics, substituting IRAMs for ITMs for the reasons explained in Section 2. 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 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 by 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 $\mathcal{C}$. 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.

Next, we describe how different parties communicate. Initially, the diffusion functionality sets the variable round to be 1. It also maintains a Receive() string (register) 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() tapes. 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, $\mathcal{A}$ and $\mathcal{Z}$.

Next, we consider the complications in the modeling due to the analysis of Bitcoin in the concrete security setting. Both in [22] and [37] a modified version of the standard simulation-based paradigm of [14] is followed, where there exist both a malicious environment and a malicious adversary. In addition, the underlying computational problem is modeled in a non black-box way using a random oracle (RO), and the computational power of the adversary is then bounded by limiting the number of queries it can make to the RO per round. Since in this work the underlying computational problem is modeled in a black-box way, an alternative approach to bound the adversary's power is needed.

A naïve first approach is to only bound the computational power of $\mathcal{A}$. Unfortunately this will not work for several reasons. Firstly, nothing stops the environment from aiding the adversary, i.e., computing witnesses, and then communicating with it through their communication channel or some other subliminal channel; secondly, even if we bound the *total* number of steps of $\mathcal{A}$, it is not clear how to bound the steps it is taking per round in the model of [14], which we build on. Furthermore, if the adversary is able to send, say, θ messages in each round, it can force each honest party to take θ times the verification time extra steps per round. If we do not bound θ, then the

adversary will be able to launch a DOS attack and spend all the resources the honest parties have[2].

In order to capture these considerations 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 7. Let $(t_{\mathcal{A}}, \theta)$-good be a predicate defined on executions in the hybrid setting described above. Then E *is* $(t_{\mathcal{A}}, \theta)$*-good*, where E is one such execution, if

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

Definition 8. Given a predicate Q and bounds $t_{\mathcal{A}}, \theta, t, n \in \mathbb{N}$, with $t < n$, we say that protocol Π satisfies property Q for n parties assuming the number of corruptions is bounded by t, provided that for all PPT $\mathcal{Z}, \mathcal{A}$, the probability that $Q(\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}})$ is false and the execution is $(t_{\mathcal{A}}, \theta)$-good is negligible in λ.

3.2 Iterated search problems

An *iterated search problem* (ISP) $\mathcal{I}$ specifies a collection $(I_\lambda)_{\lambda \in \mathbb{N}}$ of distributions.[4] For every value of the security parameter $\lambda \geq 0$, I_λ is a probability distribution of *instance descriptions*. An instance description Λ specifies: (i) finite, non-empty sets X, W, and (ii) 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 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:

- *Solving* algorithm $M_\Lambda(x)$: A probabilistic algorithm that takes as input an instance description $\Lambda[X, W, R]$, and a problem statement x and outputs a witness w.
- *Verification* algorithm $V_\Lambda(x, w)$: A deterministic algorithm that that takes as input an instance description $\Lambda[X, W, R]$, 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 an instance description $\Lambda[X, W, R]$, a problem statement x, and a valid witness w and outputs a new instance $x' \in X$.

In the sequel, we will omit writing Λ when it is clear from the context. 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, $M(x)$ outputs such a witness with overwhelming probability in λ. As an example, we present Bitcoin's underlying computational problem captured as an ISP:

- I_λ is the uniform distribution over functions $H : \{0,1\}^* \to \{0,1\}^\lambda$ in some family[5] of hash functions $\mathcal{H}$, i.e., $\Lambda = \{H\}$;
- $X = \{0,1\}^\lambda, W = \{0,1\}^* \times \{0,1\}^\lambda$;
- $R = \{(x, w) | H(H(x||m)||ctr) < T, \text{ for } w = m||ctr\}$;
- $V(x, w)$ checks whether $H(H(x||m)||ctr) < T$, for $w = m||ctr$;

[2] This problem is extensively discussed in [2], Section 3.4.

[3] The adversary cannot use the running time of honest parties that it has corrupted; it is activated instead of them during their turn. Also, note that it is possible to compute this number by counting the number of configurations that $\mathcal{A}$ or $\mathcal{Z}$ are activated per round.

[4] Here we follow the notation used to define *subset membership problems* in [16]. We remark that no other connection exists between the two papers.

[5] In the actual protocol a double invocation of SHA-256 is used.

– $M(x)$ tests whether $V(x,(m,ctr))$ is true, for different m, ctr pairs, until it finds a solution, and
– $S(x,w) = S(x,(m,ctr)) = H(H(x||m)||ctr)$.

where T is a protocol parameter that has to do with how hard it is solve a problem instance. For simplicity, in our exposition the hardness parameter for each ISP is fixed, and we do not capture it explicitly.

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

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

Finally, in the same spirit of Boneh *et al.* [12]'s definition of an iterated sequential function (cf. Definition 1), 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, which in turn allows for the solving algorithm S to be parallelizable. In that sense, our notion relaxes the strict convergence criterion of [12].

Definition 9. An ISP $\mathcal{I} = (V, M, S)$ is (t, δ, k_0)-*hard* iff it holds that:

– for sufficiently large $\lambda \in \mathbb{N}$, for all $\Lambda[X,W,R] \in I_\lambda$, $x \in X$, and for all polynomially large $k \geq k_0$:

$$\Pr\left[\begin{matrix}(w_i)_{i\in[k]} \leftarrow M^k(x) : (x,(w_i)_i) \in R \\ \wedge\, \mathsf{Steps}_{M^k}(x) \leq k\cdot t\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $M^k(x)$ denotes k iterations of M on inputs $(x, S(x, M(x)), \ldots)$, and

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

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

3.3 An ISP-based Bitcoin protocol

Next, we describe the Bitcoin backbone protocol with respect to an ISP $\mathcal{I} = (M, V, S)$, which we call $\Pi_{\mathsf{iPL}}(\mathcal{I})$. With foresight, we note that in Section 4 we will introduce a different ISP-based protocol, which we will prove secure.

At the start of the protocol, all parties have as common input a randomly sampled instance description $\Lambda[X,W,R]$ and a randomly sampled statement $x_{\mathtt{Gen}} \in X$. 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) $x_1 = x_{\mathtt{Gen}}$, (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 $x_{\mathtt{Gen}}$) 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.

Remark 1. In the ISP-based Bitcoin backbone protocol we have just described, transactions are not explicitly considered neither in the mining algorithm nor in the messages exchanged by the protocol. This corresponds to a special case of the Bitcoin Backbone protocol, where the environment only sends transactions at the rounds that some block was mined. Hence, the necessary conditions derived for $\Pi_{\mathsf{iPL}}(\mathcal{I})$, are weaker than those of the Bitcoin Backbone that also model transactions.

Security properties of the blockchain. A number of desired basic properties for the blockchain were introduced in [22, 31, 37]. 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 of the property, presented in [31, 37], 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 $\mathcal{C}$.

Definition 10 ((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 $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, 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 11 (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 $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, 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 [22] an important lemma was established relating to the rate at which the chains of honest players were increasing as the Bitcoin backbone protocol was run. This was explicitly considered in [31] as a property under the name *chain growth.*

Definition 12 (Chain Growth)**.** The chain growth property Q_{cg} with parameters $\tau \in \mathcal{R}$ (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 $|\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau \cdot s$.

For the purposes of our analysis we will also make use of lower and upper bounds in *honest chain growth.* Honest chain growth refers to the chain growth property when limited to executions where the adversary remains silent. The lower bound of honest chain growth is at least as big as the chain growth parameter τ. On the other hand, an upper bound for the property is guaranteed, since the running time of honest parties per round is limited.

