On the Foundations of Proof-of-Work Based Blockchain Protocols

Georgios Panagiotakos

Doctor of Philosophy

Laboratory for Foundations of Computer Science

School of Informatics

University of Edinburgh

2020

Abstract

Proof-of-work (PoW) based blockchain protocols, are protocols that organize data into blocks, connected through the use of a hash function to form chains, and which make use of PoW to reach agreement, i.e., proofs that require spending some amount of computational power to be generated. This type of protocols rose into prominence with the advent of Bitcoin, the first protocol that provably implements a distributed transaction ledger against an adversary that controls less than half of the total computational power in the network, in a setting where protocol participants join and leave dynamically without the need for a registration service. Protocols in this class were also the first to be shown sufficient to solve consensus under similar conditions, a problem of fundamental importance in distributed computing. In this thesis, we explore foundational issues of PoW-based blockchain protocols that mainly have to do with the assumptions required to ensure their safe operation.

We start by examining whether a common random string that is shared at the start of the protocol execution among the protocol participants is required to efficiently run such protocols. Bitcoin's security is based on the existence of such a string, called the genesis block. On the other hand, protocols found in previous works that do not assume such a setup are inefficient, in the sense that their round complexity strongly depends on the number of protocol participants. Our first contribution is the construction of efficient PoW-based blockchain protocols that provably implement a distributed ledger and consensus without such setup.

Next, we turn our attention to the PoW primitive. All previous analyses model PoW using a random oracle. While satisfactory as a sanity check, the random oracle methodology has received significant criticism and shown not to be sound. We make progress by introducing a non-idealized security model and appropriate computational assumptions that are sufficient to implement a distributed ledger or consensus when combined with the right PoW-based protocol.

Finally, we analyze GHOST, a recently proposed blockchain protocol, and prove its security against a byzantine adversary under similar assumptions as Bitcoin. Previous works only considered specific attacks.

To my parents
Vassili and Despoina

Acknowledgements

The development of this thesis would not have been possible without the company and help of many people. I would like to name and thank a few of them.

Firstly, I would like to thank Aggelos Kiayias for his scientific and psychological support all these years; a PhD cannot even begin without establishing a solid student-advisor relation. In addition, I would like to thank Juan Garay for his scientific advice as well as his philosophical insights on life and the universe.

Furthermore, I would like to thank my two examiners Stefan Dziembowksi and Markulf Kohlweiss for the interesting discussion we had during my PhD examination. Many other people from the university staff also deserve a thank you for their patience, especially Kousha Etessami and Myrto Arapinis who helped with my annual reviews.

I would also like to thank my fellow (former) PhD students Yiannis Tselekounis and Thomas Zacharias, as they helped me immensely on maturing scientifically. Many thanks also to all other people with which we had many interesting scientific and non-scientific discussions.

Finally, I would like to thank my family and friends for their continuous support all these years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Agreement among a number of geographically separated and distrustful entities is a fundamental problem of human societies that goes back to the first methods devised for voting. With the rise of computer networks, especially the Internet, the problem has been reconsidered in a setting where communication is performed over an untrusted medium and parties use keys issued by a certification authority to establish the authenticity of their messages. An important application of this paradigm is secure electronic payments and in general e-commerce. However, the need for powerful centralized entities, such as the certification authority, has made this paradigm prone to abuse as seen for example in the case of Wikileaks [LE10].

An alternative proposal was recently put forth with the release of the Bitcoin protocol. Bitcoin implements a distributed payment system, which in contrast to most previously known such systems, does not need a central authority to operate. Instead, it relies on a distributed protocol to establish an ever-growing common history of transactions. Most interestingly, it is the first such system that can operate in a *permissionless* setting, where parties can join or leave dynamically without the need of a registration service.[1] To do so, it neatly combines many older technologies in a entirely new paradigm, including: digital signatures [DH76], Merkle trees [Mer87], proof-of-work (PoW) [DN93], and hash chains [Dam89, Mer89].

The main idea behind the protocol is that transactions are organized in blocks, that are connected though the use of a hash function to form chains, which parties stochas-

[1] In comparison, older protocols assumed that protocol participants were registered, e.g., through a PKI. Note, that this is also the case for Proof-of-Stake based protocols [KRDO17, CM19].

tically try to extend by computing a PoW, i.e., a proof that they spent some amount of computational power in a specific time interval. At any point, the history of accepted transactions is decided by the blockchain that contains the most work. While there may be some disagreement about which blockchain is that, due to network delay and adversarial manipulation, it has been argued that if the majority of the computational power is controlled by honest participants everyone will eventually agree on the same history.

These arguments were formalized in a series of papers [GKL15, PSS17, GKL17, ES14, ML14, BGM$^+$18, BMTZ17], starting from the publication of the Bitcoin whitepaper [Nak09] in 2008. In addition, there has been significant work on many more PoW-based blockchain protocols [AD15, SZ15, BKT$^+$19, BCG$^+$14, EGSvR15], i.e., protocols that organize data into hash chains and make use of PoW to reach agreement. In particular, in [GKL15] a protocol in this class was the first to be shown sufficient to solve consensus, a well-studied problem of central importance in the theory of distributed computing [PSL80, LSP82, DLS88, FLP85, Bor96], in the permissionless setting.

In spite of all this progress, our understanding of the assumptions that this class of protocols relies on to operate securely remains incomplete, including the following issues:

- They either rely on the existence of a public trusted setup that becomes available to all parties at the start of the protocol execution, as Bitcoin, or are inefficient [AD15], in the sense that the round complexity of the protocol strongly depends on the number of protocol participants.

- They are analyzed in idealized computational models, thus obscuring the required computational assumptions regarding the underlying PoW primitive.

- Finally, while there have been proposals for different blockchain protocols, e.g., [SZ15], it is unclear whether they offer any real advantage compared to Bitcoin, or if they open up the space for new attacks.

In this thesis, we attempt to shed some light to these shortcomings, by analyzing existing protocols and proposing alternatives.

1.2 Overview and Contributions

Next, we give an overview of each of the issues mentioned above and our respective contributions. To aid understanding, we first give a summary of the results in [GKL15], the first formal analysis of Bitcoin against a byzantine adversary, which serves as the basis from which we develop our results.

Summary of [GKL15]. The core of the Bitcoin protocol, called the Bitcoin backbone, is extracted and analyzed in a synchronous, permissionless and Byzantine setting, where message delivery is provided by a "diffusion" mechanism that is guaranteed to deliver all messages, without however preserving their order and allowing the adversary to arbitrarily inject its own messages. The number of parties is assumed to be fixed but unknown, and to take advantage of PoWs parties are assumed to have bounded access to a hash function modeled in the Random Oracle Model (ROM) [BR93].

The analysis includes the formulation of fundamental properties of the underlying blockchain data structure, which parties maintain and try to extend by generating PoWs, called common prefix and chain quality. It is then shown how applications such as consensus (aka Byzantine agreement) [PSL80, LSP82] and a robust public transaction ledger, i.e., one that satisfies persistence of transactions entered in the ledger and transaction liveness, can be built "on top" of such properties. All results are proven under the assumption that: (i) the adversary controls strictly less than half of the hashing power, (ii) the block generation rate is suitably bounded, and (iii) all parties—honest and adversarial—"wake up" and start computing at the same time, or equivalently, there exists a common random string, called the genesis block in the context of Bitcoin, which is only made available at the exact time when the protocol execution is to begin.

1.2.1 Trusted Setup

Bitcoin assumes the existence of "fresh" public trusted setup, in order to cope with adversaries trying to precompute blocks long before the start of the protocol, and break its security. While satisfactory in some cases, such a trusted setup might be unrealistic in other PoW-based systems where details may have been released a lot earlier than the actual time when the system starts to run. That's from a practical point of view. At a foundational level, one would in addition like to understand what kind of cryptographic primitives can be realized without any trusted setup assumption and based on PoWs, and whether that is in particular the case for the Bitcoin functionality and its enabling properties mentioned above.

The former question was previously considered by Andrychowicz and Dziembowski [AD15], who, building on previous suggestions by Aspnes *et al.* [AJK05] of using PoWs as an identity-assignment tool and constructions by Fitzi *et al.* [Fit03, CFF$^+$05] showing how to morph "graded" consistency into global consistency, showed how to create a consistent PKI using PoWs and no other trusted setup (in the ROM), which can then be used to run secure computation protocols (e.g., [Yao82, GMW87]) and realize any cryptographic functionality assuming that the majority of the computational power is controlled by honest parties.

While this in principle addresses the foundational concerns, it leaves open the questions of doing it in a *scalable* way—i.e., with round complexity independent of the number of parties.

Contribution. In Chapter 3 we answer the question stated above in the affirmative. First, we present a Bitcoin-like protocol that neither assumes a simultaneous start nor the existence of an unpredictable genesis block, and has round complexity independent of the number of participants, despite the adversary being allowed (polynomial in the security parameter) precomputation time.

Second, we present applications of our bootstrapped construction, starting with its original one: a distributed ledger, despite the potentially disruptive behavior of parties harnessing less than $1/2$ of the hashing power. Next, we consider the problem of setting up a PKI in our unauthenticated network setting *from scratch*. As mentioned above, the idea of using PoWs as an identity-assignment tool was put forth by Aspnes *et al.* [AJK05]. Here we build on this idea as well as on the "2-for-1 PoWs" technique from [GKL15] to use our bootstrapped protocol to assign identities to parties. The assignment relation will possibly assign more than one identity to the same party, while guaranteeing that the majority of them is assigned to honest parties. Such an identity infrastructure/"pseudonymous PKI" has numerous applications, including the bootstrapping of a proof-of-stake protocol [KRDO17, KN12], and the election of honest-majority "subcommittees," which would enable the application of traditional Byzantine fault-tolerant techniques for ledger creation and maintenance (cf. [ACC$^+$17]) to permissionless (as opposed to permissioned) networks.

Finally, applying the 2-for-1 PoWs technique we solve the consensus (aka Byzantine agreement) problem [PSL80, LSP82] probabilistically and from scratch, even if the adversary has almost the same hashing power as the honest parties[2], and with round

[2]Thus marking a contrast with the $\frac{2}{3}$ lower bound for consensus on the number of honest parties in

complexity independent of the number of parties. Indeed, all our protocols have round complexity linear in the security parameter, and enjoy simultaneous termination.

Subsequent Work: In an unpublished manuscript [EFL17], Eckey et al., show how to achieve broadcast and consensus without trusted setup in the ROM and in a quasi-constant expected number of communication rounds, assuming honest parties control the majority of the computational power and the non-parallelizability of repeated squaring, thus improving on the round complexity of our protocol.

1.2.2 Proof-of-Work

Blockchain protocols are seen as a prominent application of the "proof of work" (PoW) concept to the area of consensus protocol design. In more detail, a PoW scheme is typified by a "proving" and a "verification" algorithm, whose fundamental property 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. The concept was introduced in the work of Dwork and Naor [DN93] initially as a spam protection mechanism. Subsequently, it found applications in other domains such as Sybil attack resilience [Dou02a] and denial of service protection [JB99, Bac02], prior to its application to the domain of distributed consensus hinted at early on by Aspnes *et al.* [AJK05].

Despite the evolution of our understanding of the PoW primitive, as exemplified in [AT17, BRSV18, BGJ$^+$16], there has been no definitive analysis of the primitive in the context of blockchain protocol security. Intuitively, PoWs are useful for agreement because they make message passing (moderately) hard and hence generate stochastic opportunities for the parties running the protocol to unify their view of the current state of the system. This fundamentally relies on an assumption about the aggregate computational power of the honest parties, but not on their actual number, in relation to the computational power of the parties that may deviate from the protocol ("Byzantine")—a hallmark of the permissionless setting where Bitcoin is designed for. Despite the fact that the Bitcoin blockchain has been analyzed formally [GKL15, PSS17, GKL17, BMTZ17], the required PoW properties have not been identified and the analysis has been carried out in the Random Oracle Model [BR93]. The same is true for a wide variety of other protocols in the space, including [AD15, KMS14, GKLP18].

We stress that although the RO model has been extensively used in the security

the traditional network setting with no setup [Bor96].

analysis of practical protocols and primitives, it has also received significant criticism [CGH04, Nie02, GK03, FS86], with the RO methodology shown not to be sound. Therefore, it is critical to discover security arguments for PoW-based blockchain protocols that do not rely on the RO model.

Contribution. In Chapters 4 and 5 we make considerable progress in understanding and formalizing the security of PoW schemes that are suitable for blockchain and consensus protocols.

First, in Chapter 4, we note that all attempts to formalize a building block that is sufficient for designing consensus protocols in this setting, rely on a very strong independence assumption about adversarial accesses to the underlying computational resource. We relax this assumption by putting forth a primitive, which we call *signatures of work* (SoW). Distinctive features of our new notion are *moderate unforgeability*—producing a sequence of SoWs, for chosen messages, does not provide an advantage to an adversary in terms of running time; and *honest* signing time independence—most relevant in concurrent multi-party applications, as we show.

Then, we present a new security model, that is based on that of [GKL15], and allows for a security analysis that does not rely on idealized assumptions, such as the RO. Armed with SoW and the new security model, we present a new permissionless consensus protocol which is secure assuming an honest majority of computational power, thus improving on the known provably secure consensus protocols in this setting, which rely on the strong independence property mentioned above in a fundamental way.

Next, in Chapter 5, we identify a new class of search problems, *iterated search problems* (ISP), which are closely related to the design of secure blockchain protocols. We prove that (i) the Bitcoin blockchain protocol implies a hard ISP problem, but ISP hardness is not by itself sufficient to prove its security, and (ii) a suitably enhanced class of ISPs is sufficient to imply, via construction, a blockchain protocol secure against an adversary that controls up to $1/3$ of the computational power in the common reference string (CRS) model; for the latter we also assume the existence of a non-interactive zero-knowledge (NIZK) proof system and a collision resistant hash family.

We then put forth a specific proposal for an enhanced ISP based on a cryptographic hash function, whose hardness is closely related to that of Bitcoin's hash-based ISP construction for which no attacks have been discovered. As a corollary, we obtain a blockchain protocol whose security reduces to the existence of a NIZK proof system and simple properties of a hash function: collision resistance, the ISP hardness of the hash-based scheme and a weak computational randomness extraction property of the hash function. Finally, we note that all hash properties are efficiently falsifiable [Nao03, GW11], i.e., they can be expressed as a game between an adversary and a challenger, where the challenger can efficiently determine (and output) when the adversary wins the game.

1.2.3 Alternative Blockchain Protocols

A fundamental open problem in the area of blockchain protocols is whether the Bitcoin protocol is the optimal solution for building a secure transaction ledger. There are many reasons to believe that this is not the case, e.g., the selfish mining attack against fairness in [ES14], the attack against common prefix and chain quality for increased block generation rates in [PSS17].

A recently proposed and widely considered alternative is the `GHOST` protocol [SZ15] which, notably, was proposed to be at the core of Ethereum as well as other recent proposals for improved Bitcoin-like systems. While the `GHOST` variant is touted as offering superior performance compared to Bitcoin without a security loss, previous analyses only consider security against specific attacks.

Contribution. In Chapter 6, we study the security of the `GHOST` protocol against a byzantine adversary. We introduce a new formal framework for the analysis of blockchain protocols that relies on trees (rather than chains) and we showcase the power of the framework by providing a unified description of the `GHOST` and Bitcoin protocols, the former of which we extract and formally describe. We then prove that `GHOST` implements a "robust transaction ledger" under the same assumptions as Bitcoin, and hence it is a provably secure alternative to it; moreover, our bound for the liveness parameter is superior to that proven for the Bitcoin backbone in line with the original expectation for `GHOST`. Our proof follows a novel methodology for establishing that `GHOST` is a robust transaction ledger compared to previous works, which may be of independent interest and can be applicable to other blockchain variants.

Subsequent work. In [KRs18], it is shown that `GHOST` satisfies consistency, a property similar to common prefix, in a model where the adversary can delay messages for up to Δ rounds. On the negative side, an attack is presented that breaks consistency in the short-term, for high block generation rates. As no asymptotic bounds are derived, it remains unclear if the effects of the attack persist in the long-term. More generally, designing high performance PoW-based blockchain protocols is an actively explored research direction, we point to [PS18, BKT$^+$19] for some recent developments.

1.3 Roadmap

The thesis is organized as follows: First, in Chapter 2, we present the basic notation, security model, and security definitions that are going to be used throughout the text. In Chapters 3,4,5 and 6 we present our results. Each of these chapters contains a detailed introduction of the specific problem addressed there as well as related work, limitations and future directions. In more detail, in Chapter 3, a protocol that implements an efficient ledger without trusted setup is presented and analyzed in the ROM. The Signature-of-Work primitive and a non-idealized security model is presented in Chapter 4, together with a protocol that implements consensus in this model. In Chapter 5, the notion of a hard Iterated Search Problem (ISP) is introduced and is shown to be necessary, but not sufficient, to prove that the Bitcoin protocol implements a transaction ledger. A sufficient enhanced ISP security notion is also introduced in the same chapter as well as a candidate instantiation based on Bitcoin's underlying hash-based PoW construction. Next, in Chapter 6, the `GHOST` protocol is analyzed against a byzantine adversary in the ROM, and is shown to be secure under the same assumptions as Bitcoin. Concluding, in Chapter 7 we present how some of the main ideas of this thesis were developed. Finally, in Appendix A we present a number of mathematical facts that are going to be used in the thesis, while in Appendix B we present the security definitions of known cryptographic primitives that we are going to use.