Robust public transaction ledgers. A *public transaction ledger* [22] 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 $\text{tx} \in \mathcal{T}$. Ledgers correspond to chains in the backbone protocol. In the protocol execution there also exists an oracle Txgen that generates valid transactions. Note, that it is possible for the adversary to create two transactions that are conflicting; valid ledgers must not contain conflicting transaction. We will assume that the oracle is unambiguous, i.e., that 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 comes in conflict with it.

Definition 13. A protocol Π implements a *robust public transaction ledger* if 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, then tx will always be reported in the same position in the ledger by any honest player from this round on.
- **Liveness:** Parameterized by $u, k \in \mathbb{N}$ (the "wait time" and "depth" parameters, resp.), provided that a transaction either (i) issued by Txgen, or (ii) is neutral, is given as input to all honest players continuously for u consecutive rounds, then there exists an honest party who will report this transaction at a block more than k blocks from the end of the ledger.

Finally, note that persistence and liveness imply in a straightforward way the common-prefix and chain-growth properties, with parameters k and $\tau = \frac{1}{u}$, respectively. Through the rest of this section, we take advantage of this fact, to argue about the necessary conditions that the ISP that Bitcoin utilizes must satisfy.

3.4 Iterated hardness is necessary

We now analyze the necessary conditions that Bitcoin's underlying ISP should satify in order for Bitcoin to be secure. In particular, we show that the ISP used in Bitcoin must be hard (Definition 9). 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. In addition, other, somewhat secondary properties are necessary to state our subsequent result. We start with those.

First, we introduce some additional notation. For some $\Lambda[X, W, R] \in I_\lambda$, denote the distribution over X induced by the standard sampler by $\tilde{X}_0(\Lambda)$, and let $\tilde{X}_i(\Lambda)$ be equal to $S(\tilde{X}_{i-1}(\Lambda), M(\tilde{X}_{i-1}(\Lambda)))$. $\tilde{X}_i(\Lambda)$ follows the distribution of the i-th problem in a sequence generated using the solver M and starting from a randomly sampled problem in X.

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 14. 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

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

Next, we define t_{ver} to be an upper bound on the the running time of the verification algorithm V for R.

Definition 15. An ISP $\mathcal{I} = (V, M, S)$ is t_{ver}-*verifiable* iff algorithm V takes time at most t_{ver} (on all inputs).

Intuitively, the time to verify that a tuple is in R 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 $\mathcal{I}$ is t_{ver}-verifiable, and collision resistant. We are now ready to state the necessary condition regarding ISPs.

Theorem 16. *Let $n, t, t_{\mathcal{H}}, t_{\mathcal{A}}$ such that $t_{\mathcal{A}} = c \cdot t_{\mathcal{H}}$, for some $c \in (0,1)$, and $\mathcal{I}$ be an ISP. If $\Pi_{\mathsf{iPL}}(\mathcal{I})$ satisfies the common-prefix and chain growth properties with parameters k and τ, s, respectively, then there exists $\delta \in (0,1)$ such that $\mathcal{I}$ is $(\frac{(n-t)t_{\mathcal{H}}}{\tau}, \delta, \max\{k, s \cdot \tau\})$-hard.*

Proof. Let τ' be an upper bound on the (honest) chain growth rate, where $\tau \leq \tau'$, and let $\delta = 1 - c \cdot \frac{\tau}{\tau'}$. It holds that $\delta \in (0,1)$. W.l.o.g., let parameter r_0 of the chain growth property be 0.

For the sake of contradiction, assume that $\mathcal{I}$ is not $(\frac{(n-t)t_{\mathcal{H}}}{\tau}, \delta, \max\{k, s \cdot \tau\})$-hard. First, we show that the first part of the definition of the hardness property is satisfied. Since, τ is a lower bound on the chain growth rate of the honest parties, there exists a RAM M' (the parallel execution of n instances of M) that after $m \cdot \frac{(n-t)t_{\mathcal{H}}}{\tau}$ steps (or $\frac{m}{\tau}$ rounds) computes m blocks with overwhelming probability on λ, for any $m \geq s \cdot \tau$. This implies that, there exists an adversary $\mathcal{A}$ that breaks the second part of the property with non-negligible probability, for some $m_0 \geq \max\{k, s \cdot \tau\}$. We are going to use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ that breaks the security of Bitcoin.

$\mathcal{A}'$ runs $\mathcal{A}$ on input (Λ, x). $\mathcal{A}$ after $(1-\delta)m_0\frac{(n-t)t_{\mathcal{H}}}{\tau}$ steps, i.e., less than $(1-\delta)m_0\frac{(n-t)t_{\mathcal{H}}}{\tau \cdot t_{\mathcal{A}}}$ rounds, will have extended x by at least m_0 blocks with non-negligible probability. On the other hand, due to the honest chain growth upper bound, during the same time, the honest parties will extend their chains by at most

$$(1-\delta)m_0\frac{(n-t)t_{\mathcal{H}}}{\tau \cdot t_{\mathcal{A}}}\tau' \leq c\frac{\tau}{\tau'}m_0\frac{1}{c}\frac{\tau'}{\tau} \leq m_0$$

blocks. It follows that with non-negligible probability the adversary can create a fork of size greater than $m_0 \geq k$ and break the common-prefix property, which is a contradiction. □

Note, that as c approaches 1 the theorem states that the resulting ISP should be secure for smaller δ. This is in line with our intuition: The better the security of the resulting protocol is, the harder the requirements from the ISP are.

The existence of hard ISPs is implied by the existence of iterated sequential functions (cf. Definition 1).

Lemma 17. *If there exist (t, δ)-iterated sequential functions, then there exist $(t, \delta, 1)$-hard ISPs.*

Proof. Let $f_\lambda : X \to X$ be a (t, δ)-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, M is the standard solver of f_λ, and $I_\lambda = \{[X, X, R]\}$. It is easy to see that $\mathcal{I}$ is a $(t, \delta, 1)$-hard ISP for the standard solver of f_λ. The lemma follows. □

We conclude this subsection by stating a stronger hardness property that allows the adversary some precomputation time and is also implied by Bitcoin.

Definition 18. An ISP $\mathcal{I} = (V, M, S)$ is (t, δ, k_0)-*hard against precomputation* iff there exists a RAM M as in Definition 9 and for any PPT RAM $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, for sufficient large $\lambda \in \mathbb{N}$, $k \geq k_0$, there exists a polynomially large m such that:

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

Theorem 19. *Let $n, t, t_{\mathcal{H}}, t_{\mathcal{A}}$ such that $t_{\mathcal{A}} = c \cdot t_{\mathcal{H}}$, for some $c \in (0,1)$, and $\mathcal{I}$ be an ISP. If $\Pi_{\mathsf{iPL}}(\mathcal{I})$ satisfies the common-prefix and chain growth properties with parameters k and τ, s, respectively, then there exists $\delta \in (0,1)$ such that $\mathcal{I}$ is $(\frac{(n-t)t_{\mathcal{H}}}{\tau}, \delta, \max\{k, s \cdot \tau\})$-hard against precomputation.*

Proof. For the sake of contradiction, assume the there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the hardness of $\mathcal{I}$. We are going to describe only $\mathcal{A}'_1$, as the second phase of the attack is as in Lemma 16. $\mathcal{A}'_1$ is going to take as input Λ. Then, it is going to run $\mathcal{A}_1$ with input Λ, and forward its output st to $\mathcal{A}'_2$. In the meantime, honest parties will have produced chains of at most polynomial length in λ. $\mathcal{A}'_2$ will pick the longest among these chains, and run $\mathcal{A}_2$ with input st, x, where x is the problem defined by the most recent block in the chain. The attack then proceeds as in the previous Lemma to break the common prefix property. Note, that the distribution of x is as defined in the hardness property. □

3.5 Iterated hardness is 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. The main reason being that the honest solving algorithm must be parallelizable—exactly the opposite of what sequential functions guarantee.

More formally, we will argue that 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. We first give a formal definition of this property, and then prove that is implied by Bitcoin.

Definition 20. Let $\delta \in (0,1)$ and $n \in \mathbb{N}$. An ISP instance $\mathcal{I} = (V, M, S)$ is (n, δ)-*non-interactive parallelizable* (NIP), iff there exists a polynomial $p(\cdot)$, such that for infinitely many $\lambda \in \mathbb{N}$, some polynomially large $m \in \mathbb{N}$, and some $(a_i)_{i\in[n]} \in [n]^n$ it holds that:

$$\Pr_{\substack{\Lambda[X,W,R]\leftarrow I_\lambda;\\ x_0,\ldots,x_n\leftarrow \tilde{X}_m(\Lambda)}}[\min\{\mathsf{Steps}_M(x_{a_i})\}_{i\in\{1,\ldots,n\}} < \frac{\mathsf{Steps}_M(x_0)}{n\cdot\delta}] \geq \frac{1}{p(\lambda)}.$$

Lemma 21. *Let $\mathcal{I}$ be an ISP that is not $(n-t,\delta)$-NIP. If $t_{\mathcal{A}} \geq \delta \cdot t_{\mathcal{H}}(n-t)$, then there exists an adversary that breaks the common prefix property in* $\text{VIEW}^{t,n}_{\Pi_{\mathsf{iPL}}(\mathcal{I}),\mathcal{A},\mathcal{Z}}$ *with non-negligible probability.*

Proof. For the sake of contradiction, assume that the lemma does not hold. Since $\mathcal{I}$ is $(n-t,\delta)$-NIP, it follows that for sufficiently large λ, any polynomially large m, and any $(a_i)_{i\in[n-t]} \in [n-t]^{n-t}$:

$$\Pr_{\substack{\Lambda[X,W,R]\leftarrow I_\lambda\\ x_0,\ldots,x_{n-t}\leftarrow \tilde{X}_m(\Lambda)}}[\min\{\mathsf{Steps}_M(x_{a_i})\}_{i\in\{1,\ldots,(n-t)\}} < \frac{\mathsf{Steps}_M(x_0)}{(n-t)\cdot\delta}] \leq \mathsf{negl}(\lambda)$$

We will describe an adversary $\mathcal{A}$ that corrupts t parties, and breaks the common prefix property with non-negligible probability. We are going to argue that the adversary starting from $x_{\mathtt{Gen}}$ can create a private chain of maximum length by iteratively invoking M. Let $T_{\mathcal{H}}$ be the number of steps it takes for at least one of the honest parties to extend its chain by one block, i.e., $T_{\mathcal{H}} = \min\{\mathsf{Steps}_M(x_i)\}_{i\in\{1,\ldots,n-t\}}$ where $(x_i)_i$ are the inputs used by the honest parties. During the same time, $\mathcal{A}$ runs a single invocation of M for $T_{\mathcal{A}} \geq T_{\mathcal{H}}\delta(n-t)$ steps. By our assumption, and with overwhelming probability it holds that

$$\mathsf{Steps}_M(x_0) \leq (n-t)\cdot\delta\cdot\min\{\mathsf{Steps}_M(x_i)\}_{i\in\{1,\ldots,n-t\}} = \delta(n-t)T_{\mathcal{H}} \leq T_{\mathcal{A}},$$

where x_0 is the problem that the adversary is solving. Hence, $\mathcal{A}$ successfully computes a block on his own during the same period, and thus increases the length of the fork by one block. $\mathcal{A}$ repeats

this process, until a deep-enough fork is generated, and the common prefix property is violated. Note, that the distribution of problems solved both by the honest parties and the adversary follows the distribution defined in the NIP property. □

The optimal scenario here would be for $\mathcal{I}$ to be $(n, 1-\mathsf{negl}(\lambda))$-NIP. That is, if the total number of steps of the single running machine were smaller by a negligible factor, then with non-negligible probability the n smaller parties would find a solution first; i.e., no computational power would be lost by distributing the computation. On the flip side, the worst-case scenario is for $\mathcal{I}$ to be $(n, \frac{1}{n})$-NIP. That is, with non-negligible probability a single machine manages to compute a solution by taking the same number of steps as one of the n distributed machines.

Next, we prove that iterated sequential functions are not parallelizable.

Lemma 22. *Let $f(k, x)$ be an (t, ϵ)-iterated sequential function. For all $n \in \mathbb{N}$, the implied ISP* is not *$(n, \frac{1}{n})$-NIP.*

Proof. Let M be the solver for f. By definition, we have that for some polynomial $\mathsf{poly}(\cdot)$ it holds that

$$\min\{\mathsf{Steps}_M(x_i)\}_{i\in\{1,\ldots,n\}} = \mathsf{Steps}_M(x_0) = t \cdot \mathsf{poly}(\log(t), \lambda)$$

for any inputs $x_0, \ldots, x_n$. Hence, it follows that

$$\min\{\mathsf{Steps}_M(x_i)\}_{i\in\{1,\ldots,n\}} \geq \frac{\mathsf{Steps}_M(x_0)}{n \cdot \frac{1}{n}}$$

and the lemma follows. □

It follows from the previous two lemmas that if Bitcoin is based on an iterated sequential function, an adversary that has a little more power than one of the honest parties can break the security of the protocol.

Corollary 23. *Let $f(k, x)$ be an (t, ϵ)-iterated sequential function, and $\mathcal{I}$ be the implied ISP instance. If $t_{\mathcal{A}} \geq t_{\mathcal{H}}$, then there exists an adversary that breaks the common prefix property in* $\mathrm{VIEW}^{t,n}_{\Pi_{\mathsf{iPL}}(\mathcal{I}),\mathcal{A},\mathcal{Z}}$ *with non-negligible probability.*

Remark 2. Our analysis, with minor differences, also applies to other well-known PoW-based blockchain protocols. For example, in the GHOST protocol [41] parties choose their chain by iteratively choosing the "heaviest" subtree of blocks, starting from the first block. By similar arguments as above, it is implied that it is necessary for the iterated search problem to be hard with respect to the rate at which the honest parties generate blocks (cf. the chain growth rate in Bitcoin). While this may sound as an improvement, it opens up the space for other attacks against the search problem, e.g., the ability to cheaply generate witnesses for a statement if you already know, one harms the protocol.

4 Sufficient Conditions and a Provably Secure ISP-based Blockchain

As we showed earlier, the existence of hard ISPs is necessary, but not sufficient, to prove secure most well-known PoW-based blockchain protocols. In this section we first define an enhanced, "blockchain-friendly" notion of security for ISPs, encompassing hardness, which then show to be sufficient to implement a provably secure blockchain. We conclude the section with a candidate ISP proposal that satisfies the extra security properties and is plausibly hard.

4.1 Enhanced ISPs

We now present a set of extra ISP properties, besides hardness, that are specific to the use of an ISP in blockchain applications. All properties presented here concern the solving algorithm M. Later on, in Section 4.3, we present a candidate problem that satisfies all these properties. 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.

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 or regularly. In order to combat this attack, we introduce two properties. First, we say that an ISP $\mathcal{I}$ is (t, α)-*successful* when the probability that M computes a witness in t steps is at least α.