Chapter 2

Preliminaries

2.1 Notation

We introduce basic notation and definitions that are used in the rest of this thesis. For $k \in \mathbb{N}^+$, $[k]$ denotes the set $\{1,\ldots,k\}$. For strings x,z, $x||z$ is the concatenation of x and z, and $|x|$ denotes the length of x. We denote sequences by $(a_i)_{i\in I}$, where I is the index set which will always be countable. For a set X, $x \leftarrow X$ denotes sampling a uniform element from X. For a distribution $\mathcal{U}$ over a set X, $x \leftarrow \mathcal{U}$ denotes sampling an element of X according to $\mathcal{U}$. By $\mathcal{U}_m$ we denote the uniform distribution over $\{0,1\}^m$. 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 $\varepsilon > 0$, we say that X,Y are ε-*close* when $\Delta(X,Y) \leq \varepsilon$. We denote by $H_\infty(X) = \log(\max\{\Pr[X=s]|s \in U\}^{-1})$ the min-entropy of X.

We call a function $\nu : \mathbb{N} \to \mathbb{R}$ negligible, if for every $c \in \mathbb{N}$, there exists a κ_0, such that for all $\kappa > \kappa_0$: $\nu(\kappa) < 1/\kappa^c$. We will sometimes denote that some function f is negligible by $f(\kappa) < \mathsf{negl}(\kappa)$. It holds that for every negligible function $\nu(\cdot)$, the function $\nu(\cdot) \circ \mathsf{poly}(\cdot)$ is negligible; we use $\mathsf{poly}(\cdot)$ for some fixed, but unspecified polynomial.

A Probabilistic Polynomial Time (PPT) algorithm, is one that always halts after a polynomial, in the length of the input, number of steps independently of its internal coin tosses. By $y \leftarrow M(x)$, we denote the execution of randomized algorithm M on input x, with output y. $M(x;r)$ denotes the execution of M on input x with randomness r. A non-uniform algorithm is a pair $(M,(a_i)_{i\in\mathbb{N}})$, where M is an algorithm in some computational model and $(a_i)_i$ is an infinite sequence of strings where $|a_i| = \mathsf{poly}(i)$. For every x, the computation of M is on input $(x, a_{|x|})$.

An ensemble $(X_i)_{i\in I}$, is a sequence of random variables indexed by I. Two ensembles $X = (X_\kappa)_{\kappa\in\mathbb{N}}, Y = (Y_\kappa)_{\kappa\in\mathbb{N}}$ are statistically indistinguishable if the function $\mu(\kappa) \overset{def}{=} \Delta[X_\kappa, Y_\kappa]$ is negligible. X, Y are computationally indistinguishable, if for every non-uniform PPT algorithm [1] D, there exists a negligible function $\nu(\cdot)$, such that for all $\kappa \in \mathbb{N}$:

$$|\Pr[D(1^\kappa, X_\kappa) = 1] - \Pr[D(1^\kappa, Y_\kappa) = 1]| < \nu(\kappa)$$

We use $X \approx Y$ and $X \overset{c}{\approx} Y$ to denote statistical and computational indistinguishability, respectively. We point to [Gol06] for more details about the notions of computational indistinguishability, PPT and non-uniform algorithms.

A *blockchain*, or simply a *chain* is a sequence of *blocks*, where a block is a specifically formated string that is going to be defined differently for the different protocols presented. The rightmost block is the *head* of the chain, denoted $\text{head}(\mathcal{C})$. Note that the empty string ε is also a chain; by convention we set $\text{head}(\varepsilon) = \varepsilon$. For some block B and some chain $\mathcal{C}$, we denote by $\mathcal{C}B$ the chain resulting by extending $\mathcal{C}$ by B, where $\text{head}(\mathcal{C}B) = B$. Consider a chain $\mathcal{C}$ of length m ($\text{len}(\mathcal{C}) = m$) and any nonnegative integer k. We denote by $\mathcal{C}^{\lceil k}$ the chain resulting from the "pruning" of the k rightmost blocks. Note that for $k \geq \text{len}(\mathcal{C})$, $\mathcal{C}^{\lceil k} = \varepsilon$. If $\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$ we write $\mathcal{C}_1 \preceq \mathcal{C}_2$.

2.2 Security Model

In this section, we describe the security model that we are going to use throughout most of this work. Our model is a variant of the synchronous model presented in [GKL15] for the analysis of the Bitcoin backbone protocol, which in turn is based on Canetti's formulation of "real world" execution for multi-party cryptographic protocols [Can00, Can01].

Computational model. An execution of some protocol Π is defined with respect to an "environment" program $\mathcal{Z}$, a "control" program C, and an "adversary" program $\mathcal{A}$. At a high level, $\mathcal{Z}$ is responsible for providing inputs to and obtaining outputs from different instances of Π, C is responsible for supervising the spawning and communication of all these programs, and $\mathcal{A}$ aims to disrupt the goals set by the protocol. All these programs are formulated as "interactive Turing machines" (ITM) communicating through shared tapes. In this setting, "resource bounded computation" is captured by requiring that the length of the execution of $\mathcal{Z}$ is "bounded" by some polynomial

[1] It will always be clear from the context the computational model under which D is specified.

in the security parameter, and that C is polynomial time computable. Proposition 3 in [Can01] guarantees that if these conditions are satisfied, the whole execution can be simulated by a polynomial time Turing machine.

We will consider executions where the set of of parties $\{P_1, ..., P_n\}$ is fixed and hardcoded to C. Moreover, we consider a "hybrid" model of computation [CF01], where the adversary $\mathcal{A}$ as well as all parties in the execution can access a number of "ideal" functionalities as subroutines; the functionalities are also modeled as ITMs and will be presented later in detail. Initially $\mathcal{Z}$ is activated. $\mathcal{Z}$ can make special requests that result in the spawning of different parties and $\mathcal{A}$. In turn, $\mathcal{A}$ can corrupt different parties by sending messages of the form (Corrupt, P_i) to C, with the limitation that the total number of parties corrupted should be at most t; t is a parameter of the execution. We assume an active or "byzantine" adversary that is activated in the place of corrupted parties and is free to perform whatever actions it wants to.

Depending on the chapter, we will also explicitly specify whether we consider a static adversary that chooses which parties to corrupt at the beginning of the execution, or an adaptive one that chooses them dynamically throughout the execution.

Communication and Synchrony. We are going to assume a *synchronous setting* of computation, where the current round is known to all parties, and messages sent at one round are received at the beginning of the next one. In the classical (permissioned) setting synchronous computation can be simulated assuming that there is a known upper bound on network delays, and that the clocks of honest parties are loosely synchronized, see [DLS88] for a fault-tolerant implementation. While there have been efforts to do the same in the permissionless setting [BGK$^+$19, FGKR18], we are not going to explore this question further in this work.

The influence of the adversary in the network is going to be actively malicious, following the standard cryptographic modeling. Firstly, we assume that the adversary is rushing, in the sense that it is immediately informed about the messages sent by honest parties. Message transmission is handled by a "diffusion" functionality, that ensures the integrity of the messages sent, but does not provide any guarantees about the transmission order or the sender of each message, i.e., communication is not authenticated.[2] Moreover, the adversary will also be allowed to send messages to subsets of the party set, with the consequence that the view of honest parties is not guaranteed to be the same.

All the above concerns are captured by the diffusion functionality $\mathcal{F}_{\text{DIFF}}$. The func-

[2]In practice, such a functionality is implemented through a gossiping protocol, e.g., see [FG10].

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

Following [GKL15], we choose to have the network functionality also handle synchrony. A more modular approach was followed in [BMTZ17], using both a network and a clock functionality. We note, that all our results can be easily adapted to work in that model.

Precomputation and Trusted Setup. In contrast to [GKL15], where all parties are spawned at the first round, in our model the environment may choose to spawn *all* parties later in the execution.[3] This change allows us to capture adversarial precomputation since it permits the environment to activate the adversary an arbitrary number of times (bounded by a polynomial in the security parameter κ of course) before the honest parties get activated.

This distinction is not important for the analysis of Bitcoin, which anyway relies on the existence of a genesis block [Bit], i.e., a "fresh" common reference string (CRS) sampled from a high min-entropy distribution that is received by all parties to render any adversarial precomputation useless. However, it becomes important in Chapter 3, where we implement a transaction ledger without relying on a fresh CRS. In any case, at each chapter we explicitly specify whether we assume a CRS or not.

Resource Restricted Computation. In order for the PoW mechanism to be useful, we need to restrict the computational resources of the machines participating in the computation, in an exact (compared to asymptotic) manner. Here we present one way to do this in the Random Oracle Model (ROM), originally proposed in [GKL15]. Note,

[3] We note that the diffusion functionality will not wait for honest parties to be marked as finished to proceed to the next round before they are spawned. Thus, it is possible that the adversary will be the only one activated for some rounds.

that in Chapter 4 we propose an alternative way to formulate resource restricted computation, without resorting to idealized models.

The adversary $\mathcal{A}$ and all parties have access to a Random Oracle functionality [BMTZ17]. The control program C limits the access to the functionality in the following manner: Each honest party can make up to q queries per round, while the adversary can make $t' \cdot q$ queries, where t' is the number of parties it has corrupted and q is a parameter of the execution. Unlike the adversary, honest parties are also allowed to perform verification queries for free [4], i.e., queries that verify the hash value of some message. This assumption is made to avoid trivial adversaries that perform Denial-of-Service attacks by spamming honest parties with incorrect hash values. It is justified heuristically, since this type of attacks is currently prevented in Bitcoin and relevant systems; we point to [Section 3.4, [AD15]] for extensive discussion on this issue. This is called the q-bounded ROM.

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^κ, for $\kappa \in \mathbb{N}$. For simplicity, we denote this random variable ensemble by $\text{VIEW}^{P,t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$. 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 random variables are defined depends on the coins of all honest parties, $\mathcal{A}$,$\mathcal{Z}$ and the coins of any randomized functionality assumed, e.g., the RO functionality.

The statement we are going to prove about a property Q of the execution is of the following form:

Definition 1. Given a predicate Q and bounds $t,n \in \mathbb{N}$, with $t < n$, we say that protocol Π satisfies property Q for n parties assuming the number of corruptions is bounded by t in the q-bounded ROM, provided that for all PPT $\mathcal{Z},\mathcal{A}$, the probability that $Q(\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}})$ is false is negligible in κ.

Other Modeling Considerations. In the literature, there have been a number of different modeling attempts trying to capture other aspects of the permissionless setting. In [PSS17, AD15], the fully synchronous diffusion functionality, is replaced with one which only guarantees delivery after Δ rounds. In principle, our results can be translated in this setting using ideas found in [Section 7,[GKL15]]. In another

[4] In [AD15], another way of dealing with the DoS problem is presented, by restricting the total number of messages transmitted by the adversary per round.

work [GKL17], the dynamic joining and leaving of new parties is considerer, and it is shown that an adapted version of the Bitcoin Backbone implements a transaction ledger in this setting, assuming that the rate at which the number of active parties changes is bounded. A UC version of the permissionless model is presented in [BMTZ17], where it is shown that Bitcoin can be used to implement an ideal ledger functionality. Finally, a number of works investigate the rationality of Bitcoin and relevant protocols, e.g., see [BGM$^+$18].

2.3 Security Definitions

In this section, we provide the main security definitions that we are going to use in this thesis. We start with three security properties that have to do with the blockchain data structure. Then, we present the notion of a public robust transaction ledger, which captures the main application of the Bitcoin protocol. Finally, we recite the classical consensus definition.

2.3.1 Blockchain Security

We recite a number of basic security properties that have to do with the blockchain data structure, proposed in [GKL15, KP15, PSS17, GKLP18]. These properties concern protocols that maintain a chain of blocks. We note that for the protocols presented in the rest of this thesis, it will always be clear what is the chain that each honest party maintains. Moreover, we call a block that is first diffused in the network by an honest party, an *honest block*.

A definitional contribution of this thesis is the introduction of "initialization" parameters k_0, r_0 on the relevant properties. These parameters are used to signify that a property holds after a certain condition is met, e.g., after r_0 rounds have passed. They are mainly used in Chapter 3, where the blockchain build there requires an initialization phase before the basic blockchain security properties start to hold.

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 parties [GKL15]. Here we will consider a stronger variant of the property, presented in [PSS17], 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—cf. Section 2.3.2).

Definition 2 ((Strong) Common Prefix Property)**.** The *strong common prefix property* Q_{cp} with parameters $k, r_0 \in \mathbb{N}$ 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, with $r_0 < r_1 \leq r_2$, 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 party's chain.

Definition 3 (Chain Quality Property)**.** The *chain quality property* Q_{cq} with parameters $\mu \in \mathbb{R}$ and $k, k_0 \in \mathbb{N}$ 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 $\mathcal{C}$, excluding the first k_0 blocks, the ratio of adversarial blocks is at most μ.

Further, in the derivations in [GKL15] an important lemma was established relating to the rate at which the chains of honest parties were growing as the Bitcoin backbone protocol was run. This was explicitly considered in [KP15] as a property under the name *chain growth.* This property along with common prefix and chain quality were shown sufficient for the black-box proof of application-level properties [PSS17] (in this case, transaction ledger *liveness*; see Section 2.3.2).

Definition 4 (Chain Growth Property)**.** The chain growth property Q_{cg} with parameters $\tau \in \mathbb{R}$ (the "chain speed" coefficient) and $s, r_0 \in \mathbb{N}$ states that for any round $r > r_0$, where honest party P has chain $\mathcal{C}_1$ at round r and chain $\mathcal{C}_2$ at round $r+s$ in $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that $|\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau \cdot s$.

Note, that whenever we omit mentioning r_0 or k_0, it is implied that they are both equal to 0.

2.3.2 Robust Public Transaction Ledger

A *robust public transaction ledger* [GKL15], 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 $\text{tx} \in \mathcal{T}$. Each transaction tx may be associated with one or more *accounts*. The *position* of transaction tx_j in ledger $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, is (i, j), where $x_i = \langle tx_1, \ldots, tx_u \rangle$ for $j \leq u$. Ledgers correspond to chains of transaction blocks in the protocols presented in this thesis.

Transaction ledgers are also defined with respect to an oracle Txgen, which controls a set of accounts and generates transactions on their behalf. In an execution both the environment $\mathcal{Z}$ and the parties will have access to Txgen. Two types of queries can be issued to Txgen:

- $\mathsf{GenAccount}(1^\kappa)$: It generates an account α.
- $\mathsf{IssueTrans}(1^\kappa, \tilde{\mathrm{tx}})$: It returns a transaction tx, provided that $\tilde{\mathrm{tx}}$ is a suitably formed string, otherwise $\perp$.

The definition is parameterized by a symmetric relation $C : \mathcal{T} \times \mathcal{T} \to \{true, false\}$, which indicates whether two transactions are conflicting. Any sequence of non-conflicting transactions constitutes a valid ledger in $\mathcal{L}$. Oracle Txgen is called *unambiguous*, if it holds that for any PPT $\mathcal{A}$, the probability that $\mathcal{A}^{\mathsf{Txgen}}$ outputs two conflicting transactions $\mathrm{tx}, \mathrm{tx}'$, where tx or tx' were generated by Txgen, is negligible in κ. A transaction is called *neutral*, if there does not exist any $\mathrm{tx} \in \mathcal{T}$ that comes in conflict with it. For convenience, we assume that strings of the form $\{0,1\}^\kappa$, are neutral transactions.

Finally, we slightly generalize the definition of [GKL15], by assuming that in the ledgers maintained by our protocols, transactions up to some position are flagged as k-stable, for some $k \in \mathbb{N}$. In Bitcoin, a transaction is k-stable if it is at least k blocks deep in the chain. Other protocols may employ different stability rules. Further, we assume that if a transaction in a ledger is k-stable, then all transactions that precede it in the ledger are also k-stable. In our analyses parameter k will be related to the error probability with which the different properties will hold.

We proceed to give the definition of a robust transactions ledger. The first property of a robust ledger is persistence, which ensures that if a transaction is declared as stable by some party, then all other parties will output this transaction at exactly the same position in the ledger. Persistence implies that the order of stable transactions remains unchanged. The second property, called liveness, guarantees that if a transaction is neutral or is generated by the unambiguous oracle Txgen and is provided as input to all honest parties for long enough, then everyone will declare this transaction as stable.

Definition 5. A protocol Π implements a *robust public transaction ledger* in the synchronous setting, if after some round r_0 the following two properties are satisfied:

- *Persistence:* Parameterized by $k \in \mathbb{N}$ (the "depth" parameter), if in a certain round an honest party reports a ledger that contains a k-stable transaction tx, then tx will always be reported in the same position in the ledger by any honest party 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) neutral, is given as input to all honest parties continuously for u consecutive rounds, then all honest parties will report this transaction as k-stable from the end of this round interval

and on.

Closely related definitions of a robust transaction ledger have been explored in [PSS17, BMTZ17]. An interesting open question is comparing these definitions with the ones found in the study of state machine replication [Sch90], e.g., atomic broadcast [CBPS10, DSU04].

2.3.3 Consensus

Next, we give the definition of the well-known consensus problem (a.k.a. Byzantine agreement) [PSL80, LSP82]. Parties take an initial input $x \in V$ (we assume $V = \{0, 1\}$).

Definition 6. A protocol Π solves the consensus problem provided it satisfies the following properties:

- **Agreement.** All honest parties output the same value, eventually.
- **Validity.** If all honest parties have the same input, then they all output this value.

For a brief history of the problem, we refer the reader to the beginning of Section 4.1.

Chapter 3

Bootstrapping the Blockchain

3.1 Introduction

As the first decentralized cryptocurrency, Bitcoin [Nak09] has ignited much excitment, not only for its novel realization of a central bank-free financial instrument, but also as an alternative approach to classical distributed computing problems, such as reaching agreement distributedly in the presence of misbehaving parties. Formally capturing such reach has been the intent of several recent works, notably [GKL15], where the core of the Bitcoin protocol, called the Bitcoin *backbone*, is extracted and analyzed. The analysis includes the formulation of fundamental properties of its underlying *blockchain* data structure, which parties ("miners") maintain and try to extend by generating "proofs of work" (PoW, aka "cryptographic puzzle" [DN93, RSW96, Bac97, JB99])[1], called *common prefix* and *chain quality*. It is then shown in [GKL15] how applications such as consensus (aka Byzantine agreement) [PSL80, LSP82] and a robust public transaction ledger (i.e., Bitcoin) can be built "on top" of such properties, assuming that the hashing power of an adversary controlling a fraction of the parties is strictly less than $1/2$.

Importantly, those properties hold assuming that all parties—honest and adversarial—"wake up" and start computing at the same time, or, alternatively, that they compute on a common random string only made available at the exact time when the protocol execution is to begin (see further discussion under related work below). Indeed, the coinbase parameter in Bitcoin's "genesis" block, hardcoded into the software, contains text from *The Times* 03/Jan/2009 issue [Bit], arguably unpredictable.

[1]In Bitcoin, solving a proof of work essentially amounts to brute-forcing a hash inequality based on SHA-256.

While satisfactory in some cases, such a trusted setup/behavioral assumption might be unrealistic in other PoW-based systems where details may have been released a lot earlier than the actual time when the system starts to run. A case in point is Ethereum, which was discussed for over a year before the system officially kicked off. That's from a practical point of view. At a foundational level, one would in addition like to understand what kind of cryptographic primitives can be realized without any trusted setup assumption and based on PoWs, and whether that is in particular the case for the Bitcoin backbone functionality and its enabling properties mentioned above.

The former question was recently considered by Andrychowicz and Dziembowski [AD15], who, building on previous suggestions by Aspnes *et al.* [AJK05] of using PoWs as an identity-assignment tool and constructions by Fitzi *et al.* [Fit03, CFF$^+$05] showing how to morph "graded" consistency into global consistency, showed how to create a consistent PKI using PoWs and no other trusted setup, which can then be used to run secure computation protocols (e.g., [Yao82, GMW87]) and realize any cryptographic functionality assuming an honest majority among parties. While this in principle addresses the foundational concerns, it leaves open the questions of doing it in *scalable* way—i.e., with round complexity independent of the number of parties, and in the context of blockchain protocols in particular, designing one that is *provably* secure without a trusted setup.

Our contributions. In this chapter we answer the above questions in the affirmative. First, we present a Bitcoin-like protocol that neither assumes a simultaneous start nor the existence of an unpredictable genesis block, and has round complexity essentially independent of the number of participants [2]. Effectively, the protocol, starting "from scratch," enables the coexistence of multiple genesis blocks with blockchains stemming from them, eventually enabling the players to converge to a single blockchain. This takes place despite the adversary being allowed (polynomial in the security parameter) pre-computation time. We work in the same model as [GKL15] and we assume a $1/2$ bound on adversarial hashing power. We call this protocol the *bootstrapped* (Bitcoin) backbone protocol. A pictorial overview of the protocol's phases, preceded by a period of potential precomputation by the corrupt players, is given in Figure 3.1.

Second, we present applications of our bootstrapped construction, starting with its original one: a distributed ledger, i.e., a public and permanent summary of all transac-

[2]"Essentially" because even though there will be a dependency of the round complexity of the setup phase on the probability of computing PoWs, which in turn depends on the number of parties, this dependency can be made small enough so as to be considered a constant. See Remark 2.

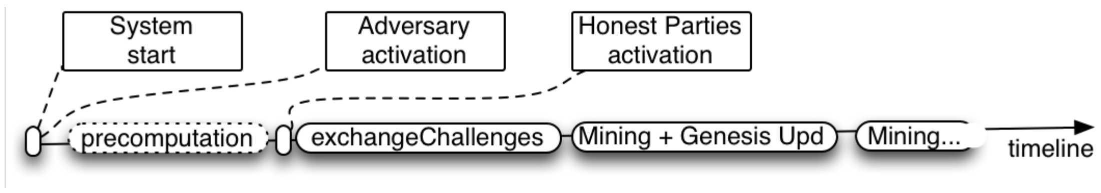


Figure 3.1: *Overview of the bootstrapped Bitcoin backbone protocol.*

tions that honest parties can agree on as well as add their own, despite the potentially disruptive behavior of parties harnessing less than $1/2$ of the hashing power. This entails proving that the ledger's required security properties (Persistence and Liveness — cf. [GKL15]) hold in a genesis block-less setting.

Next, we consider the problem of setting up a PKI in our unauthenticated network setting *from scratch*. As mentioned above, the idea of using PoWs as an identity-assignment tool was put forth by Aspnes *et al.* [AJK05]. Here we build on this idea as well as on the "2-for-1 PoWs" technique from [GKL15] to use our bootstrapped protocol to assign identities to parties. The assignment relation will possibly assign more than one identity to the same party, while guaranteeing that the majority of them is assigned to honest parties. Such an identity infrastructure/"pseudonymous PKI" has numerous applications, including the bootstrapping of a proof-of-stake protocol [KRDO17, KN12], and the election of honest-majority "subcommittees," which would enable the application of traditional Byzantine fault-tolerant techniques for ledger creation and maintenance (cf. [ACC$^+$17]) to permissionless (as opposed to permissioned) networks.

Finally, applying the 2-for-1 PoWs technique we also solve the consensus (aka Byzantine agreement) problem [PSL80, LSP82] probabilistically and from scratch, even if the adversary has almost the same hashing power as the honest parties[3], and with round complexity independent of the number of parties. Indeed, all our protocols have round complexity linear in the security parameter, and enjoy simultaneous termination.

Related work. Nakamoto [Nak08] proposed Bitcoin, the first decentralized currency system based on PoWs while relaxing the anonymity property of a digital currency to mere pseudonymity. This work was followed by a multitude of other related proposals including Litecoin, Primecoin [Kin13], and Zerocash [BCG$^+$14], and further analysis improvements (e.g., [ES14, EGSvR15]), to mention a few.

[3]Thus marking a contrast with the $\frac{2}{3}$ lower bound for consensus on the number of honest parties in the traditional network setting with no setup [Bor96].

As mentioned above, we work in a model that generalizes the model put forth by Garay *et al.* [GKL15], who abstracted out and formalized the core of the Bitcoin protocol—the Bitcoin *backbone*. As presented in [GKL15], however, the protocol considers as valid any chain that extends the empty chain, which is not going to work in our model. Indeed, if the adversary is allowed polynomial-time pre-computation, he can prepare a very long, private chain; then, by revealing blocks of this chain at the rate that honest players compute new blocks, he can break security. As also mentioned above, to overcome this problem one can assume that at the time honest parties start the computation, they have access to a fresh common random string (a "genesis" block). Then, if we consider as valid only the chains that extend this block, all results proved in [GKL15] follow, since the probability that the adversary can use blocks mined before honest players "woke up" is negligible in the security parameter. In this chapter we show how to establish such genesis block directly, and in a number of rounds essentially independent of the number of participants.

To our knowledge, the idea of using PoWs to distributedly agree on something (specifically, a PKI) in an unauthenticated setting with no trusted setup was first put forth by Aspnes *et al.* [AJK05]. There, it was suggested to use them as an identity-assignment tool to combat *Sybil* attacks [Dou02b], by establishing that the number of identities assigned to honest and adversarial parties is proportional to their computational power, respectively. In more detail, for one of the algorithms of [AJK05], if the adversary controls less than half of the total computational power, it will also control less than half of the total number of the generated identities. It is then proposed to run this procedure as a pre-processing step of an authenticated broadcast algorithm. However, these protocols require that the PKI is consistent, a property not explored in [AJK05].

It is in [AD15], where Andrychowicz and Dziembowski address the more general goal of secure computation in this setting based on PoWs, as mentioned earlier; the PoWs are used to build a "graded" PKI, where keys have "ranks." The graded PKI is an instance of a "graded agreement," or "partial consistency" problem [Fit03, CFF$^+$05, GKKO07], where honest parties do not disagree "by much," according to some metric. In [Fit03], Fitzi calls this the *b-set-neighboring* problem ("proxcast" in [CFF$^+$05]), with b the number of possible "grades," and shows how to achieve global consistency by running the b-set-neighboring protocol multiple times. In [AD15], the fact that unreliable broadcast is available among honest parties is used to achieve the same—global consistency on a PKI, where the number of identities each party gets is proportional to its hashing power, as suggested in [AJK05].

The protocol in [AD15], however, suffers from a total running time that depends on the number of parties, because of two factors: (1) the way in which it uses PoWs, and (2) the use of the Dolev-Strong authenticated broadcast protocol (run multiple times in parallel based on the graded PKI), which takes a linear number of rounds. Regarding (1), and in more detail, in order to assign exactly one key per party, a low variance PoW scheme is used. This implies that the time needed by an honest party to mine a PoW is going to be proportional to the ratio of the adversarial hashing power to the hashing power of the weakest honest party. Otherwise, the "rushing" adversary would be able to compute more identities in the additional time she has due to the latency of the communication infrastructure.[4] Regarding (2), we note that potentially an expected-constant-round protocol could be used instead of Dolev-Strong, although the parallel composition of n instances would require more involved techniques [CCGZ16].

Furthermore, having a PKI allows parties to generate an unpredictable beacon (in the random oracle model), which is then suggested in [AD15] as a genesis block-generation method for a new cryptocurrency. Yet, no formal treatment of the security of the resulting blockchain protocol is presented, and—as already mentioned—the round complexity of the suggested genesis block generation procedure is linear in the number of participants, both in contrast to our work.

As in [AD15], Katz *et al.* [KMS14] also consider achieving pseudonymous broadcast and secure computation from PoWs ("cryptographic puzzles") and the existence of digital signatures without prior PKI setup, but under the assumption of a fresh CRS. Finally, Pass *et al.* [PSS17] consider a partially synchronous model of communication where parties are not guaranteed to receive messages at the end of each round but rather after a specified delay Δ (cf. [DLS88]), and show that the backbone protocol can be proven secure in this setting. In principle, our results about the bootstrapped backbone protocol can be extended to their setting as shown in [GKL15].

Limitations and directions for future research. A practical limitation of the protocols we present in this chapter, as well as of the protocol in [AD15], is the communication intensive challenge-exchange phase. We conclude Section 3.4 with an additional modification to the protocol that somewhat addresses this concern, which among other

[4] On the flip side, the benefit of the approach in [AD15] is that when all honest parties have the *same* hashing power, a PKI that maps each party to exactly one identity and preserves an honest majority on the keys can be achieved. However, in today's environments where even small devices (e.g., mobile phones, smart watches) have powerful CPUs with different clock frequencies, this assumption is arguably weak.

makes parties generate a PoW for each of the challenges they send. Other interesting directions are further minimizing the initialization cost, and designing a challenge-exchange phase whose security does not rely in the ROM.

Organization of the chapter. The rest of the chapter is organized as follows. In Section 3.2 we provide the details of the model. In Section 3.3 we present the bootstrapped Bitcoin backbone protocol and its analysis. Applications are presented in Section 3.4: a robust public transaction ledger, PKI generation and consensus without trusted setup and with round complexity independent of the number of parties.

3.2 Preliminaries

We assume that we are in the synchronous q-bounded ROM setting and without a fresh CRS, as defined in Section 2.2. We assume everywhere a static adversary.

3.3 The Bootstrapped Backbone Protocol

We begin this section by presenting the "bootstrapped" Bitcoin backbone protocol, followed by its security analysis. In a nutshell, the protocol is a generalization of the protocol in [GKL15], which is enhanced in two ways: (1) an initial challenge-exchange phase, in which parties contribute random values, towards the establishment of an unpredictable genesis block, despite the precomputation efforts of corrupt parties, and (2) a ranking process and chain-validation predicate that, in addition to its basic function (checking the validity of a chain's content), enables the identification of "fresh" candidate genesis blocks. The ranking process yields a graded list of genesis blocks and is inpired by the "key ranking" protocol in [AD15], where it is used to produce a "graded" PKI, as mentioned in Section 3.1.

Before describing the bootstrapped backbone protocol in detail, we highlight its unique features.

– *No trusted setup*[5] *and individual genesis block mining.* Parties start without any prior coordination and enter an initial challenge-exchange phase, where they will exchange random values that will be used to construct "freshness" proofs for candidate genesis blocks. The parties will run the initial challenge-exchange phase

[5]While the term "no trusted setup" was used in [AD15] to describe our setting, we note that the random oracle can be used to agree on a non-fresh CRS. However, the term is not without any justification, as in practice in the place of the random oracle we would use an unkeyed hash function, e.g., SHA-256, and thus we wouldn't assume any setup.

for a small number of rounds, and subsequently will try to mine their own genesis blocks individually. Once they mine or accept a genesis block from the network they will engage in mining further blocks and exchanging blockchains as in Bitcoin's blockchain protocol. On occasion they might switch to a chain with a different genesis block. Nevertheless, as we will show, quite soon they will stabilize in a common prefix and a single genesis block.

- *Freshness of genesis block impacts chains' total weight.* Chains rooted at a genesis block will incorporate its weight in their total valuation. Genesis blocks can be quite "heavy" compared to regular blocks and their total valuation will depend on how fresh they are. Their weight in general might be as much as a linear number of regular blocks in the security parameter. Furthermore, each regular block in a chain accounts for 3 units in terms of the total weight of the chain, something that, as we show, will be crucial to account for differences in terms of weight that are assigned to the same genesis block by different parties running the protocol (cf. Remark 1).
- *Personalized chain selection rule.* Given the co-existence of multiple genesis blocks, a ranking process is incorporated into the chain selection rule that, in addition to its basic function (checking the validity of a chain's content) and picking the longest chain, it now also takes into account the freshness degree of a genesis block from the perspective of each party running the protocol. The ranking process effectively yields a graded list of genesis blocks and is inspired by the "key ranking" protocol in [AD15], where it is used to produce a "graded" PKI (see further discussion below). The weight value for each genesis block will be thus proportional to its *perceived* "freshness" by each party running the protocol (the fresher the block the higher its weight). It follows that honest parties use different chain selection procedures since each predicate is "keyed" with the random coins that were contributed by each party in the challenge-exchange phase (and thus guaranteed to be fresh from the party's perspective). This has the side effect that the same genesis block might be weighed differently by different parties. Despite these differences, we show that eventually all parties accept the same chains as valid and hence will unify their chain selection rule in the course of the protocol.
- *Robustness is achieved after an initial period of protocol stabilization.* All our modifications integrate seamlessly with the Bitcoin backbone protocol [GKL15], and we are able to show that our blockchain protocol is a robust transaction ledger, in the sense of satisfying the properties of persistence and liveness. Nevertheless,

contrary to [GKL15], the properties are satisfied only after an initial period of rounds where persistence is uncertain and liveness might be slower; this is the period where the parties still stabilize the genesis block and they might be more susceptible to attacks. Despite this, a ledger built on top of our blockchain will be available immediately after the challenges exchange phase. Furthermore, once the stabilization period is over the robust transaction ledger behavior is guaranteed with overwhelming probability (in the length of the security parameter).

3.3.1 Protocol Description

After an initial ("challenge") phase, each party is to maintain a data structure called a "blockchain". Each party's chain may be different, but, as we will prove, under certain well-defined conditions, the chains of honest parties will share a large common prefix. We proceed to give some more details about the structure of the blockchains in our protocol, following [GKL15].

A *block* is any triple of the form $B = \langle s, x, ctr \rangle$ where $s \in \{0,1\}^\kappa, x \in \{0,1\}^*, ctr \in \mathbb{N}$ are such that satisfy predicate $\mathsf{validblock}^D_q(B)$ defined as

$$(H(ctr, G(s,x)) < D) \wedge (ctr \leq q),$$

where H, G are cryptographic hash functions (e.g., SHA-256) modelled as random oracles. The parameter $D \in \mathbb{N}$ is also called the block's *difficulty level*. The parameter $q \in \mathbb{N}$ is a bound that in the Bitcoin implementation determines the size of the register ctr; in our treatment we allow this to be arbitrary, and use it to denote the maximum allowed number of hash queries in a round. We do this for convenience and our analysis applies in a straightforward manner to the case that ctr is restricted to the range $0 \leq ctr < 2^{32}$ and q is independent of ctr.