Definition 24. An ISP $\mathcal{I} = (V, M, S)$ is (t, α)-*successful* iff for sufficiently large $\lambda \in \mathbb{N}$ it holds that

$$\Pr_{\substack{\Lambda[X,W,R] \leftarrow I_\lambda; \\ x \leftarrow X}} \left[\mathsf{Steps}_M(x) < t\right] \geq \alpha.$$

Moreover, as shown in Section 3.5, it is necessary for the honest solver M to be parallelizable. The next property has to do with the (limited) independence of the runtime of different invocations of the honest solver. This will prove crucial in ensuring that when multiple parties work together, the distribution of the number of them who succeed in producing a witness has some "good" variance and concentration properties. (However, this property alone is not sufficient to imply parallelization, as can be seen for example with the repeated squaring problem.)

Definition 25. An ISP $\mathcal{I} = (V, M, S)$ has *almost-independent runtime* iff for any polynomial $p(\cdot)$, for sufficiently large $\lambda \in \mathbb{N}$, and for all $\Lambda[X, W, R] \in I_\lambda$, there exists a set of mutually independent random variables $\{Y_i\}_{i \in [p(\lambda)]}$ such that for any $x_1, \ldots, x_{p(\lambda)} \in X$ it holds that $\Delta[(\mathsf{Steps}_M(x_i))_i, (Y_i)_i] \leq \mathsf{negl}(\lambda)$.

The iterative 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. The next property states that there exists an efficient simulator whose running time and output is computationally indistinguishable from the distributions of the time it takes to compute a witness w for some statement x and the next statement $S(x, w)$, respectively.

Definition 26. An ISP $\mathcal{I} = (V, M, S)$ is *t-next-problem simulatable* iff there exists an t-bounded RAM $\mathcal{S}$ such that for sufficiently large $\lambda \in \mathbb{N}$ and for all $\Lambda[X, W, R] \in I_\lambda$, it holds that

$$\{(S(x, M(x)), \mathsf{Steps}_M(x))\}_{x \in X} \overset{c}{\approx} \{\mathcal{S}(\Lambda)\}_{x \in X}$$

The next property has to do with a party's ability to "cheaply" compute witnesses for a statement, if it already knows one. We call this ISP property *witness malleability*.

Definition 27. An ISP $\mathcal{I} = (V, M, S)$ is *t-witness malleable* iff there exists a t-bounded RAM Φ such that for sufficiently large $\lambda \in \mathbb{N}$, and for all $\Lambda[X, W, R] \in I_\lambda$, it holds that

$$(x, \Phi(x, w)) \in R, \text{ for all } (x, w) \in R \text{ and } \{S(x, \Phi(x, w))\}_{(x,w) \in R} \overset{c}{\approx} \{S(x, M(x))\}_{(x,w) \in R}.$$

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

Definition 28. An ISP $\mathcal{I} = (V, M, S)$ is $\{t_{\mathsf{ver}}, t_{succ}, \alpha, t_{hard}, \delta_{hard}, k_0, t_{nps}, t_{mal}\}$-*enhanced* iff it is correct, t_{ver}-verifiable, (t_{succ}, α)-successful, $(t_{hard}, \delta_{hard}, k_0)$-hard against precomputation, almost runtime independent, t_{nps}-next-problem simulatable, and t_{mal}-witness malleable.

Remark 3. Gentry and Wichs in [27] 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 a caveat due to the concrete security approach our work takes: the challenger should take as input the number of steps of the adversary.

4.2 The provably secure ISP-based protocol

The main challenges that our protocol has to overcome is to achieve security guarantees similar to iterative hardness, in a setting where the adversary can also take advantage of the work of honest parties. Towards this end, blocks in our protocol instead of exposing the relevant witness computed, contain a proof of knowledge of such a valid witness through a non-interactive zero-knowledge (NIZK) proof. Moreover, to further reduce the security guarantees required by the ISP, 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 2).

4.2.1 Protocol description

The protocol 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 $L = \{(\Lambda[X, W, R], x, x') | \exists w \in W : (x, w) \in R \wedge S(x, w) == x'\}$.[6] Π_{NIZK} also supports labels, which we denote as a superscript on P and V. Moreover, as in [22], our protocol is parameterized by functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ that capture higher-level applications (such as Bitcoin). We assume that at the start of the execution all parties share a common reference string (CRS), 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 a reference string Ω, each randomly sampled from $I_\lambda, X, \mathcal{H}, \{0,1\}^{q(\lambda)}$, respectively.

First, we introduce some notation needed to understand the description of the algorithms (similarly to Section 3.3). We use the terms block and chain to refer to tuples of the form $\langle s, m, x, \pi \rangle \in \{0,1\}^\lambda \times \{0,1\}^* \times X \times \{0,1\}^{poly(\lambda)}$, and sequences of such tuples, respectively. The rightmost (resp., leftmost) block of chain $\mathcal{C}$ is denoted by $\mathrm{head}(\mathcal{C})$ (resp., $\mathrm{tail}(\mathcal{C})$). Each block contains the hash of the previous block s, a message m, the next problem x to be solved, and a NIZK proof π. We denote by $B_{\mathtt{Gen}} = \langle 0^\lambda, 0^\lambda, x_{\mathtt{Gen}}, 0^\lambda \rangle$ a special block called the *genesis block*. 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; and (iii) $s_{i+1} = H(s_i, m_i, x_i)$ and $\mathsf{V}^{s_{i+1}}((\Lambda, x_{i-1}, x_i), \pi_i)$ is true for all $i \in [k]$ (see Algorithm 1). We call $H(s_i, m_i, x_i)$ the *hash of block* B_i and denote it by $H(B_i)$, and define $H(\mathcal{C}) \stackrel{\Delta}{=} H(\mathrm{head}(\mathcal{C}))$.

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 its' chain. If it succeeds, and computes a witness w, it will then compute a NIZK proof that it knows a witness w such that $(x, w) \in R$ and $S(x, w) == x'$, for some $x' \in X$, and

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

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

with label $H(H(s, m, x), m', x')$, where m' is the output of the input contribution function $\mathrm{I}(\cdot)$; see Algorithm 3. Then, the party diffuses the new 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

In order to turn the above protocol into a protocol realizing a public transaction ledger suitable definitions were given in [22] for functions $V(\cdot), R(\cdot), I(\cdot)$. Here we change those definitions slightly as shown in Table 1. We denote the new public ledger protocol by $\Pi_{\mathsf{nPL}}(\mathcal{I})$.

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

function $\mathsf{pow}(m', \mathcal{C})$
  $\langle s, m, x, \pi' \rangle \leftarrow \text{head}(\mathcal{C})$
  $w \leftarrow M(x)$ ▷ 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.
    $s' \leftarrow H(s, m, x)$ ▷ Compute the hash of the last block.
    $s'' \leftarrow H(s', m', x')$ ▷ 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

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 $\mathcal{L}$, and each m_i starts with a neutral transaction of the form $r\|\|i$, where r is a string of length λ.
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 of the form $r\|\|\|\mathcal{C}\|$, where r is sampled uniformly at random.

Table 1: *The instantiation of functions* $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ *for protocol* Π_{nPL}.

4.2.2 Security analysis

In this section we prove that $\Pi_{\mathsf{nPL}}(\mathcal{I})$ implements a robust public transaction ledger (cf. Definition 13), assuming the underlying ISP $\mathcal{I}$ is $\{t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{hard}, \delta_{hard}, k_0, t_{nps}, t_{mal}\}$-enhanced; we define $t'_{\mathcal{H}}$ 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.