A chain $\mathcal{C}$ with $\mathsf{head}(\mathcal{C}) = \langle s', x', ctr' \rangle$ can be extended to a longer chain by appending a valid block $B = \langle s, x, ctr \rangle$ that satisfies $s = H(ctr', G(s', x'))$. In case $\mathcal{C} = \varepsilon$, by convention any valid block of the form $\langle s, x, ctr \rangle$ may extend it. In either case we have an extended chain $\mathcal{C}_{\mathsf{new}} = \mathcal{C}B$ that satisfies $\mathsf{head}(\mathcal{C}_{\mathsf{new}}) = B$.

Following [GKL15], our protocol description intentionally avoids specifying the type of values that parties try to insert in the chain, the type of chain validation they perform (beyond checking for its structural properties with respect to the hash functions $G(\cdot), H(\cdot)$), and the way they interpret the chain. In the protocol description, these actions are abstracted by the external functions $\mathrm{V}(\cdot), \mathrm{I}(\cdot)$[6], $R(\cdot)$ which are specified by the application that runs "on top" of the backbone protocol.

The protocol is specified as Algorithm 1. At a high level, the protocol first executes a challenge-exchange phase for $l+1$ rounds (l will be determined later), followed by the basic backbone functions, i.e., mining and broadcasting blocks; a crucial difference here with respect to the original backbone protocol is that the chain validation process must also verify candidate genesis blocks, which in turn requires updating the validation function as the protocol proceeds. (This, however, only happens in the next l rounds after the challenge phase.) The protocol's supporting algorithms are specified next.

[6]We assume that no two invocations of $\mathrm{I}(\cdot)$ performed by honest parties return the same result, as formalized by the Input Entropy condition in [GKL15].

Algorithm 1 The *bootstrapped backbone* protocol, parameterized by the *input contribution function* $\mathrm{I}(\cdot)$, the *chain reading function* $\mathrm{R}(\cdot)$, and parameter l.

```
1: $\mathcal{C} \leftarrow \varepsilon$
2: $st \leftarrow \varepsilon$
3: $round \leftarrow 1$                                         ▷ Global variable $round$
4: $Gen \leftarrow \emptyset$                                   ▷ Set of candidate genesis blocks
5: $Rank \leftarrow \langle \varepsilon \rangle$                ▷ Weights of candidate genesis blocks
6: $(\mathbf{c}, \mathbf{A}, c) \leftarrow \mathsf{exchangeChallenges}(1^\kappa)$
7: while TRUE do
8:     $k \leftarrow round - l - 2$          ▷ At round $\ell + 2$ no new genesis blocks are accepted.
9:     $M_{Gen} \leftarrow \{(\langle s', x', ctr' \rangle, \langle A'_{l+1}, \ldots, A'_{l+2-k} \rangle)\}$ from RECEIVE()
10:    $M_{Chain} \leftarrow$ chains $\mathcal{C}'$ found in RECEIVE()
11:    $(Gen, Rank) \leftarrow \mathsf{updateValidate}(\mathbf{c}, \mathbf{A}, M_{Gen}, Gen, Rank)$
12:    $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, M_{Chain}, Gen, Rank)$
13:    $\langle st, x \rangle \leftarrow \mathrm{I}(st, \tilde{\mathcal{C}}, round, \text{INPUT}(), \text{RECEIVE}())$
14:    $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}}, c)$
15:    if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ then
16:        if $\mathcal{C} = \varepsilon$ then                  ▷ New genesis block has been produced
17:            DIFFUSE( $(\mathcal{C}_{\mathsf{new}}, \langle A_{l+1}, \ldots, A_{l+2-(k+1)} \rangle)$ )
18:        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
19:        DIFFUSE($\mathcal{C}$)
20:    $round \leftarrow round + 1$
21:    if INPUT() contains READ then
22:        write $\mathrm{R}(\mathbf{x}_{\mathcal{C}})$ to OUTPUT()
```

The challenge-exchange phase. In order to generate an unpredictable genesis block, parties first execute a "challenge-exchange" phase, where they broadcast, for a given number of rounds $(l+1)$, randomly generated challenges that depend on the challenges received in the previous rounds. The property that is assured is that an honest party's k-round challenge, $2 \leq k \leq l+1$, depends on the $(k-1)$-round challenges of all honest parties. This dependence is made explicit through the random oracle. The code of the challenge-exchange phase is shown in Algorithm 2.

Algorithm 2 The *challenge-exchange* function. Note that variable *round* is global, and originally set to 1.

```
1: function exchangeChallenges($1^\kappa$)
2:     $c_1 \overset{R}{\leftarrow} \{0,1\}^\kappa$
3:     DIFFUSE($c_1$)
4:     $round \leftarrow round + 1$
5:     while $round \leq l + 1$ do
6:         $A_{round} \leftarrow$ $\kappa$-bit messages found in RECEIVE()
7:         $r_{round} \overset{R}{\leftarrow} \{0,1\}^\kappa$
8:         $A_{round} \leftarrow A_{round} || r_{round}$
9:         $c_{round} \leftarrow H(A_{round})$                ▷ Compute challenge
10:        DIFFUSE($c_{round}$)
11:        $round \leftarrow round + 1$
12:    return $(\langle c_1, \ldots c_l \rangle, \langle A_2, \ldots A_{l+1} \rangle, c_{l+1})$
```

Validation predicate update. In the original backbone protocol [GKL15], the chain validation function (called validate—see below) performs a validation of the structural properties of a given chain $\mathcal{C}$, and remains unchanged throughout the protocol. In our case, however, where there is no initial fresh common random string, the function plays the additional role of checking for valid genesis blocks, and parties have to update their validation predicate as the protocol advances (for the first l rounds after the challenge phase).

Indeed, using the challenges distributed in the challenge-exchange phase of the protocol, parties are able to identify fresh candidate genesis blocks that have been shared during that phase and are accompanied by a valid proof. In addition, the valid genesis blocks are ranked with a negative dependence on the round they were received. In order to help other parties to also identify the same genesis blocks, parties diffuse the valid genesis blocks they have accepted together with the additional information needed by the other parties for verification. The validation predicate update function is shown in Algorithm 3. *Gen* is the set of candidate genesis blocks, while *Rank* is a map that stores their weights.

Chain validation. A chain is considered valid if in addition to the checks performed

Algorithm 3 The *validation predicate update* function.

```
1: function updateValidate($\mathbf{c}, \mathbf{A}, M_{Gen}, Gen, Rank$)
2:     $k \leftarrow round - l - 2$
3:     if $(k < 1) \vee (k \geq l)$ then
4:         return $(Gen, Rank)$    ▷ Updates happen during rounds $\{\ell+3,\ldots,2\ell+1\}$
5:     for each $(\langle s,x,ctr\rangle, \langle A'_{l+1},\ldots,A'_{l+2-k}\rangle)$ in $M_{Gen}$ do
6:         if validblock$^D_q(\langle s,x,ctr\rangle) \wedge \langle s,x,ctr\rangle \notin Gen$ then
7:             $flag \leftarrow (H(A'_{l+1}) = s) \wedge (c_{l+1-k} \in A'_{l+2-k})$
8:             for $i = l+2-k$ to $l$ do
9:                 if $H(A'_i) \notin A'_{i+1}$ then
10:                    $flag \leftarrow$ FALSE
11:            if $flag =$ TRUE then
12:                $Gen \leftarrow Gen \cup \langle s,x,ctr\rangle$
13:                $Rank[\langle s,x,ctr\rangle] \leftarrow l - k$
14:                DIFFUSE$(\langle s,x,ctr\rangle, \langle A'_{l+1},\ldots,A'_{l+2-k},A_{l+1-k}\rangle)$    ▷ Augment $A'$ sequence with own $A$ value.
15:    return $(Gen, Rank)$
```

by the basic backbone protocol regarding the chain's structural properties, its genesis block is in the *Gen* list, which is updated by the updateValidate function (Algorithm 3). The chain validation function is shown in Algorithm 4.

Chain selection. The objective of the next algorithm in Algorithm 1, called maxvalid, is to find the "best possible" chain. The accepted genesis blocks have different *weights* depending on when a party received them. It is possible that the same genesis block is received by honest parties in two different rounds (as we show later, those rounds have to be consecutive). In order to take into account the "slack" introduced by the different views honest parties may have regarding the same block, as well as the different weights different blocks may have, we let the *weight of a chain $\mathcal{C}$* be equal to the *weight of its genesis block plus three times its length minus one*. The chain selection function is shown in Algorithm 5.

Algorithm 4 The *chain validation predicate*, parameterized by q, D, the hash functions $G(\cdot), H(\cdot)$, and the *content validation predicate* $V(\cdot)$. The input is $\mathcal{C}$.

```
1: function validate($\mathcal{C}$, $Gen$)
2:     $b \leftarrow V(\mathbf{x}_{\mathcal{C}}) \wedge (C \neq \varepsilon) \wedge (\text{tail}(\mathcal{C}) \in Gen)$
3:     if $b = \text{True}$ then
4:         $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
5:         $s' \leftarrow H(ctr, G(s, x))$
6:         repeat
7:             $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
8:             if $\mathsf{validblock}_q^D(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$ then
9:                 $(s', \mathcal{C}) \leftarrow (s, \mathcal{C}^{\lceil 1})$ ▷ Retain hash value and remove the head from $\mathcal{C}$
10:            else
11:                $b \leftarrow \text{False}$
12:        until $(\mathcal{C} = \varepsilon) \vee (b = \text{False})$
13:    return $b$
```

Algorithm 5 The function that finds the “best” chain. The input is a set of chains and the list of genesis blocks.

```
1: function maxvalid($\mathcal{C}$, $M_{Chain}$, $Gen$, $Rank$)
2:     $temp \leftarrow \varepsilon$
3:     $maxweight \leftarrow 0$
4:     for each $\mathcal{C}'$ in $\mathcal{C} \cup M_{Chain}$ do
5:         if validate($\mathcal{C}', Gen$) then
6:             $weight \leftarrow Rank(\text{tail}(\mathcal{C}')) + 3(|\mathcal{C}'| - 1)$
7:             if $maxweight < weight$ then
8:                 $maxweight \leftarrow weight$
9:                 $temp \leftarrow \mathcal{C}'$
10:    return $temp$
```

The proof-of-work function. Finally, we need to modify the proof-of-work function in [GKL15], so that when a genesis block is mined, the challenge computed in the last round of the challenge-exchange phase will be included in the block. This, is required so that other honest parties accept this block as valid and rank it accordingly. The code is presented in Algorithm 6.

Algorithm 6 The *proof of work* function, parameterized by q, D and hash functions $H(\cdot), G(\cdot)$. The input is $(x, \mathcal{C}, c)$.

```
1: function pow($x, \mathcal{C}, c$)
2:     if $\mathcal{C} = \varepsilon$ then
3:         $s \leftarrow c$                                    ▷ c is required to prove freshness
4:     else
5:         $\langle s', x', ctr' \rangle \leftarrow$ head($\mathcal{C}$)
6:         $s \leftarrow H(ctr', G(s', x'))$
7:     $ctr \leftarrow 1$
8:     $B \leftarrow \varepsilon$
9:     $h \leftarrow G(s, x)$
10:    while $(ctr \leq q)$ do
11:        if $(H(ctr, h) < D)$ then                            ▷ Proof of work found
12:            $B \leftarrow \langle s, x, ctr \rangle$
13:            break
14:        $ctr \leftarrow ctr + 1$
15:    return $\mathcal{C}B$                                    ▷ Extend chain
```

Figure 3.2 presents the overall structure (phases and corresponding rounds) of the bootstrapped backbone protocol. Next, we turn to its analysis.

Remark 1. To understand some of our design choices we briefly give some examples of simpler protocols that don't work. For the first example, assume that we only have one round of challenge exchange i.e. l equal to 1. With some non-negligible probability, the adversary can send one block to half of the honest parties and another block to the other half. By splitting the honest parties in two groups such that no one in the first group will choose the chain of the second and vice versa, agreement becomes impossible. Moreover, l must be large enough so that at least one honest party computes a genesis block with overwhelming probability. Otherwise the adversary can choose to remain silent and no genesis block will be mined with non-negligible probability.

For the second example assume that blocks weigh less than 3 units, as in the original protocol. Also, assume that somehow the problem of the first example was avoided and honest parties only adopted chains with genesis blocks that everyone had in their

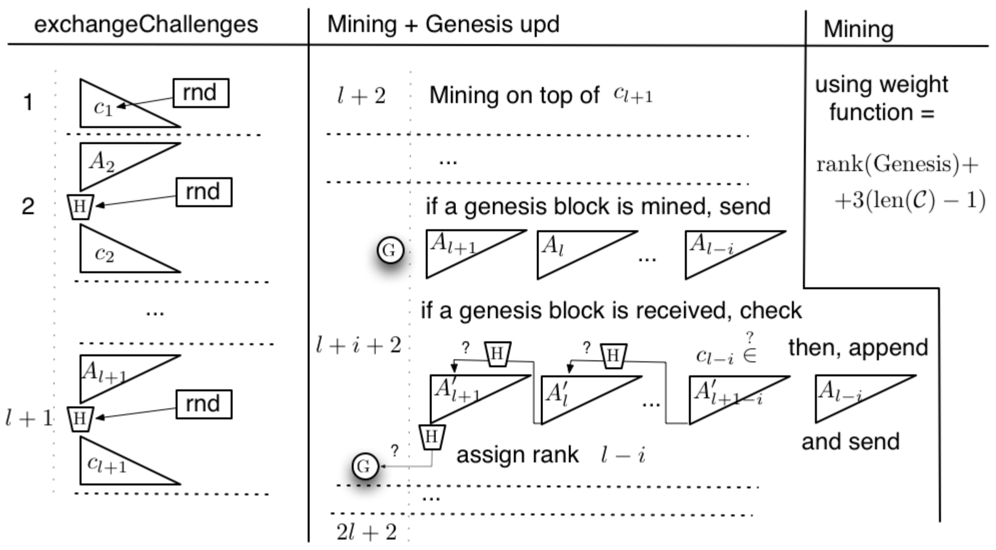
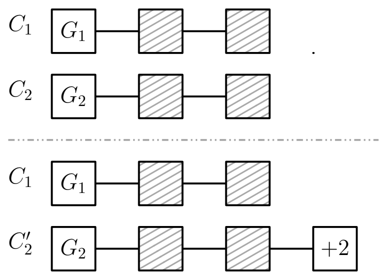


Figure 3.2: *The different phases of the bootstrapped Bitcoin backbone protocol.*

genesis block list. In this case, uniquely successful rounds would not imply agreement on a single chain (see Figure 3.3) (an important fact of previous analyses of blockchain protocols [GKL15, PSS17]), as the adversary would have been able to take advantage of the different views that honest parties have regarding the weight of genesis blocks. However, if we set the block weight to 3, this event becomes impossible and makes the analysis a lot easier.



	P_1	P_2
G_1	6	5
G_2	4	5
C_1	10	9
C_2	8	9
C'_2	10	11

Figure 3.3: *An example where blocks weigh 2 units. The weights of the respective chains are depicted in the table. Initially party P_1 has adopted chain $\mathcal{C}_1$ and party P_2 chain $\mathcal{C}_2$. Then, a uniquely successful round occurs and $\mathcal{C}_2$ is extended to $\mathcal{C}'_2$. Notice that, P_1 will not adopt $\mathcal{C}'_2$ since it has the same weight as C_1. If the new block weighted 3 units instead, all parties would have adopted chain $\mathcal{C}'_2$.*

3.3.2 Analysis of the Bootstrapped Backbone Protocol

First, we provide some additional definitions that will become handy in the analysis. We saw in the previous section that genesis blocks are assigned weights, and, further, that a single genesis block may have different weights for different parties depending on when they received it. We extend this notion to chains of blocks.

Definition 7. Let C be a valid chain and P be some party. If $\mathcal{C} \neq \varepsilon$, and its genesis block B was received until round $2l+1$ by party P, who assigned it weight $w_P(B)$, then we define the weight of $\mathcal{C}$ with respect to party P to be:

$$w_P(\mathcal{C}) = w_P(B) + 3(|\mathcal{C}| - 1).$$

Otherwise, $w_P(\mathcal{C}) = -1$.

In [GKL15], all parties assign the same weight to the same chain, i.e., the length of the chain; thus, for all parties P_i, P_j we have that $w_{P_i}(\mathcal{C}) = w_{P_j}(\mathcal{C})$. In contrast, in our case the genesis block of each chain may have different weight for different parties, akin to some bounded amount of "noise" that is party-dependent being added to the chain weights. We are going to show that if the amount of noise is at most 1, then by letting each new block weigh 3 units our protocol satisfies the chain growth, common prefix and chain quality properties.

Definition 8. Regarding chains and their weight:

- Define $h_{\mathcal{C}} = \max_{P \in \mathcal{P}}\{w_P(\mathcal{C})\}$ and $\ell_{\mathcal{C}} = \min_{P \in \mathcal{P}}\{w_P(\mathcal{C})\}$, where $\mathcal{P}$ is the set of honest parties.
- Let $\mathcal{C}(B)$ denote the truncation of chain $\mathcal{C}$ after its block B.
- For a block B of a chain $\mathcal{C}$, define $h_{\mathcal{C}}(B) = h_{\mathcal{C}(B)}$ and similarly for $\ell_{\mathcal{C}}(B)$. (Sometimes we will abuse notation and write $\ell(B)$ instead of $\ell_{\mathcal{C}}(B)$.)
- For chains $\mathcal{C}_1$ and $\mathcal{C}_2$, define $\mathcal{C}_1 \cap \mathcal{C}_2$ to be the chain formed by their common prefix.

The following are important concepts introduced in [GKL15], which we are also going to use in our analysis:

Definition 9. Regarding blocks and rounds:

- A block is called *honest*, if an honest party was the first to diffuse this block,
- A round is called *successful* if at least one honest block was diffused for the first time in this round;

- A round is called *uniquely successful* if *exactly* one honest block was diffused for the first time in this round;
- A block is called *adversarial*, if is not an honest block, and *u.s. block*, if it is an honest block, and was first diffused in a uniquely successful round.

Recall that our model is "flat" in terms of computational power in the sense that all honest parties are assumed to have the same computational power while the adversary has computational power proportional to the number of players that it controls. The total number of parties is n and the adversary is assumed to control up to t of them (honest parties do not know any of these parameters). Obtaining a new block is achieved by finding a hash value that is smaller than the difficulty parameter D. Thus, the success probability that a single hash query produces a solution is $p = \frac{D}{2^\kappa}$, where κ is the length of the hash.

For each round j, we define the Boolean random variables X_j and Y_j as follows. Let $X_j = 1$ iff j was a *successful round*, and let $Y_j = 1$ iff j was a *uniquely successful round*. With respect to a set of rounds S, let $Z(S)$ denote the number of *new* PoWs obtained by the adversary during the rounds in S (i.e., in $qt|S|$ queries). Also, let $X(S) = \sum_{j\in S} X_j$ and define $Y(S)$ similarly.

We define α, β, γ, f, following the analysis of [GKL15], as follows:

- $pq(n-t) \geq \mathbb{E}[X_i] = 1-(1-p)^{q(n-t)} = \alpha$
- $\mathbb{E}[Y_i] \geq (n-t)pq(1-p)^{q(n-t)-1} = \gamma$
- $pqt = \beta \geq \mathbb{E}[Z(\{i\})]$
- $f = pq(n-t) + \beta = pqn \geq \alpha + \beta$

where i is some round of the execution. Note that $\gamma|S| \leq \mathbb{E}[Y(S)] \leq \mathbb{E}[X(S)]$ and $\mathbb{E}[Z(S)] \leq \beta|S|$.

For the rest of our analysis, we are going to make certain assumptions about the power of the adversary compared to that of the honest parties, as well as the protocol parameters.

Assumption 1. For some $\delta \in (0,1)$, we assume that:

- $t < (1-\delta)(n-t)$; *(Honest majority)*
- $3(1+\delta)f \leq \delta^2$; *(Bounded block generation rate)*
- $l > \frac{(1-\delta)\kappa/\gamma+3}{1-3(1+\delta)f}$ *(Challenge-exchange phase length)*

Remark 2. The dependency of γ on n does not undermine the scalability of the round complexity of our protocol which depends on l. This claim is argued on the basis that the difficulty level D can be set proportional to $1/(n-t)$, so that γ can be treated as a constant and then l is in essence independent of n (note that both parameters would be polynomials in κ). For example, if $D = \frac{2^\kappa}{18(n-t)q}$, then $f < 1/9$ and $\frac{1}{\gamma} \leq 11$.

Next, we show that our assumption implies that the expected number of uniquely successful rounds is more than that of the adversarial blocks.

Lemma 10. *Assume Assumption 1. It holds that* $\gamma \geq (1+\delta)\beta$.

Proof. We have that:

$$\gamma = (n-t)pq(1-p)^{q(n-t)-1} > (n-t)pq(1-(n-t)pq) \geq \frac{\beta}{1-\delta}(1-\frac{\delta^2}{3}) \geq (1+\delta)\beta$$

where the first inequality follows from the Bernoulli inequality, while the second inequality follows from the honest majority and bounded block generation rate assumptions. □

Due to the previous lemma, and w.l.o.g, for the rest of the analysis we assume that $\beta = \frac{\gamma}{1+\delta}$.

Next, we define a set of "bad" events related to the random oracle, which we later show to occur only with negligible probability.

Definition 11. We say that a bad event happened for the RO in an execution iff either of the following happens:

- A collision is found;
- the RO is queried with string m, that contains a substring m' that is sampled later from a distribution with min entropy at least $\omega(\log(\kappa))$;

The above "bad" events not occurring, imply that the blockchains produced in an execution satisfy certain structural properties, e.g., that no cycle exist. These properties were captured at a higher level in [GKL15] under the following definition, which we include for completeness. In any case, we note that if the events of Definition 11 do not happen in some execution, it is implied that the events of Definition 12 also do not occur.

Definition 12. 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. A *prediction* occurs when a block extends one honest block which was computed at a later round.

To simplify the probabilistic part of our analysis, we define a set of well-behaved executions, which we call typical, and show that they occur with overwhelming probability.

Definition 13. An execution is typical, if, $\kappa > \frac{81}{\delta(1-\delta)}$ and for any set S of at least $(1-\delta)\delta^2\kappa$ consecutive rounds, it holds that:

- If $\min(S) \geq l+2$, then $(1-\delta/8)\alpha|S| < X(S)$ and $\max((1-\delta/8)\gamma|S|, 1) < Y(S)$;
- $Z(S) < (1+\delta/9)\beta|S|$, and $Z(S)+X(S) \leq (1+\delta)f|S|$;
- no bad events happen for the RO (Definition 11).

Theorem 14. *Assume Assumption 1. An execution is typical with probability* $1 - \mathit{negl}(\kappa)$.

Proof. We will show that the negation of each of the events described, happens only with negligible probability in κ. Then, using the union bound we will argue that none of them happen, again with negligible probability.

First, take some sequence of rounds S, which satisfies the conditions of Definition 13. For the first item, note that $\Pr[Y(S)=0] \leq (1-\gamma)^{(1-\delta)\delta^2\kappa} < e^{-(1-\delta)\delta^2\gamma\kappa}$. As argued in Remark 2, difficulty D is chosen so that γ is lower bounded by a constant in $(0,1)$. Hence, $\Pr[Y(S)=0] = \mathsf{negl}(\kappa)$. For the second and third items we are going to use the Chernoff bound. We have that:

$$\Pr[Y(S) \leq (1-\frac{\delta}{8})\mathbb{E}[Y(S)]] \leq e^{-\Omega(\delta^2\gamma\kappa)}\ ,\ \Pr[X(S) \leq (1-\frac{\delta}{8})\mathbb{E}[X(S)]] \leq e^{-\Omega(\delta^2\alpha\kappa)}$$
$$\text{and } \Pr[Z(S) \geq (1+\frac{\delta}{9})\beta|S|] \leq e^{-\Omega(\delta^2\gamma\kappa)}.$$

All these events happen with negligible probability.

Next, we argue about the bad events for the RO. First, the probability that a collision happens in a polynomially bounded execution, is easily bounded by $\binom{L}{2}2^{-\kappa} = \mathit{negl}(\kappa)$, where L is the length of the execution, and κ is the length of the hash. Regarding the second event, it is equivalent to predicting the result of a high min-entropy source, with only a polynomial number of guesses. This event happens only with negligible probability.

To finish the proof, note that there is only a polynomial number of round intervals S, and thus only a polynomial number of events, all happening with negligible probability in κ. By an application of the union bound, the theorem follows.

□

κ :	security parameter
D :	block difficulty parameter
n :	number of parties
t :	number of corrupted parties
q :	number of queries to the RO per round per party
β :	expected number of blocks computed by the adversary per round
α :	lower bound on the rate of successful rounds
γ :	lower bound on the rate of uniquely successful rounds
f :	upper bound on the rate of computed blocks
δ :	honest parties' advantage from Assumption 1
l :	length of challenge-exchange phase

Table 3.1: *The parameters in our analysis. Parameters $\kappa, n, t, D, q, l \in \mathbb{N}$, while $\alpha, \beta, \gamma, f, \delta \in \mathbb{R}$.*

Finally, we show that in a typical execution, the number of uniquely successful rounds exceeds the number of blocks the adversary mines, in any large enough interval. This result will be crucial in proving that honest parties converge on a single genesis block. For a summary of our notation we refer to Table 3.1.

Lemma 15. *Assume a typical execution and Assumption 1. For any set of consecutive rounds $S = \{i, \dots, j\}$, where $|S| \geq (1-\delta)\kappa$, it holds that $Y(S \setminus \{i\}) > (1+\frac{5\delta}{9})Z(S)$.*

Proof. We have that:

$$\begin{aligned} Y(S \setminus \{i\}) &> (1-\frac{\delta}{8})\gamma(|S|-1) \geq (1-\frac{\delta}{8})(1+\delta)\beta(|S|-1) \\ &\geq (1+\frac{5\delta}{9})(1+\frac{\delta}{9})\beta|S| > (1+\frac{5\delta}{9})Z(S) \end{aligned}$$

where the third inequality follows from the fact that $(1-\frac{\delta}{8})(1+\delta) \geq (1+\frac{5\delta}{9})(1+\frac{\delta}{9})(1+\frac{\delta}{80})$. □

Properties of the genesis block generation process. We now establish a number of properties of the genesis block generation process.

Lemma 16 (Graded Consistency)**.** *If any honest party P_i accepts genesis block B with $w_{P_i}(B) > 1$, then for any honest party P_j, $w_{P_j}(B) \geq w_{P_i}(B) - 1$.*

Proof. Let $w_{P_i}(B) = k > 1$. Since P_i accepted B with rank k at some round r, he must have received a message of the form $(B, E_{l+1}, .., E_{k+1})$, where

- B is a valid block that contains $H(E_{l+1})$;
- E_{k+1} contains c_k and for $k+2 < j \leq l+1$, E_j contains $H(E_{j-1})$; and
- c_k is the challenge computed by P_i at round k.

Since $k > 0$, according to Algorithm 3, P_i at the same round is going to diffuse $(B, E_{l+1}, .., E_{k+1}, A_k)$, where $H(A_k) = c_k$ is contained in E_{k+1} and A_k contains all the messages received by P_i at round k. All honest-party challenges of round $k-1$ were received in this round; therefore, all honest parties have accepted or will accept block B by the next round and the lemma follows. □

Lemma 17 (Validity)**.** *Genesis blocks computed by honest parties before round* $2l+1$*, will be accepted by all honest parties in the next round.*

Proof. Suppose honest party P_i mined genesis block B at round m. According to Algorithm 1, B contains the challenge he has computed at the last round of the challenge-exchange phase. In addition, when the party diffuses it, it includes the message sets $A_{l+1}, \ldots, A_r$, where A_j contains the messages received by P_i at round j and $r = 2l+2-m$. Since P_i is honest, the following hold:

- B is a valid block that contains $H(A_{l+1})$;
- for $r+1 < j \leq l+1$, A_j contains $H(A_{j-1})$;
- if c_r is the challenge sent by some honest party at round r, then c_r is contained in A_{r+1}; and
- all honest parties are going to receive the message.

Thus, all honest parties are going to accept B at round $m+1$ and the lemma follows. □

Lemma 18 (Freshness)**.** *Assume a typical execution and let* $r \leq l+2$*. Then, every block computed before round* r *cannot be part of some chain with genesis block* B*, where* $w_P(B) \geq \min(l, r-1)$ *for some honest party* P*.*

Proof. If $l \leq r-1$, note that $l-1$ is the greatest rank that can be assigned by Algorithm 3, hence it always holds that $w_P(B) < l$. Next, we analyze the other case, where $r-1 < l$.

We first look into the case where the block in the statement is a genesis block. For the sake of contradiction, suppose the adversary mines some genesis block B *before* round r, and this block is accepted by some honest party P with a value greater or equal to $r-1$. In the worst case, that means that the adversary also created sets $A_{l+1}, \ldots, A_{r+1}$ such that:

- B is a valid block that contains $H(A_{l+1})$;
- for $r+1 < j \leq l+1$, A_j contains $H(A_{j-1})$; and
- if c_r is the r round challenge of P, c_r is part of A_{r+1}.

Due to the random nonce honest parties add to their challenge at every round, the probability that the adversary can guess c_r before round r is negligible in κ. Hence, to put it more simply, the adversary must compute sets $A_{l+1},\ldots,A_{r+1}$ such that $H(A_{l+1}||H(\ldots H(A_{r+1})\ldots))$ is equal to c, where c is the value in some of the blocks he has precomputed before seeing c_r, and $c_r \in A_{r+1}$.

By the typicality of the execution we can show the following: First, since c_r is sampled from some high min-entropy distribution, the adversary will compute $H(A_{r+1})$, for the first time, after seeing c_r. This in turn, implies that it will compute $H(A_{r+2}||H(A_{r+1}))$ after computing $H(A_{r+1})$ for the first time, since $H(A_{r+1})$ is sampled from a uniform distribution. Inductively, $H(A_{l+1}||H(\ldots H(A_{r+1})\ldots))$ must have been queried after c_r was sampled. This is a contradiction, since the RO was queried with a string containing c, which is equal to $H(A_{l+1}||H(\ldots H(A_{r+1})\ldots))$, before the later query happen for the first time. Therefore, the adversary cannot create a genesis block that invalidates the statement of our lemma.

Otherwise, suppose that there exists some non-genesis adversarial block B', that has been mined before round r and is part of a chain with genesis block B, where for some honest party P, $w_p(B) \geq r-1$. Again due to collision resistance, we can show that with overwhelming probability B must have been mined before B', and thus as we proved for the first case, the probability that a genesis block with these properties exists is negligible in κ. Hence, the lemma follows. $\square$

Weak chain growth. We now turn our attention to the weight of chains and prove a *weak* chain-growth property. In the original Bitcoin backbone protocol [GKL15], it was proved that chains grow at least at the rate of successful rounds, independently of the adversary's behavior. Here, at least initially, the chains of honest parties grow in a "weak" manner, in the sense that the adversary is able to slow down this growth by using his own blocks. Later on, we will show that after some specific round our protocol also achieves optimal chain growth.

Lemma 19. *Assume a typical execution and let rounds r,s be such that $l+2 \leq r \leq s < 2l+2$, and let $S = \{r,\ldots,s-1\}, S' = \{2,\ldots,s-1\}$. Suppose that at round r an honest party, say, P_1 has*[7] *a chain $\mathcal{C}$ such that $w_{P_1}(\mathcal{C}) = d$. Then, by round s, every honest*

[7]We use the terminology "P has a chain at round r", to denote the fact that P at the end of $r-1$ has

party P will have received a chain $\mathcal{C}'$ of weight at least $w_P(\mathcal{C}') = d - 2 + 3 \cdot Y(S) - Z(S')$.

Proof. We start by dealing with the case where $s < 2l+1$. Let $\ell(r') = d$ iff d is the minimum value of the set $\{\ell_{\mathcal{C}} | P$ is honest and at round r' adopts chain $\mathcal{C}\}$. Observe that if at round r, P_1 has a chain $\mathcal{C}$ of weight $w_{P_1}(\mathcal{C}) = d$, then he broadcast $\mathcal{C}$ at an earlier round. Due to Lemma 16, and by the fact that $r \leq s < 2l+1$, it follows that every honest party P will receive $\mathcal{C}$ by round r, and $w_P(\mathcal{C}) \geq d-1$. It is easy to see that if each honest party P at some round $r' < 2l+1$ has received a chain $\mathcal{C}$ where $w_P(\mathcal{C}) \geq k$, then for every round $s' : r' \leq s' < 2l+1$, it holds that $\ell(s') \geq k-1$. Thus for every round $s' : r \leq s' < 2l+1$ it holds that $\ell(s') \geq d-2$.

Next, we prove that uniquely successful rounds increase the value of ℓ.

Claim 1. Suppose round r is uniquely successful and $\ell(r) = d$. Then for any round $s : r < s < 2l+1$ it holds that $\ell(s) \geq d+2$. Moreover, if the adversary has not diffused by round s any chain $\mathcal{C}$ that contains an adversarial block B such that $\ell_{\mathcal{C}}(B) = d+2$, it holds that $\ell(s) \geq d+3$.

Proof of Claim. The proof is quite straightforward. For the first part, since $\ell(r) = d$ and r is uniquely successful, an honest party will broadcast a chain $\mathcal{C}$ at round r where $\ell_{\mathcal{C}} \geq d+3$. Thus, at round $r+1$ all parties will receive a chain that has weight at least $d+3$ according to their view. This implies that, at worst, they may adopt a chain of the same weight, hence in any case it holds that $\ell(s) \geq d+2$.