To ease the presentation, we will introduce two variables that capture most of the costs related to $\mathcal{I}$ and Π_{NIZK}. Let t_{bb} (bb for backbone) be an upper bound on the number of steps needed to run the code of an honest party in one round, besides the calls to $M, \mathsf{P}, \mathsf{V}$. By carefully analyzing protocol Π_{nPL} one can extract an upper bound on this value[7]. We introduce constants $t'_{\mathcal{A}}$ and $t'_{\mathcal{H}}$ to denote : (i) 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 run the NIZK extractor once, and (ii) the minimum

[7] Note that t_{bb} depends on the running time of three external functions: $V(\cdot), I(\cdot)$ 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.

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

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

number of steps that an honest party takes running M per round, respectively. They amount to:

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

It holds that in any round at least $n - t$ (non-corrupted) parties will run M for at least $t'_{\mathcal{H}}$ steps.

Next, we introduce some additional 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 and some block B, 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. Also, let $X(R) = \sum_{j \in R} X_j$ and define $Y(R)$ and $X_B(R)$ similarly. Finally, we denote by $\beta = ((1 - \delta_{hard}) \cdot t_{hard})^{-1}$ an upper bound on the rate at which the adversary can compute witnesses in the iterated hardness game.

We are now ready to state our main computational assumption regarding the honest parties and the adversary. In particular, we are going to assume that the honest parties have enough computing power in order for the rate of uniquely successful rounds to outperform the rate at which the adversary generates blocks. Note that in our approach the running time of the adversary and the running time of honest parties do not necessarily have the same value, i.e., the adversary may use a superior solving algorithm. Hence, the computational assumption takes into account the parameters related to the ISP (refer to Table 2 for the relevant parameters). Finally, note that the computational power assumption implies that the honest parties must have at least double the adversary's computational power.

Assumption 1 (**Computational Power Assumption**). It holds that $\gamma \geq 2(1 + \delta)\beta \cdot t'_{\mathcal{A}}$, for some $\delta \in (0, 1)$.

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

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
σ :	quality of concentration of random variables in typical executions
k :	number of blocks for the common-prefix property
k_0 :	convergence parameter of ISP hardness
ℓ :	number of blocks for the chain-quality property

Table 2: *The parameters in our analysis:* $\lambda, n, t, t_{\mathcal{H}}, t_{\mathcal{A}}, \theta, k, \ell$ *are in* $\mathbb{N}$, $f, \gamma, \beta, \delta, \sigma$ *are in* $(0, 1)$.

is similar to the Merkle-Damgard transform [17], 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[8] with the same hash.

Lemma 29. *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.

Fact 1. 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 $j \in [k]$, $H(\mathrm{head}(\mathcal{C})) = \mathrm{MD}(H(B_j), ((m_i, x_i))_{i\in\{j+1,\ldots,k\}})$.

Let $\mathcal{C}_1 = B_{\mathtt{Gen}}, B_1, \ldots, B_{\lfloor|\mathcal{C}_1|\rfloor}, \mathcal{C}_2 = B_{\mathtt{Gen}}, B'_1, \ldots, B'_{\lfloor|\mathcal{C}_2|\rfloor}$, $z = ((m_i, x_i))_{i\in\lfloor|\mathcal{C}_1|\rfloor}$ and $z' = ((m'_i, x'_i))_{i\in\lfloor|\mathcal{C}_2|\rfloor}$. 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 Fact 1, 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 also 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 included in a fixed position in $m_{|\mathcal{C}_1|}, m'_{|\mathcal{C}_2|}$ (cf. Table 1), it follows that $m_{|\mathcal{C}_1|} \neq m'_{|\mathcal{C}_2|}$, which in turn implies that $B_{|\mathcal{C}_1|} \neq B'_{|\mathcal{C}_2|}$. Since $H(\mathrm{head}(\mathcal{C}_1)) = H(\mathrm{head}(\mathcal{C}_2))$, 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 there exists ℓ less or equal to $|\mathcal{C}_1|$, such that $\mathrm{MD}(H(B_{\mathtt{Gen}}), ((m_i, x_i))_{i\in[\ell]}) = \mathrm{MD}(H(B_{\mathtt{Gen}}), ((m'_i, x'_i))_{i\in[\ell]})$ and $(m_\ell, x_\ell) \neq (m'_\ell, x'_\ell)$. The lemma thus follows. □

The following two properties, introduced in [22], regarding the way blocks are connected are implied by Lemma 29.[9]

[8] We stress that if two valid blocks differ only on the NIZK proof, they are considered the same. Note, that in both cases the rest of the contents of the blocks are the same.

[9] A third property, called "prediction," also introduced in [22], 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 30. 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 31. *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.

Lemma 32. *For any set of consecutive rounds R, where $|R| \geq k_0/\beta t'_{\mathcal{A}}$, and for any party P, the probability that P mined some honest block B during the first round of R and $Z_B(R) \geq \beta t'_{\mathcal{A}}|R|$, is* $\mathsf{negl}(\lambda)$.

Proof. Let $R = \{i' | i \leq i' < i + s\}$ and let E be the event where in $\text{VIEW}^{t,n}_{\Pi_{\mathsf{nPL}}(\mathcal{I}),\mathcal{A},\mathcal{Z}}$ the adversary has mined at least $\beta t'_{\mathcal{A}} s$ blocks until round $i + s$ that belong to the same chain and descend honest block B mined by party P at round i. For the sake of contradiction, assume that the lemma does not hold, and thus the probability that E holds is non-negligible. Using $\mathcal{A}$, we will construct an adversary $\mathcal{A}'$ that breaks the iterative hardness 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 force $\mathcal{A}$ 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 $\mathcal{Z}$ as variable st. It can do all this, since in the iterated hardness 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 generated from (possibly zero) iterated invocations of M and S on some randomly sampled input in X, as in Definition 18. 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 block that party P is extending at round i, and m_1 the input it produced using $I(\cdot)$. $\mathcal{A}'_2$ 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 abort. Otherwise, if it is successful and produces some witnesses w, it will broadcast the following block:

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

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 simuleatability 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 $\mathcal{S}$ 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 block that 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 at the same time the number of adversarial blocks between B_1 and B_j is maximized. 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 $B' = \langle \emptyset, x, w, \emptyset \rangle$. $\mathcal{A}_2'$ firsts 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: $\pi'' = \mathsf{S}_2^{H(H(B^*), m'', x'')}(\Omega, (\Lambda, x^*, S(x', w'')), tk)$. 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 $\text{VIEW}^{t,n}_{\Pi_{\mathsf{nPL}}(\mathcal{I}), \mathcal{A}, \mathcal{Z}}$.