Next, for the sake of contradiction, suppose that there exists a round $s > r$, such that the adversary has not broadcast any block $B' = \text{head}(\mathcal{C}')$ such that $\ell_{\mathcal{C}'}(B') = d+2$ and $\mathcal{C}'$ is valid, and $\ell(s) < d+3$. Since at round $r+1$ honest parties receive $\mathcal{C}$, they will all adopt a chain that weighs in their view at least $d+3$. Otherwise, they would adopt $\mathcal{C}$. Moreover, they will never adopt a chain with smaller weight. Hence, the only way $\ell(r+1) = d+2$ is if a chain that has weight $d+2$ for some honest party was broadcast at some round. By our assumption, an honest party has mined the head of this chain. Since $\ell(r) = d$, he must have done that before round r, otherwise the chain would weigh at least $d+3$ for any honest party. However, if he mined this chain before round r, at round r all honest parties would have received this chain and $\ell(r) = d+2$, which is a contradiction. Hence, the claim follows. ⊣

Now, note that from Lemma 18, the adversary can only use blocks he computed after round 1 of the challenge exchange phase to slow down chain growth. We have

adopted this chain. Otherwise, we use the term "P has adopted a chain at round r".

two cases. In the first case, $Y(S) \leq Z(S')$. The claim above guarantees that every time a uniquely successful round r' happens, $\ell(r'+1) \geq \ell(r')+2$. Thus, by repeatedly applying this argument we immediately get that:

$$\begin{aligned}\ell(s) &\geq d-2+2Y(S) \geq d-2+3Y(S)-Y(S)\\ &\geq d-2+3Y(S)-Z(S'),\end{aligned}$$

which implies that at round s all honest parties have received a chain that has sufficient weight according to the lemma.

Otherwise, $Y(S) > Z(S')$. Note that for every uniquely successful round, in order for the condition of the claim above to hold, the adversary must broadcast different blocks that have weight at least $\ell(r)+2 = d$. Thus, for at least $Y(S)-Z(S')$ uniquely successful rounds the condition of the claim will not hold and for any such round r', $\ell(r'+1) \geq \ell(r')+3$. Thus,

$$\begin{aligned}\ell(s) &\geq d-2+3(Y(S)-Z(S'))+2Z(S')\\ &\geq d-2+3Y(S)-Z(S').\end{aligned}$$

We can extend this result for the case where $s = 2l+1$. If $r = s$, then we have already argued that all parties will have chains that weigh at least $d-2$ at round r. Otherwise, if $r < s$, by our previous arguments we know that it holds that $\ell(s-1) \geq d-2+3Y(S \setminus \{s-1\}) - Z(S' \setminus \{s-1\})$. If $s-1$ is uniquely successful, it follows that all parties will adopt chains that weigh at least $d-2+3Y(S)-Z(S') \geq d-2+3Y(S)-Z(S)$. Otherwise, the lemma is trivially satisfied. $\square$

Universal chain validity. A novelty of our construction is that the same genesis block may have different weight for different parties. Unfortunately, it could be the case that due to the adversary's influence, a genesis block is valid for one party but invalid for another. This could lead to disagreement, in the sense that some honest parties may adopt a chain that others don't because it is not valid for them. We will show that with overwhelming probability such an event cannot occur for our protocol; as such, chain validity is a "universal" property; if some honest party accepts a chain $\mathcal{C}$ as valid, then $\mathcal{C}$ will also be valid for all other parties.

Notice, that in order to prove the following lemma we need l to be greater than a value that depends on $1/\gamma$, i.e. the expected time it takes for honest parties to mine a block, and the security parameter κ (see also Remark 1). Intuitively l should be large enough so that (i) honest parties mine at least one block at this time interval, and (ii) any adversarial chain that is based on a genesis block diffused at the end of the bootstraping phase will never be adopted by honest parties (because such genesis block will have too small weight in comparison).

Lemma 20. *Assume a typical execution and Assumption 1, and that at round r an honest party P has chain $\mathcal{C}$. Then, $\mathcal{C}$ will also be valid for all other parties from this round on.*

Proof. For the sake of contradiction, suppose there exists some honest party P' such that P' has received chain $\mathcal{C}$ at round r and it is not valid for him. The only reason this may happen is that P' has not accepted $\mathcal{C}$'s genesis block B. By Lemma 17 all honest parties know the genesis blocks mined by other honest parties, thus B must have been computed by the adversary. By Lemma 16, B was accepted by honest parties only at round $2l+1$; no honest party accepts new genesis blocks after this round. We will show that with overwhelming probability in κ, no honest party will ever accept a chain based on B.

We start by showing this fact for the round $r' = 2l+1$. Without loss of generality, suppose that P is the first honest party that adopts a chain based on B at round r'. By the typicality of the execution, it holds that honest parties mine at least one genesis block before round $l+2+(1-\delta)\kappa/\gamma$, i.e., in $(1-\delta)\kappa/\gamma$ rounds, with weight at least $l-(1-\delta)\kappa/\gamma$. Hence, it follows from Lemma 19 that every honest party at round r' will have a chain of weight at least $l-(1-\delta)\kappa/\gamma-2+3(Y(S')-1)-Z_1(S)$, where $S' = \{l+2,\ldots,r'-1\}, S = \{2,\ldots,r'-1\}$ and $Z_1(S)$ is the set of blocks the adversary has broadcast to slow down chain growth during rounds in S.

On the other hand, since block B is adversarial, and chain $\mathcal{C}$ is accepted for the first time by an honest party at round r', all of its blocks must be adversarial; possibly $\mathcal{C}$ contains just B. By definition block B weighs 1 for P. Thus, by Lemma 18 the adversary can start extending $\mathcal{C}$ at round 2. However, the blocks that the adversary uses to slow down chain growth cannot also be used for $\mathcal{C}$, because they belong to chains whose genesis block has been announced earlier. So let $Z_2(S)$ denote the blocks mined by the adversary in S, that are not in $Z_1(S)$. In order for $\mathcal{C}$ to be accepted by some honest party, it must hold that its weight is bigger than that of other chains received:

$$1+3(Z_2(S)-1) \geq l-(1-\delta)\kappa/\gamma-2+3(Y(S')-1)-Z_1(S).$$

Since the blocks counted in $Z_1(S)$ and $Z_2(S)$ are different, and $S' \subseteq S$, the above implies:

$$3Z(S) \geq l - (1-\delta)\kappa/\gamma + 3Y(S') - 3. \tag{3.1}$$

Now, by the typicality of the execution, it holds that $Z(S') < Y(S')$, since $|S'| \geq (1-\delta)\kappa$, and $Z(S \setminus S') < (1+\delta)l\beta$, since $|S \setminus S'| = l \geq (1-\delta)\kappa$. Then, since by our assumptions we have that

$$3(1+\delta)l\beta < l - (1-\delta)\kappa/\gamma - 3, \tag{3.2}$$

it follows that inequality 3.1 cannot hold and thus no honest party will ever accept a chain based on B at round r'. By exactly the same arguments for round $r'+1$, we can show that no honest party will adopt a chain that starts with B.

Note now, that this would imply that Lemma 19 should hold for $r, s \leq 2l+2$. Then, once again we can apply the same techniques as above for round $r'+2$, and show that no honest party will adopt a chain starting with B. Inductively, it can be shown that this fact holds for any r, and thus the lemma follows.

□

The complete version of the weak chain growth lemma follows from the argument we've made above.

Corollary 21. *Assume a typical execution and Assumption 1. Let rounds r, s be such that $l+2 \leq r \leq s$, and let $S = \{r, \ldots, s-1\}, S' = \{2, \ldots, s-1\}$. Suppose that at round r an honest party, say, P_1 has a chain $\mathcal{C}$ such that $w_{P_1}(\mathcal{C}) = d$. Then, by round s, every honest party P will have received a chain $\mathcal{C}'$ such that $w_P(\mathcal{C}') \geq d - 2 + 3 \cdot Y(S) - Z(S')$.*

A bound on adversarially precomputed blocks. The honest parties begin mining right after the challenge-exchange phase. Note that it does not help the adversary to precompute blocks before the challenge-exchange phase, except for the small probability of the event that some of his blocks happen to extend future blocks. We have shown that the adversary cannot create a private chain that honest parties will adopt if he starts mining at the first round of the challenge-exchange phase. It is though possible to start mining *after* the first round in order to gain some advantage over the honest parties. The following lemma provides a bound on the number of blocks mined during the challenge-exchange phase with sufficient weight so that they can be later used by the adversary.

Lemma 22 (Precomputed blocks)**.** *Assume a typical execution and Assumption 1. Let R be the set that contains any adversarial block B mined before round $l+2$, where $h(B) > l-1-(1-\delta)\delta^2\kappa$. Then, $|R| < \frac{5\delta}{9}\kappa\beta$.*

Proof. Let $B = \text{head}(\mathcal{C})$ be a block that is contained in R and B' be the genesis block of $\mathcal{C}$. First, we are going to show that if B' was computed before round $l+1-2(1-\delta)\delta^2\kappa$, then B must have been computed after round $l+1-(1-\delta)\delta^2\kappa$. Suppose that block B' was computed at round $l+1-r^*$ of the challenges phase, for $r^* > 2(1-\delta)\delta^2\kappa$; thus, by Lemma 18 it holds that $h_{\mathcal{C}}(B') < l-r^*$. In order for $\mathcal{C}$ to have the required weight, the adversary must have mined at least

$$\lceil ((l-1-(1-\delta)\delta^2\kappa)-(l-r^*-1))/3\rceil = \lceil (r^*-(1-\delta)\delta^2\kappa)/3\rceil$$

blocks, starting from round $l+1-r^*$. Due to the typicality of the execution, it holds that the adversary after $r^*-(1-\delta)\delta^2\kappa$ rounds has computed at most $(1+\delta)(r^*-(1-\delta)\delta^2\kappa)\beta$ blocks. By our assumptions, $\beta < f/2 \leq 1/6$, and thus this number of blocks is not sufficient to get a chain of the required weight, that is

$$(1+\delta)(r^*-(1-\delta)\delta^2\kappa)\beta < \lceil (r^*-(1-\delta)\delta^2\kappa)/3\rceil.$$

Therefore, the adversary will start mining B after round $l+1-(1-\delta)\delta^2\kappa$, for any B in R.

Next, we are going to bound the number of blocks the adversary can compute in $2(1-\delta)\delta^2\kappa$ rounds; recall that we are interested in blocks that were mined after round $l-2(1-\delta)\delta^2\kappa$ and before round $l+2$. Due to typicality, the adversary mines at most $(1+\delta/9)\cdot 2(1-\delta)\delta^2\kappa\beta(< \frac{5\delta}{9}\kappa\beta)$ blocks in $2(1-\delta)\delta^2\kappa$ rounds. The lemma follows. □

We are now ready to prove the common prefix, chain quality and chain growth properties.

Common Prefix. Every time a uniquely successful round happens all honest players converge to one chain, unless the adversary diffuses some new block. This turns out to be a very important fact and a consequence of it is described in the next lemma.

Lemma 23. *Suppose block B in chain $\mathcal{C}$ is a u.s. block and consider a chain $\mathcal{C}'$ such that $B \notin \mathcal{C}'$. If $\ell_{\mathcal{C}'} \geq \ell_{\mathcal{C}}(B)-1 \geq 0$, then there exists a unique adversarial block $B' \in \mathcal{C}'$ such that $\ell_{\mathcal{C}'}(B') \in [\ell_{\mathcal{C}}(B)-1, \ell_{\mathcal{C}}(B)+1]$. Moreover, if B is not a genesis block, then B' will also not be a genesis block.*

Proof. Assume block B was mined at some round r. If B is not a genesis block, then for any honest block B'' mined before round r it should hold that $\ell(B'') \leq \ell(B) - 2$. Otherwise, at round r no honest party would choose the parent of B to mine new blocks. If B is a genesis block, then no other honest party has mined a block in some previous round. On the other hand, for any honest block B'' mined after round r it must hold that $\ell(B'') \geq \ell(B) - 1 + 3 = \ell(B) + 2$, since honest parties will only extend chains of length at least $\ell(B) - 1$ after this round. Thus, if a block with weight in the given interval exists, it must be adversarial.

We next proceed to argue that such a block always exists. The only reason that such a block may not exist, is if $\ell_{\mathcal{C}'}(\text{tail}(\mathcal{C}')) > \ell_{\mathcal{C}}(B) + 1$, which for the sake of contradiction we assume. First, take the case where B is a genesis block. By our assumption, it is implied that all honest parties receive $\text{tail}(\mathcal{C}')$ before B. Hence, no honest party would compute another genesis block, which is a contradiction. For the other case, suppose B is not a genesis block, and let B'' be the parent of B. Then, $h_{\mathcal{C}}(B'') \leq \ell_{\mathcal{C}}(B) - 2$, and thus $h_{\mathcal{C}}(B'') < \ell_{\mathcal{C}'}(\text{tail}(\mathcal{C}'))$. This implies than every honest party received $\text{tail}(\mathcal{C}')$ before block B''. But then, no honest party would mine on the parent of B, because it would have lower weight than B', which leads to a contradiction. Note, that the last argument holds, even if without our initial assumption, i.e., by just assuming B' is a genesis block. Hence, the lemma follows. □

We use Lemma 23 in order to show that the existence of a fork implies that the adversary has mined blocks proportional in number to the time the fork started.

Theorem 24. *Assume a typical execution and Assumption 1. Then, the strong common-prefix property holds with parameters $k \geq 2\kappa f$ and $r_0 = 2\ell + 2$.*

We first prove a weaker lemma, called common-prefix lemma in [GKL15]. After proving this lemma for our own model, we can apply the same ideas as in [GKL15] to get a proof for the theorem.

Lemma 25. *Assume a typical execution and Assumption 1, and consider two chains $\mathcal{C}_1$ and $\mathcal{C}_2$. If $\mathcal{C}_1$ is adopted by an honest party at a round after round $2l + 2$, and $\mathcal{C}_2$ is either adopted by an honest party or diffused at the same round, and has weight $\ell_{\mathcal{C}_2} \geq \ell_{\mathcal{C}_1}$, then $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \preceq \mathcal{C}_1$, for $k \geq 2\kappa f$.*

Proof. We first show that a fork between two chains implies that the adversary must have mined a number of blocks proportional to the uniquely successful blocks associated with these chains.

Claim 2. *Let $\mathcal{C}_1, \mathcal{C}_2$ be two chains diffused at some point of the execution, and $B'_0, \ldots, B'_k$ be u.s. blocks in chains $\mathcal{C}'_0, \ldots, \mathcal{C}'_k$ in increasing order of round mined. Then, if $\ell_{\mathcal{C}_1 \cap \mathcal{C}_2} < \ell(B'_0) - 1$ and for all $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}'_k$: $\ell(B'_k) - 1 \leq \ell_{\mathcal{C}}$, there exist different adversarial blocks $B_0, \ldots, B_k$ such that for $i \in \{0, \ldots, k\}$: $\ell(B_i) \in [\ell(B'_0) - 1, \ell(B'_k) + 1]$ and $B_i \in \{\mathcal{C}'_0, \ldots, \mathcal{C}'_k\} \cup \{\mathcal{C}_1, \mathcal{C}_2\}$.*

Proof of Claim. We iterate over $U = \{(B'_0, \mathcal{C}'_0), \ldots, (B'_k, \mathcal{C}'_k)\}$ in the order of increasing index. Note that by the Claim in Lemma 19, if $(B, \mathcal{C})$ and $(B', \mathcal{C}')$ are two consecutive elements of U, then

$$\ell_{\mathcal{C}'}(B') - \ell_{\mathcal{C}}(B) \geq 2 \tag{3.3}$$

Consider $(B, \mathcal{C}) \in U$ and suppose all the previous elements have been associated with a distinct adversarial block. In particular, let $(\bar{B}, \bar{\mathcal{C}})$ be the previous one associated to $(\bar{B}', \bar{\mathcal{C}}')$. To choose the adversarial block $(B', \mathcal{C}')$ to associate with $(B, \mathcal{C})$ we consider the following cases (an example is presented in Figure 3.4 covering most of the cases). In each case, we define what $\mathcal{C}'$ should be, and B' is determined by Lemma 23; our assumptions ensure that the preconditions of the lemma are always satisfied.

- If $\mathcal{C} \notin \{\mathcal{C}_1, \mathcal{C}_2\}$ and $\bar{\mathcal{C}} \in \{\mathcal{C}_1, \mathcal{C}_2\}$, then we have two cases. If $\bar{B}' \notin \mathcal{C}$, then let $\mathcal{C}' = \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \bar{\mathcal{C}}'$. Moreover, let B^* in chain $\mathcal{C}^* = \mathcal{C}$ be the block guaranteed from Lemma 23 for block $\bar{B}$ in chain $\mathcal{C}$. This block will be used as a matching in a subsequent step. Otherwise, $\bar{B}' \in \mathcal{C}$. Then $\mathcal{C}'$ should be chosen appropriately from $\{\mathcal{C}_1, \mathcal{C}_2\}$, so that in the next transition from a chain not in $\{\mathcal{C}_1, \mathcal{C}_2\}$ to a chain in $\{\mathcal{C}_1, \mathcal{C}_2\}$, the corresponding block from Lemma 23 does not intersect with the previously chosen block. This is always possible since any of the two chains can be selected.

- If $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\}$ and $\bar{\mathcal{C}} \notin \{\mathcal{C}_1, \mathcal{C}_2\}$, then $\mathcal{C}' \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}$. If $\mathcal{C}^*$ is defined from a previous application of the first rule, we match B with B^*.

- If $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\}$ and $\bar{\mathcal{C}} \in \{\mathcal{C}_1, \mathcal{C}_2\}$, then let $\mathcal{C}' \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}$.