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

Proof. We start by describing a sequence of hybrids:

- Hybrid H_0: The view of the adversary in $\text{VIEW}^{t,n}_{\Pi_{\mathsf{nPL}}(\mathcal{I}), \mathcal{A}, \mathcal{Z}}$.
- 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+1} to $H_{1+n \cdot s}$: In hybrid $H_{1+u \cdot v}$, 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 $H_{1+u \cdot v-1}$ is indistinguishable from $H_{1+(u \cdot v)}$ by contradiction. 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. 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 not indistinguishable, we can construct a PPT distinguisher D that distinguishes $(x_0, S(\tilde{X}_m, M(\tilde{X}_m)))$ from $(x_0, S(x_0, M(x_0)))$, where $\tilde{X}_m$ is the distribution that x is sampled from in the hardness experiment. This is a contradiction, since by the next-problem simuleatability property it follows that:

$$(x_0, S(\tilde{X}_m, M(\tilde{X}_m))) \overset{c}{\approx} (x_0, \mathcal{S}(\Lambda)) \overset{c}{\approx} (x_0, S(x_0, M(x_0)))$$

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:

$$(x, S(x, M(x))) \overset{c}{\approx} (x, \mathcal{S}(\Lambda)) \overset{c}{\approx} (x, S(x', M(x'))) \overset{c}{\approx} (x, S(x', \Phi(x', w')))$$

Note also, that the simulated running time, is computationally indistinguishable from the running time parties take when running M. The claim follows by the fact that $H_{1+n\cdot s}$ 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 extends the statement defined by the previous block by 1 witness, 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 $\beta t'_{\mathcal{A}} s + 1$, 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. W.l.o.g. assume that the adversary has computing power at least as big as one party, i.e., $t'_{\mathcal{A}} \geq t_{\mathcal{H}}$. Then, $\mathcal{A}'_2$ will have computed $\beta t'_{\mathcal{A}} s + 1 \geq \beta(s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}}) \geq k_0$ blocks in $s \cdot t'_{\mathcal{A}} + t_{\mathcal{H}}$ steps with non-negligible probability, which is a contradiction to our initial assumption that $\mathcal{I}$ is hard. $\square$

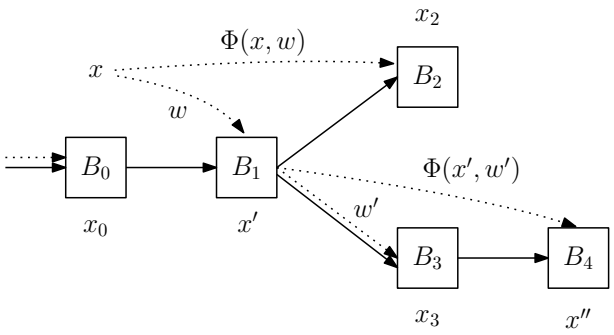

Figure 1: *A possible scenario according to Lemma 32. The blocks have been generated in order B_0, B_1, B_2, B_3, B_4, with B_3 being the only adversarial block. The cases where a valid witness is either known or can be extracted, and a NIZK proof has either been computed or simulated correspond to the dotted and normal edges, respectively.*

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 $k_0 \cdot (t'_{\mathcal{A}} + nt'_{\mathcal{H}})$-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 33. *For any set of consecutive rounds R, where $|R| \geq k_0/\beta(t'_A + nt'_{\mathcal{H}})$, and for any party P, the probability that P mined some honest block B during the first round of R and $Z_B(R) + X_B(R) > \beta(t'_A + nt'_{\mathcal{H}}) \cdot |R|$, is* $\mathsf{negl}(\lambda)$.

Next, we prove lower bounds on the rate of successful and uniquely successful rounds. Our proof crucially depends on the runtime independence property of $\mathcal{I}$. Specifically, the property implies that for some set of rounds the sum of the Bernoulli random variables of the event that a round is uniquely successful, concentrate around the mean. In turn, this implies that we can lower-bound the rate of uniquely successful rounds with good probability.

Lemma 34. *Let $\gamma = (n-t)\cdot\alpha(1-\beta t_{\mathcal{H}})^{n-1}$, $f = (1-(1-\alpha)^{n-t})$, and $\sigma \in (0,1)$. For any set of consecutive rounds R, with $|R| \geq \frac{\lambda}{\gamma\sigma^2}$, the following two events occur with negligible probability in λ:*

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

Proof. 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 $R = \{1,\ldots,s\}$.

Since this lemma talks about the steps taken by M, we are going to use the almost independent runtime property of $\mathcal{I}$, and do all the analysis on the mutually independent random variable defined by this property. For the rest of this proof, unless explicitly stated, assume that the $Steps_M(x)$ random variable refers to its idealized mutually independent version. We first buildup some notation to help in our analysis. For $\Lambda[X,W,R] \in I_\lambda$ and array $(x_{i,j}) \in X^{s\times n}$, let:

- random variable $P_{i,j} = 1$ if $Steps_M(x_{i,j}) \leq t_{i,j}$, and 0 otherwise;
- random variable $Y_i = 1$ if $\sum_{j=1}^{n} P_{i,j} = 1$, and 0 otherwise.
- random variable $X_i = 1$ if $\sum_{j=1}^{n} P_{i,j} \geq 1$, and 0 otherwise.
- random variable $Y = \sum_{i\in[s]} Y_i$, $X = \sum_{i\in[s]} X_i$.

It easily follows from the Successful property that $\Pr[P_{i,j}=1] \geq \alpha$. Next, we also prove an upper bound for $\Pr[P_{i,j}=1]$.

Claim 2. $\Pr[P_{i,j}=1] \leq t_{\mathcal{H}}\beta$

Proof. For the sake of contradiction, assume that $\Pr[P_{i,j}=1] > t_{\mathcal{H}}\beta$. By the iterative hardness property, we have that for $k \geq k_0$, any $(1-\delta)t_{hard}\cdot k$-bounded adversary will compute k witnesses with negligible probability in λ. This implies that the expected number of blocks any such adversary computes is at most k. By our assumption and the linearity of expectation, the expected number of blocks the honest solver can mine on the same time is greater than $(t_{\mathcal{H}}\beta)\cdot(1-\delta_{hard})t_{hard}\cdot k/t_{\mathcal{H}} = k$, where $\beta = ((1-\delta_{hard})t_{hard})^{-1}$. This is a contradiction, and the claim follows. ⊣

We continue by showing that the random variables we have defined are mutually independent.

Claim 3. *The random variable families $(P_{i,j})_{i\in[s],j\in[n]}$, $(Y_i)_{i\in[s]}$, and $(X_i)_{i\in[s]}$ are mutually independent.*

Proof. First, notice that the independence of the scheme implies independence of $(P_{i,j})$. We will show this for two random variables and the extension to m variables will be obvious. Let $P_1, P_2 \in$

$(P_{i,j})_{i,j}$, $b_1, b_2 \in \{0,1\}$ and $x_1, x_2 \in X$ then

$$\begin{aligned}
\Pr[P_1 =& b_1 \wedge P_2 = b_2] = \Pr[Steps_M(x_1) \in S_1 \wedge Steps_M(x_2) \in S_2] \\
&= \sum_{(s_1,s_2)\in S_1\times S_2} \Pr[Steps_M(x_1) = s_1 \wedge Steps_M(x_2) = s_2] \\
&= \sum_{(s_1,s_2)\in S_1\times S_2} \Pr[Steps_M(x_1) = s_1] \cdot \Pr[Steps_M(x_2) = s_2] \\
&= \sum_{s_1\in S_1} \Pr[Steps_M(x_1) = s_1] \cdot \sum_{s_2\in S_2} \Pr[Steps_M(x_2) = s_2] \\
&= \Pr[Steps_M(x_1) \in S_1] \cdot \Pr[Steps_M(x_2) \in S_2] \\
&= \Pr[P_1 = b_1] \cdot \Pr[P_2 = b_2]
\end{aligned}$$

where S_1, S_2 are either $[0,t]$ or (t,∞). We use the independence property on the third line.