- If $\mathcal{C} \notin \{\mathcal{C}_1, \mathcal{C}_2\}$ and $\bar{\mathcal{C}} \notin \{\mathcal{C}_1, \mathcal{C}_2\}$, then let $\mathcal{C}' \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \bar{\mathcal{C}}'$. The adversarial block guaranteed by Lemma 23 for $\mathcal{C}'$ is not common to $\mathcal{C}_1, \mathcal{C}_2$ due to $\ell_{\mathcal{C}_1 \cap \mathcal{C}_2} < l(B'_0) - 1$.

- If $B = B_0$, then if $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\}$, $\mathcal{C}' \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}$. Otherwise, as in the first case, $\mathcal{C}'$ should be chosen appropriately from $\{\mathcal{C}_1, \mathcal{C}_2\}$.

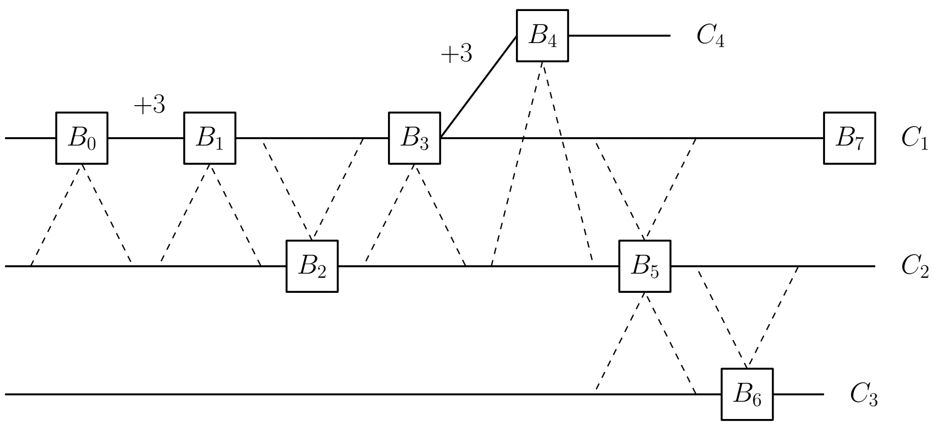


Figure 3.4: An example of the matching described in the claim in Lemma 25. $B_0, \dots, B_7$ are u.s. blocks and the dashed triangles point to the places where according to Lemma 23 there should exist an adversarial block. In this example we can use Lemma 23 two times for block B_5. The "extra" block is matched to B_7.

We need to verify that the above procedure does not assign the same block to two distinct elements of U, $(B_u, \mathcal{C}_u)$ and $(B_v, \mathcal{C}_v)$.

Note first that this is not possible if

$$|\ell_{\mathcal{C}_u}(B_u) - \ell_{\mathcal{C}_v}(B_v)| \geq 3.$$

For example, by Equation (3.3), this is true if they are not consecutive in U. To see this, observe that by Lemma 23,

$$\ell_{\mathcal{C}'_u}(B'_u) \in [\ell_{\mathcal{C}_u}(B_u) - 1, \ell_{\mathcal{C}_u}(B_u) + 1],$$

while

$$\ell_{\mathcal{C}'_v}(B'_v) \in [\ell_{\mathcal{C}_v}(B_v) - 1, \ell_{\mathcal{C}_v}(B_v) + 1].$$

Since these intervals are disjoint due to the inequality above, it follows that $B'_u \neq B'_v$.

Thus, we only need to consider the case

$$\ell_{\mathcal{C}_{u+1}}(B_{u+1}) - \ell_{\mathcal{C}_u}(B_u) = 2.$$

It is not hard to see that this situation cannot occur when B_{u+1} is a descendant of B_u. Moreover, when the blocks assigned are on different chains, it is guaranteed that they are different. All cases are directly covered by these two observations, except one: when the second rule is applied. We have two cases. In the first one, B^* is not defined (or used) by some earlier application of the first rule. Then, it follows that B_{u-2}, if defined, is not in any of $\{\mathcal{C}_1, \mathcal{C}_2\}$, and thus B'_{u-1}, the corresponding adversarial block, can be in any of $\{\mathcal{C}_1, \mathcal{C}_2\}$. Hence, B'_{u-1}, B_u will be in different chains. Otherwise, B^* is matched to B_u, which never coincides with any other block in the matching, as the previous adversarial block matched is never on the same chain. Hence, the claim follows.

$\dashv$

Now we can proceed to the core of the proof. The idea is that if a fork exists, we will use the previous claim multiple times and get a matching between a sufficiently large amount of uniquely successful blocks and adversarial blocks. Then, we will show that it is impossible for the adversary to mine that number of blocks.

Let block $B'_0 \in \mathcal{C}_1 \cap \mathcal{C}_2$ be the u.s. block mined the latest, where all subsequently mined u.s. blocks are descendants of B'_0. If no such block exists, assume there exists a block B'_0, mined in round 0, that is the parent of all genesis blocks, as in Figure 3.5, but does not affect the weight of the chains it belongs too. Thus, in this case, whenever we write for example $\mathcal{C}_1$ we mean $\mathcal{C}_1$ augmented with B'_0. In any case, it holds that any u.s. block mined after B'_0 will be a descendant of B'_0, and $B'_0 \in C_1 \cap C_2$.

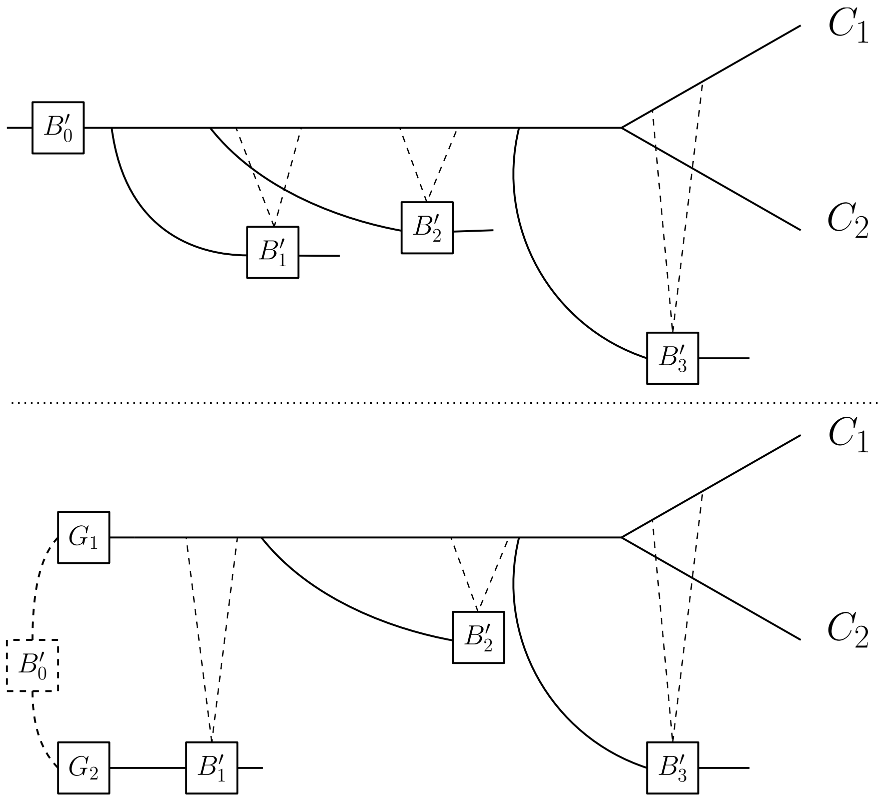


Figure 3.5: An example of two scenarios for the matching described by Lemma 25. Notice that in the second one an artificial block B'_0 has been introduced to aid our analysis.

Now, if possible, let block $B'_1 \neq B'_0$, the head of some chain $\mathcal{C}'_1$, be the u.s. block mined the latest, where (1) the last block in $\mathcal{C}_1 \cap \mathcal{C}'_1$ is in $\mathcal{C}_1 \cap \mathcal{C}_2$ and (2) for any u.s. block B' that is the head of chain $\mathcal{C}'$, and B' is mined after B'_0, it holds that the last block in $\mathcal{C}_1 \cap \mathcal{C}'_1$ is the same or an ancestor of the last block in $\mathcal{C}_1 \cap \mathcal{C}'$. Note that $B'_1 \notin \mathcal{C}_1$, otherwise it would satisfy the definition of B'_0 which is a contradiction. Let B' be the first u.s. block mined after B'_0 that is in some chain C'. B' is a descendant of the last block of $\mathcal{C}_1 \cap \mathcal{C}'_1$. Otherwise, by the definition of B'_1, it would be equal to the last block of $\mathcal{C}_1 \cap \mathcal{C}'_1$, and would satisfy the definition of B'_0 which is a contradiction. It follows that $\ell_{\mathcal{C}_1 \cap \mathcal{C}'_1} \leq \ell_{\mathcal{C}'}(B') - 3 < \ell_{\mathcal{C}'}(B') - 1$. Additionally, since some honest party has chain $\mathcal{C}_1$ at round r, it holds that $\ell_{\mathcal{C}_1} \geq \ell_{\mathcal{C}'}(B') - 1$. Thus, we can apply Claim 2 for the chains $\mathcal{C}_1$ and $\mathcal{C}'_1$ and for the u.s. blocks mined after B'_0, and up to B'_1.

We apply this process as many times as possible. B'_2, the head of some chain $\mathcal{C}'_2$, is the most recently mined u.s. block where (1) the last block in $\mathcal{C}_1 \cap \mathcal{C}'_2$ is in $\mathcal{C}_1 \cap \mathcal{C}_2$ and (2) for any u.s. block B' in chain $\mathcal{C}'$ mined after B'_1 it holds that the last block in $\mathcal{C}_1 \cap \mathcal{C}'_2$ is the same or an ancestor of the last block in $\mathcal{C}_1 \cap \mathcal{C}'$. Then we can again apply Claim 2 for the chains $\mathcal{C}_1$ and $\mathcal{C}'_2$, from the first honest block mined in a u.s. round after B'_1 and up to B'_2.

We will argue that the adversarial blocks matched in the two applications of Claim 2 so far will be different. Let B' be the next u.s. block mined after B'_1. B' will be a descendant of the (real) genesis block of $\mathcal{C}_1$. Notice that in the worst case B'_1 has been matched to a block B'' in $\mathcal{C}_1$. Hence, $l(B') - l(B'') = 3d$, for some $d \in \mathbb{N}$, since they share the same genesis block. If $l(B') - l(B'') = 0$, then if follows that $l(B') - l(B'_1) < 2$ which is impossible. Otherwise, $l(B') - l(B'') \geq 3$, and thus the block that is going to be matched to B' by Lemma 23 cannot be B''. The same argument applies, for the rest of the blocks matched with this process. The process ends when no block B'_i, for some positive i, with the desired properties exists. Notice that it may be the case that the process ends for $i = 0$, no block matching the specification of B'_1 exists.

So for any remaining u.s. block B that is the head of some chain $\mathcal{C}$, mined after B'_i, it holds that the last block of $\mathcal{C} \cap \mathcal{C}_1$ is not in $\mathcal{C}_1 \cap \mathcal{C}_2$. Moreover, since $\mathcal{C}_1, \mathcal{C}_2$ have been adopted by honest parties up to round r, it holds that $\ell_{\mathcal{C}_2} \geq \ell_{\mathcal{C}_1} \geq \ell(B) - 1$. Hence, we can apply Claim 2 for chains $\mathcal{C}_1, \mathcal{C}_2$ and for all remaining u.s. blocks. Thus, there exists a mapping between all u.s. blocks mined after B'_0 and distinct adversarial blocks that are descendants of B'_0.

Now, for the sake of contradiction, assume that $\mathcal{C}_1$ diverges from $\mathcal{C}_2$ at some round $r' \leq r - \kappa$, i.e., the last block of the their common prefix is computed before round

r'. We have two cases. If B'_0 actually exists, all adversarial blocks of the matching must have been mined on and after the round B'_0 was mined. Also, since $B'_0 \in \mathcal{C}_1 \cap \mathcal{C}_2$, it must have been mined before round r'. Therefore, there exists a set of rounds $S = \{r_0, \ldots, r', \ldots, r\}$ such that $Z(S) \geq Y(S \setminus \{r_0\})$ and $|S| \geq \kappa$. This contradicts the typicality of the execution.

In the second case, B'_0 is not a real block, and the adversary may use blocks that he has precomputed during the challenge-exchange phase. By the typicality of the execution, the honest parties have computed at least one genesis block that weighs at least $l - (1-\delta)\delta^2\kappa$. Then, all adversarial blocks in the matching must weigh at least $l - (1-\delta)\delta^2\kappa - 1$, and by Lemma 22 there are at most $5\delta/9 \cdot \beta\kappa$ such blocks computed during the challenge exchange phase. This implies that for $S = \{1, \ldots, r\}$, it holds that $Y(S) \leq 5\delta/9 \cdot \beta\kappa + Z(S)$. By Lemma 15, this is a contradiction. Hence, $\mathcal{C}_1$ and $\mathcal{C}_2$ diverge after round $r - \kappa$.

By the typicality of the execution, at most $(1+\delta)\kappa f (< 2\kappa f \leq k)$ blocks have been mined after round $r - \kappa$. Hence, the fork between $\mathcal{C}_1$ and $\mathcal{C}_2$ is less that k blocks long, and the lemma follows.

□

Next, using the common-prefix lemma we can prove that our protocol satisfies the strong common-prefix property.

Proof of Theorem 24. The proof is similar to that of [GKL15]. Essentially, we proved the common-prefix lemma and now we want a proof for the common-prefix theorem in their terms. We give a brief sketch of the proof, noting any differences that arise in our model.

Assume that there exist chains $\mathcal{C}_1, \mathcal{C}_2$ adopted by parties P_1, P_2 at rounds r_1, r_2 such that $2\ell + 2 < r_1 \leq r_2$ and $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$. Let $r \in [r_1, r_2]$ be the smallest round such that there exists an honest party P' that adopts a chain $\mathcal{C}'_2$ such that $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}'_2$. If $r = r_1$, then it holds that either $\ell_{\mathcal{C}_1} \leq \ell_{\mathcal{C}'_2}$ or $\ell_{\mathcal{C}'_2} \leq \ell_{\mathcal{C}_1}$, which is a contradiction to Lemma 25.

Otherwise, if $r > r_1$, let $\mathcal{C}'_1$ be the chain that party P' adopted at round $r-1$. Then, it holds that

$$(\mathcal{C}_2'^{\lceil k} \preceq \mathcal{C}'_1) \wedge (\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}'_1) \wedge (\ell(\mathcal{C}_2'^{\lceil k}) \geq \ell(\mathcal{C}_1^{\lceil k})) \implies \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2'^{\lceil k}$$

The first conjunct follows from the fact that $\mathcal{C}'_2$ was diffused at round $r-1$, and Lemma 25. The second one follows by the definition of r. The last one follows from the fact that P' selected chain $\mathcal{C}'_2$ despite knowing $\mathcal{C}_1$, hence it holds at least that $h_{\mathcal{C}'_2} > \ell_{\mathcal{C}_1}$, which in turn implies the third conjunct. The last implication follows from the fact that $\mathcal{C}_1^{\lceil k}$ and $\mathcal{C}_2'^{\lceil k}$ are part of the prefix of $\mathcal{C}_1'$, and $\mathcal{C}_1$ is shorter or of equal length to $\mathcal{C}_2'$ (remember that they both share the same genesis block). This is a contradiction, and the theorem follows.

□

Chain Growth. We proved that after round $2l+1$ the strong common-prefix property is satisfied. This implies that all parties share a common genesis block after this round. The next lemma shows that this is sufficient in order to get chain growth at the same level as in the original Backbone protocol after this round.

Lemma 26. *Assume a typical execution, and that at round r an honest party P_1 has a chain $\mathcal{C}$ of weight $w_{P_1}(\mathcal{C}) = d$, and all honest parties after round $r-1$ adopt chains that share the same genesis block B. Then, by round $s \geq r$, every honest party P will have received a chain $\mathcal{C}'$ such that $w_P(\mathcal{C}') \geq d-1+3\sum_{i=r}^{s-1} X_i$.*

Proof. Since all parties adopt chains with the same genesis block after round $r-1$, and P_1 has adopted a chain $\mathcal{C}$ of weight d, there are two cases: either (1) $\ell_{\mathcal{C}} = d-1$ and any chain that honest parties adopt after round $r-1$ has a weight that is congruent to d or $d-1$ modulo 3, or (2) $\ell_{\mathcal{C}} = d$ and the weight is congruent to d or $d+1$ modulo 3. This observation is implied from the fact that each extra block adds 3 units of weight to the chain and B can only have two different weights under the views of honest parties.

It is sufficient to study only one of the two cases so w.l.o.g. suppose that the weight of the chains is congruent to d or $d-1$ modulo 3. The proof is by induction on $s-r \geq 0$. For the basis ($s=r$), observe that if at round r P_1 has a chain $\mathcal{C}$ of weight $w_{P_1}(\mathcal{C}) = d$, then he broadcast $\mathcal{C}$ at an earlier round (than r). It follows that every honest party P will receive $\mathcal{C}$ by round r and $w_P(\mathcal{C}) \geq d-1$.