Next, we prove the second point of the claim. Again, w.l.o.g we only show it for 2 random variables, Y_1, Y_2 and the extension to m is obvious. Let $b_1, b_2 \in \{0,1\}$, then

$$\begin{aligned}
\Pr[Y_1 = b_1 \wedge Y_2 = b_2] &= \Pr[\sum_{j\in[n]} P_{1,j} \in S_1 \wedge \sum_{j\in[n]} P_{2,j} \in S_2] \\
&= \sum_{(s_1,s_2)\in S_1\times S_2} \Pr[\sum_{j\in[n]} P_{1,j} = s_1 \wedge \sum_{j\in[n]} P_{2,j} = s_2] \\
&= \sum_{(s_1,s_2)\in S_1\times S_2} \Pr[\sum_{j\in[n]} P_{1,j} = s_1] \cdot \Pr[\sum_{j\in[n]} P_{2,j} = s_2] \\
&= \sum_{s_1\in S_1} \Pr[\sum_{j\in[n]} P_{1,j} = s_1] \cdot \sum_{s_2\in S_2} \Pr[\sum_{j\in[n]} P_{2,j} = s_2] \\
&= \Pr[Y_1 = b_1] \cdot \Pr[Y_2 = b_2]
\end{aligned}$$

where S_1, S_2 are $\{1\}$ or $\{0,2,3,\ldots\}$ depending on b_1, b_2. The same follows for $(X_i)_{i\in[s]}$. ⊣

Next, we lower bound the expected value of random variables $(Y_i)_i$ and $(X_i)_i$.

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

Proof.

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

The inequalities follow from the successful and iterative hardness properties. Note, that in order for $\mathbb{E}[Y_i]$ to be big, α must be as big as possible and β must be as small as possible. ⊣

Claim 5. *It holds that for any $i \in R : \mathbb{E}[X_i] \geq f$*

Proof.

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

The inequality follows from the successful property. ⊣

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

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

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

These results, with only negligible difference in probability, follow for the random variables in the real execution due to the almost runtime independence property and the fact that Y and X are functions of the random variables referred by this property.

□

Following the strategy of [22], we are now ready to define the set of *typical executions* for this setting.

Definition 35 (Typical execution). An execution is σ-*typical* if and only if for any set R of consecutive rounds with $|R| \geq \frac{\max\{4k_0,\lambda\}}{\gamma\sigma^2}$, the following hold:

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

Theorem 36. *An execution of* Π_{nPL} *is* σ*-typical with probability* $1 - \mathsf{negl}(\lambda)$.

Proof. In order for an execution to not be typical one of the three points of Definition 35 must not hold. For any set of rounds R, such that $|R| \geq \frac{\max\{4k_0,\lambda\}}{\gamma\sigma^2}$, it holds that point 1 is implied by Lemma 34 and point 2 is implied by Lemma 32 and Corollary 33. Lastly, point 3 is implied by Corollary 31. Hence, 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. □

As protocol designers, our goal is to be able to tolerate $t'_{\mathcal{A}}$'s that are as large as possible. On the other hand, our proof strategy is going to be based on the fact that the rate of uniquely successful rounds doubles the rate at which the adversary produces blocks. The tension between these two separate goals is manifested in Assumption 1 (Computational Power Assumption). The next lemma exactly highlights this implication.

Lemma 37. *For any set of rounds R in a typical execution, where $|R| \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}$, and for any block B mined by an honest party during R, it holds that $Z_B(R) \leq (1-\frac{\delta}{4})\frac{Y(R)}{2}$.*

Proof.

$$Z_B(R) \leq \beta t'_{\mathcal{A}} \cdot |R| \leq \frac{1}{2} \cdot \frac{1}{1+\delta}\gamma|R| < (1-\frac{\delta}{4})\frac{Y(R)}{2}$$

The first and the last inequality follow from the assumption that the execution is typical. The middle inequality follows from Assumption 1. □

We can now use the machinery built in [22] to prove the common prefix, chain quality and chain growth properties, with only minor changes.

Higher level properties. The notion of a typical execution is at the core of the proof of security of Bitcoin in [22]. Next, we describe the changes one has to do after proving the typical execution theorem with respect to the analysis of [22], in order to prove the security of the protocol in our model. We only give brief proof sketches of lemmas and theorems from [22] that are exactly the same for our own setting.

Lemma 38. *(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 39. *(Chain-Growth). In a typical execution the chain-growth property holds with parameters $\tau = (1-\frac{\sigma}{4})f$ and $s \geq \frac{\max\{4k_0,\lambda\}}{\gamma\sigma^2}$.*

Proof. Let R be any set of at least s consecutive rounds. Then, since the execution is typical: $X(R) \geq (1-\frac{\sigma}{4})f \cdot |R| \geq \tau \cdot |R|$. By Lemma 38, 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 40. *Let B be some honest block in a typical execution. Any sequence of $k \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta nt'_{\mathcal{H}})$ consecutive blocks in some chain $\mathcal{C}$, where the first block in the sequence directly descends B, have been computed in at least $k/(\gamma + \beta nt'_{\mathcal{H}})$ rounds, starting from the round that B was computed.*

Proof. For some $k \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta nt'_{\mathcal{H}})$, assume there is a set of rounds R' , such that $|R'| < k/(\gamma + \beta nt'_{\mathcal{H}})$, and more than k blocks that descend block B have been computed. Then, there is a set of rounds R, where $|R| \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}$, such that $X_B(R) + Z_B(R) \geq k \geq |R|(\gamma + \beta nt'_{\mathcal{H}}) \geq |R|\beta(t'_{\mathcal{A}} + nt'_{\mathcal{H}})$. This contradicts the typicality of the execution, hence the lemma follows. □

Lemma 41. *(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 $k \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta nt'_{\mathcal{H}})$.*

Proof. The proof in [22] 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 [22], reaching a contradiction with

Lemma 37. 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 42. *(Common-prefix). In a typical execution the common-prefix property holds with parameter* $k \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta n t'_{\mathcal{H}})$.

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 43. *(Chain-Quality). In a typical execution the chain-quality property holds with parameter* $\mu < 1 - \delta/4$ *and* $\ell \geq \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta n t'_{\mathcal{H}})$.

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 35. 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 [22].

Lemma 44. *(Persistence). It holds that* $\Pi_{\mathsf{nPL}}(\mathcal{I})$ *with* $k = \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma+\beta n t'_{\mathcal{H}})$ *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 45. *(Liveness). It holds that* $\Pi_{\mathsf{nPL}}(\mathcal{I})$ *with* $k = \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta n t'_{\mathcal{H}})$ *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. 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. □

Using these properties we prove that protocol $\Pi_{\mathsf{nPL}}(\mathcal{I})$ implements a robust transaction ledger. Note that both Persistence and Liveness depend on the convergence parameter k_0 of $\mathcal{I}$.

Theorem 46. *Assuming the existence of a common reference string, a collision-resistant hash function family, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), an enhanced ISP problem* $\mathcal{I}$*, and model parameters* $\{n, t, t_{\mathcal{H}}, t_{\mathcal{A}}, \theta\}$ *that comply with Assumption 1, protocol* $\Pi_{\mathsf{nPL}}(\mathcal{I})$ *implements a robust public transaction ledger with parameters* $k = \frac{\max\{4k_0,\lambda\}}{\gamma\delta^2}(\gamma + \beta n t'_{\mathcal{H}})$ *and* $u = \frac{2k}{(1-\frac{\delta}{4})f}$*, except with negligible probability in* λ.

4.3 A candidate enhanced ISP

We now present an ISP problem that is plausibly hard against precomputation (Definition 18), and satisfies *all* other properties of an enhanced ISP (Definition 28).

Construction 1. Let $\mathcal{H}_d$ be collision resistant hash function family, parameterized by $d \in \mathbb{N}$, and $T \in \{0,1\}^\lambda$ be some hardness parameter. An instance of an enhanced ISP is as follows:

- I_λ is the uniform distribution over functions $H : \{0,1\}^{(d+1)\lambda} \to \{0,1\}^\lambda$ in $\mathcal{H}_d$, i.e., $\Lambda = \{H\}$;
- $X = \{0,1\}^\lambda, W = \{0,1\}^{2d\lambda}$;
- $R = \{(x,w) | H(x||w_1) < T \text{ for } w = w_1||w_2\}$;
- $M(x)$ iteratively samples w_1 from $\mathcal{U}_{d\lambda}$, and tests whether $H(x||w_1) < T$, until it finds a solution. Then, it samples a uniformly random w_2 from $\mathcal{U}_{d\lambda}$, and outputs $w_1||w_2$.
- $S(x,w) = H(H(x||w_1)||w_2)$.

Construction 1 is similar to Bitcoin's PoW construction (see Section 3.2), with the following differences. The first is that, in our construction $H(x||w_1)$ is required to be smaller than the hardness parameter T, while in Bitcoin $H((x||w_1)||w_2)$ is expected to be small, where w_1 there 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. The second difference is that 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 $T/2^\lambda$. 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 about the security of the construction.

Regarding iterated hardness, to our knowledge no reductions exist (yet) of iterative hardness to weaker assumptions in the standard model. We argue the plausibility of our construction being hard against precomputation, by the fact that Construction 1 is based on Bitcoin's ISP construction, for which no attacks are known.

Regarding the additional security properties that make up an enhanced ISP, assuming H is a randomness extractor, all properties presented in Section 4.1 are satisfied. First, the fact that $H(x||w_1)$ is statistically indistinguishable from uniform, for any $x \in X$, implies the runtime independence property. By the same argument, the solver M succeeds in each test with probability $T/2^\lambda$ independently of the previous tests. This implies that the running time of the solver can be simulated and is independent of the input x. Further, a lower bound on the probability of success after any number of steps can be computed, as required by the successful property. Finally, since w_2 is also chosen uniformly at random, the next problem produced by S will also look uniformly random, which implies the next-problem simulatability property, and by resampling w_2 uniformly at random, the witness malleability property also follows. We are thus able to state:

Lemma 47. *Let $E : \{0,1\}^{(c+1)\lambda} \times \{0,1\}^{(d-c)\lambda} \to \{0,1\}^\lambda$, where $E(x,i) \stackrel{\triangle}{=} H(x||i)$, the hash function used in Construction 1, and $c \in [1,d]$. If E is an $(c\lambda, \mathsf{negl}(\lambda))$-extractor, then Construction 1 is almost runtime independent, $O(\lambda)$-next-problem simulatable, $O(\lambda)$-witness malleable, and $(t, \mathcal{C}_{T/2^\lambda}(O(t)))$-successful for any $t \in poly(\lambda)$, where $\mathcal{C}_{T/2^\lambda}$ is the cumulative geometric distribution with parameter $T/2^\lambda$.*

Proof. Fix some $x_1, \ldots, x_m \in X$, for $m = poly(\lambda)$, and define the following random variables: $Z_{x_i} = E(x_i||\mathcal{U}_{c\lambda}, \mathcal{U}_{(d-c)\lambda}), Y_i = \mathcal{U}_\lambda$. Since E is a $(c\lambda, \mathsf{negl}(\lambda))$-extractor, and $x_i||\mathcal{U}_{c\lambda}$ has $c\lambda$ bits of entropy, it follows that $\Delta[Z_{x_i}, Y_i] \leq \mathsf{negl}(\lambda)$. Moreover, by definition the random variables $\{Z_{x_1}, \ldots, Z_{x_m}, Y_1, \ldots, Y_m\}$ are mutually independent. By a hybrid argument, this implies that

$$\Delta[(Z_{x_1}, \ldots, Z_{x_m}), (Y_1, \ldots, Y_m)] \leq \mathsf{negl}(\lambda)$$

Now, assume that $(x_1, \ldots, x_m)$ are sampled instead by some distribution $\hat{\mathcal{X}}$, as it will be the case in an actual execution, and let $(\hat{X}_1, \ldots, \hat{X}_m)$ be the random variable produced by first sampling from

$\hat{\mathcal{X}}$, and the applying $E(\cdot||\mathcal{U}_{c\lambda},\mathcal{U}_{(d-c)\lambda})$ for each x_i. It holds that

$$\Delta[(\hat{X}_1,\ldots,\hat{X}_m),(Y_1,\ldots,Y_m)] \leq \max\{\Delta[(Z_{x_1},\ldots,Z_{x_m}),(Y_1,\ldots,Y_m)]\}_{(x_1,\ldots,x_m)\in X^m} \leq \mathsf{negl}(\lambda)$$

Algorithm M on input x, iteratively samples a uniformly random $w_1||w_1'$ from $\mathcal{U}_{c\lambda}\times\mathcal{U}_{(d-c)\lambda}$, and tests whether $H(x||w_1||w_1') = E(x||w_1,w_1') < T$, until it finds a solution. The number of steps that M takes, is thus a function of $(Z_{x_1},\ldots,Z_{x_m})$, for $x_1=\ldots=x_m=x$ and $m\in poly(\lambda)$. If at each step M tested whether a value sampled from $\mathcal{U}_\lambda$ is smaller than T, its' running time would be distributed according to the geometric distribution $\mathcal{G}_p$ with parameter $p=T/2^\lambda$. By our arguments above, the statistical distance of $\mathsf{Steps}_M(x)$ from $c_1\cdot\mathcal{G}_{T/2^\lambda}+c_2$ should be negligible close, where c_1 is a small constant related to the cost of sampling a random value for each test and evaluating E, and c_2 to the cost of sampling w_2. As before, we can extend this result to multiple random variables with inputs sampled under some distribution $\hat{\mathcal{X}}$, i.e.,

$$\Delta[(Steps_M(\hat{X}_1),\ldots,Steps_M(\hat{X}_m)),(c_1\cdot\mathcal{G}_{T/2^\lambda}+c_2,\ldots,c_1\cdot\mathcal{G}_{T/2^\lambda}+c_2)] \leq \mathsf{negl}(\lambda)$$

Hence, the runtime independence property follows.

Next, note that M, after finding a small hash, it hashes again the result with a fresh randomly sampled string w_2. By the extractor property, it is implied that for any $x\in X$

$$\begin{aligned}\Delta[S(x,M(x)),\mathcal{U}_\lambda] =& \Delta[E(E(x||\mathcal{U}_{c\lambda},\mathcal{U}_{(d-c)\lambda})||\mathcal{U}_{c\lambda},\mathcal{U}_{(d-c)\lambda}),\mathcal{U}_\lambda]\\ \leq& \max\{\Delta[E(x'||\mathcal{U}_{c\lambda},\mathcal{U}_{(d-c)\lambda}),\mathcal{U}_\lambda]\}_{x'\in X} \leq \mathsf{negl}(\lambda)\end{aligned}$$

Hence, for the simulator $\mathcal{S}$ 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 simuleatability property. Note, that using the inverse transform technique, we can sample from the geometric distribution (truncated to 2^λ) in $O(\lambda)$ steps.

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 uniform.

Finally, the probability that for any input x, $M(x)$ outputs a witness after t steps, is negligibly close to the probability that the output of the geometric distribution is smaller or equal to $\frac{t-c_1}{c_2} = O(t)$. Hence, our scheme satisfies the successful property. □

5 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/`.

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

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

[32] 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.

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

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

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

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

[41] 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/.