For the inductive step, note that by the inductive hypothesis every honest party P has received a chain $\mathcal{C}'$ of weight at least $w_P(\mathcal{C}') = d' \geq d-1+3\sum_{i=r}^{s-2} X_i$ by round $s-1$. When $X_{s-1} = 0$ the statement follows directly, so assume $X_{s-1} = 1$. Observe that every honest party queried the oracle with a chain of weight at least d' at round $s-1$. It follows that every honest party P successful at round $s-1$ broadcast a chain $\mathcal{C}'$ of weight at least $w_P(\mathcal{C}') = d'+3$. For every other party P' it holds that $w_{P'}(\mathcal{C}') \geq d'+2 \geq d-1+3\sum_{i=r}^{s-1} X_i - 1$. However, no chain that an honest party adopts can have length $d'+2$, because $d'+2$ is congruent to $d-2$ modulo 3. Thus all honest parties adopt chains that have length at least $d'+3$ and the lemma follows. □

It can be easily shown that Lemma 26 implies the chain growth property after round $2l+1$.

Theorem 27. *Assume a typical execution and Assumption 1. The bootstrapped Bitcoin protocol satisfies the chain growth property for* $r_0 = 2l+2$, *speed coefficient* $\tau = (1-\delta)\gamma$ *and* $s \geq \kappa$.

Chain Quality. We first observe a consequence of Theorem 24.

Lemma 28. *Assume a typical execution and Assumption 1. From round* $2l+2$ *and onwards of the bootstrapped backbone protocol, the adversary never has a chain which is more than* $k \geq 2\kappa f$ *blocks longer than the chain of some honest party in the next round, and shares the same genesis block.*

Proof. Given any execution and an adversary that at a round r has a chain $\mathcal{C}$ which is k blocks longer than the chain $\mathcal{C}'$ of an honest party P, we can define an adversary such that at round $r+1$ the common-prefix property does not hold for parameter k. The adversary simply sends $\mathcal{C}$ to $P' \neq P$ at round r. □

Theorem 29. *Assume a typical execution and Assumption 1. Suppose* $\mathcal{C}$ *belongs to an honest party and consider any* $k \geq 16\kappa f/\gamma\delta$ *consecutive blocks of* $\mathcal{C}$ *computed after round* $2l+2$ *of the bootstrapped backbone protocol. Then, the adversary cannot have contributed more than* $(1+\frac{\delta}{2})\frac{\beta}{\gamma}\cdot k \leq (1-\frac{\delta}{4})k$ *of these blocks.*

Proof. The proof idea is as follows. Suppose that in a chain $\mathcal{C}$ a large number of consecutive blocks exist, such that a large fraction of them were computed by the adversary. Then it must be that the honest parties have contributed their blocks to other chains. Since we are at a round sufficiently away from the beginning of the protocol (at least $2l+2$), we know (by Lemma 26) that the chains of the honest parties advance analogously to their power. We obtain then a contradiction by setting up the parameters so that $\mathcal{C}$ is shorter than what Lemma 26 implies.

Let us denote by B_i the i-th block of the chain $\mathcal{C}$ of an honest party P at some round r and consider any k consecutive blocks $B_u,\ldots,B_v$. Define K as the least number of consecutive blocks $B_u,\ldots,B_w$ that include the k given ones (i.e., $v \leq w$) and have the property that there exists a round at which an honest party was trying to extend the chain ending at block B_w. Observe that K is well defined since $\mathcal{C}$ belongs to an honest party. Define also r_1 as the round that B_u was created, r_2 as the first round that an honest party attempts to extend B_w, and let $S = \{r : r_1 \leq r < r_2\}, S' = S \setminus \{r_1\}$.

Now let x denote the number of blocks from honest parties that are included in the k blocks and—towards a contradiction—assume that

$$x \leq \Big[1 - \Big(1 + \frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big]k \leq \Big[1 - \Big(1 + \frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big]K.$$

Suppose first that all the K blocks $\{B_j : u \leq j \leq w\}$ have been computed during the rounds in the set S. Then,

$$Z(S) \geq K - x \geq \Big(1 + \frac{\delta}{2}\Big)\frac{\beta}{\gamma}K \geq \Big(1 + \frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big(X(S') - 2\kappa f\Big)$$

The first inequality comes from the fact that the adversary computed $K - x$ of the K blocks, since by the definition of B_w all blocks from B_v up to the last block before B_w must be adversarial. The second one comes from the postulated relation between x and K. To see the last inequality, assume $X(S') - 2\kappa f > K$. Note first, that by Lemma 28 all honest parties at round r_1 have chains with weight at most $3 \cdot 2\kappa f$ less than the weight up to B_u. Next, note that by Theorem 24 after round r_1 all honest parties have chains starting with the same genesis block. Thus, assuming $X(S') - 2\kappa f > K$, the assumption that an honest party is on B_w at r_2 contradicts Lemma 26.

To obtain the stated bound, note that if $|S| < K/(1+\delta)f$, then, since f is bounded away from 1 by a constant, by the typicality of the execution, in $|S|$ rounds the total number of solutions is less than K. Otherwise, $|S| \geq K/(1+\delta)f \geq k/2f \geq 8\kappa/\gamma\delta$ and the bound follows from the typicality of the execution, since (using $\mathbb{E}[X(S)] \geq \gamma|S|$, and $(1+\frac{\delta}{2})(1-\frac{\delta}{4}) \geq (1+\frac{\delta}{8})$)

$$Z(S) \geq \Big(1 + \frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big(X(S') - 2\kappa f\Big) > (1 + \frac{\delta}{2})(1 - \frac{\delta}{4})\beta|S| \geq (1 + \frac{\delta}{8})\beta|S| \geq Z(S).$$

To finish the proof we need to consider the case in which these K blocks contain blocks that the adversary computed in rounds outside S. By the typicality of the execution, this event is impossible. □

Corollary 30. *Assume a typical execution and Assumption 1. The bootstrapped Bitcoin protocol satisfies the chain-quality property with parameters* $\mu = (1+\frac{\delta}{2})\frac{\beta}{\gamma}$, $\ell_0 = 2f(1+\delta)(l+1)$, *and* $\ell \geq 16\kappa f/\gamma\delta$.

Proof. By the typicality of the execution, both the honest parties and the adversary have computed at most $(1+\delta)f \cdot 2(l+1)$ blocks. The statement then follows from Theorem 29. □

3.4 Applications of the Bootstrapped Backbone Protocol

In this section we present applications of our construction, starting with its primary/original one: a distributed ledger, i.e., a public and permanent summary of all transactions that honest parties can agree on as well as add their own, despite the potentially disruptive behavior of parties harnessing less than $1/2$ of the hashing power. This entails proving that the ledger's required security properties (Persistence and Liveness — cf. [GKL15]) hold in a genesis block-less setting.

Next, we consider the problem of setting up a PKI in our unauthenticated network setting *from scratch*, i.e., without any trusted setup. As mentioned in Section 3.1, the idea of using POWs as an identity-assignment tool was put forth by Aspnes *et al.* [AJK05]. Here we build on this idea as well as on the "2-for-1 POWs" technique from [GKL15] to use our bootstrapped protocol to assign identities to parties. The assignment relation will possibly assign more than one identities to the same party, while guaranteeing that the majority of them is assigned to honest parties.

Finally, applying the 2-for-1 POWs technique we can also solve the consensus (aka Byzantine agreement) problem [PSL80, LSP82] without any trusted setup, even if the adversary has almost the same hashing power as the honest parties, and in a number of rounds independent of the number of parties. Indeed, all our protocols have round complexity linear in the security parameter, and enjoy simultaneous termination.

Compared to other works, most notably [AD15], our approach is different in the order in which it sets up a "bulletin board" and assigns identities to parties. We choose to first establish the former—i.e., the ledger—and then assign the identities; in contrast, in [AD15] identities are established first in a graded manner, and then using that infrastructure the parties can implement a broadcast channel. We now turn to the applications in detail.

Robust public transaction ledger. In order to turn the backbone protocol into a protocol realizing a public transaction ledger, suitable definitions were given for functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ in [GKL15], we recite them in Table 3.2. Namely, $\mathrm{V}(\langle x_1, \ldots, x_m)$ is true if its input is a valid ledger. Function $\mathrm{R}(\mathcal{C})$ returns the contents of the chain if they constitute a valid ledger, otherwise it is undefined. Finally, $\mathrm{I}(st, \mathcal{C}, round, \mathtt{INPUT}(), \mathtt{RECEIVE}())$ returns the largest subsequence of transactions in the input and receive tapes that constitute a valid ledger, with respect to the contents of the chain the party already has, together with a randomly generated neutral transaction. We denote the instantiation of our protocol with these functions by $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$.

Content validation predicate $\mathrm{V}(\cdot)$	$\mathrm{V}(\cdot)$ is true if its input $\langle x_1,\ldots,x_m\rangle$ is a valid ledger, i.e., it is in $\mathcal{L}$.
Chain reading function $\mathrm{R}(\cdot)$	$\mathrm{R}(\cdot)$ returns the contents of the chain if they constitute a valid ledger, otherwise it is undefined.
Input contribution function $\mathrm{I}(\cdot)$	$\mathrm{I}(\cdot)$ returns the largest subsequence of transactions in the input and receive registers that constitute a valid ledger, with respect to the contents of the chain the party already has, preceded by a neutral random transaction.

Table 3.2: *The instantiation of functions* $\mathrm{I}(\cdot),\mathrm{V}(\cdot),\mathrm{R}(\cdot)$ *for* $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$.

Chain quality, chain growth and the strong common prefix property were shown in [KP15, PSS17] to be sufficient to implement a robust ledger in a black-box manner. Our protocol satisfies all these properties after a specific condition is met. Chain quality holds after the $2f(1+\delta)(l+1)$ block in the chain of any party, as Corollary 30 dictates, and common prefix and chain growth hold after round $2l+2$, according to Theorems 24 and 27. Due to chain growth, after at most $(2(1+\delta)(1-\delta)f/\gamma+2)(l+1) \leq 14(l+1)$ rounds all necessary conditions will have been met with overwhelming probability. We omit the proofs of the following theorems, as they follow closely the ones found in [GKL15].

Lemma 31 (Persistence)**.** *Assume a typical execution and Assumption 1. Then, protocol* $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$ *satisfies Persistence after round* $2l+2$*, for* $k \geq 2\kappa f$*.*

Lemma 32 (Liveness)**.** *Assume a typical execution and Assumption 1. Further, assume oracle* Txgen *is unambiguous. Then, protocol* $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$ *satisfies Liveness after round* $14(l+1)$*, with wait time* $u = \frac{2k}{(1-\delta)\gamma}$ *rounds and depth parameter* $k \geq \frac{16\kappa f}{\gamma\delta}$*.*

Corollary 33. *If Assumption 1 holds, then protocol* $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$ *implements a robust transaction ledger with parameters* $r_0 = 14(l+1)$*,* $u = \frac{2k}{(1-\delta)\gamma}$*, and* $k \geq \frac{16\kappa f}{\gamma\delta}$*, with overwhelming probability in* κ*.*

Fast PKI setup. Next, we use the ledger to generate an honest majority PKI from scratch in a number of rounds that is linear in the security parameter.

Eyal et al. in [ES14] presented the selfish mining attack and showed that under adversarial influence the "main chain" blocks mined by honest parties may be less than their proportion of hashing power. On the other hand, for our applications we require proportional representation, e.g., if honest parties have a majority of the hashing power, they should also have a majority of the keys of the PKI generated. To address this limitation of the protocol, we make use of the *2-for-1 PoW* technique described in [GKL15].

At a high level, the technique allows to do combined mining for two PoW schemes in the price of one. In more detail, we can add additional information in the queries to the random oracle, and if the response to the query is less than some value D_1, then we consider it a valid PoW of type 1; if its reverse is greater than some value D_2 we consider it as a valid PoW of type 2. D_1 and D_2 should be appropriately chosen so that the events of success in either of these PoWs are independent. The second PoW is used to "mine" transactions, in the same way blocks are mined. This guarantees that the *number of transactions* is proportional to the hashing power of each party. By having parties broadcast their transactions on one hand, and making sure that at least one honest block that contains these transactions is in the chain of all honest parties due to liveness on the other hand, the protocol in [GKL15] manages to achieve consensus assuming honest-majority of hashing power. Note, that still the selfish mining attack applies, but it does not affect proportional representation on the transaction level.

In our case, transactions will contain the public keys, and in this way we will obtain an honest-majority PKI. However, in contrast with [GKL15], we cannot let parties start mining transactions from the beginning of the execution, since the adversary would have some additional precomputation time. Instead, we are going to wait for the public ledger to be established, and then use some of the blocks added by honest parties to guarantee that all transactions where mined recently enough. In more detail, any PoW will be represented by a triple $\langle w, ctr, label\rangle$. The verification procedure for "block level" PoWs ("block PoWs" for short) will be of the form

$$H(ctr, \langle G(w), label\rangle) < D_1,$$

while the verification procedure for the "transaction level" PoWs will be of the form

$$[H(ctr, \langle label, G(w)\rangle)]^R < D_2,$$

where $[a]^R$ denotes the reverse of the bitstring a. In w we are going to encode the information needed for each application. For example, in block PoWs, w will contain the transactions related to this block as well as the hash of the previous block. Note that by making one hash query of the form $H(ctr, \langle G(w_0), G(w_1)\rangle)$ and only two comparisons, we will be mining PoWs of both types at the same time. Moreover, if $\lceil \log(D_1) \rceil + \lceil \log(D_2) \rceil$ is less than κ, where κ is the size of the hash's output, then the events of succeeding in any of the two PoWs are independent, since they depend on different bits of the hash which are sampled independently and uniformly at random by the random oracle.

Next, we describe our protocol $\Pi^{\mathsf{PKI}}_{\mathsf{PL}}$ for a party P. L_1, L_2 are constants such that $L_1 < L_2$.

- *Initialization.* P runs $\Pi^{\mathsf{Boot}}_{\mathsf{PL}}$, as described so far, until she receives a chain of length at least L_1. We choose L_1 so that it is guaranteed that all security properties hold, and about k new blocks have been inserted in the common-prefix of the chains of all honest players.
- *2-for-1 mining.* Let $\mathcal{C}$ be P's chain at the end of the initialization phase. From now on, she is going to do 2-for-1 PoW mining, and include in her transaction PoWs (i) the hash of the $(L_1 - k)$-th block of $\mathcal{C}$, and (ii) a randomly generated public key for which she has stored the corresponding secret key. A new key must be generated every time she starts mining a new transaction. Whenever P mines a new transaction, she diffuses it to the network, and whenever she receives one, she includes it in the transactions of the block she is mining.

  The first time P receives a chain of length greater or equal to L_2, she runs the Key extraction procedure (below). The phase ends at round $\frac{L_2}{(1-\delta)\gamma}$, where P runs the Termination procedure.
- *Key extraction.* P extracts and stores a set of keys from her chain $\mathcal{C}'$ according to the following rules: She stores any public key which belongs to a transaction that (i) is in the first $L_2 - k$ blocks of $\mathcal{C}'$, and (ii) the hash of the block contained in the transaction matches the hash of the $(L_1 - k)$-th block in her chain.
- *Termination.* P outputs the keys from the key extraction phase and terminates.

Next, we prove that a consistent PKI with an honest majority is generated at the end of the execution of protocol $\Pi^{\mathsf{PKI}}_{\mathsf{PL}}$. Two properties are guaranteed: (1) honest parties output the same set of keys and (2) more than half of these keys have been generated by them. For the rest of this section let α_2, β_2, f_2 be the corresponding values of α, β, f for the difficulty level D_2, e.g. $f_2 = nq\frac{D_2}{2^\kappa}$. Moreover, for the rest of this section, assume that an execution is typical, if in addition to Definition 13 similar lower and upper bounds hold for random variables $X_2(S), Z_2(S)$ that correspond to the number of transaction that honest parties and the adversary compute in a round interval

S, respectively. Note, that such a typical execution again happens with overwhelming probability in κ.

Theorem 34. *Assume a typical execution and Assumption 1. If* $\lceil\log(D_1)\rceil + \lceil\log(D_2)\rceil \leq \kappa$*, then for parameters* $L_1 = 14(l+1)\cdot(1+\delta)f + 2k$*,* $L_2 = L_1 + 5k\cdot(1+\frac{10}{\delta})$*, and* $k \geq \frac{16\kappa f}{\gamma\delta}$*, the following hold for protocol* $\Pi^{\mathtt{PKI}}_{\mathsf{PL}}$*:*

- *All honest players output the same set of public keys;*
- *the majority of the keys are generated by honest parties; and*
- $\Pi^{\mathtt{PKI}}_{\mathsf{PL}}$ *has round complexity linear in* κ*.*

Proof. First, due to the assumption that $\lceil\log(D_1)\rceil + \lceil\log(D_2)\rceil \leq \kappa$ we prove that the events that a party succeeds in mining a PoW of any of the two types in a single query are independent.

Claim 3. *The events that a party succeeds in mining a PoW of any of the two types are independent.*

Proof. Let A,B be the events where a party succeeds in mining a PoW of type 1 and 2 respectively. Let U be the result of the combined mining oracle query. Event A depends on the first $\lceil\log(D_1)\rceil$ bits of U, while event B depends on the last $\lceil\log(D_2)\rceil$ bits. Since each bit of U is sampled independently and uniformly at random, and $\lceil\log(D_1)\rceil + \lceil\log(D_2)\rceil \leq \kappa$ it follows that A,B do not depend on the same random bits, and are thus independent. □

Let chain $\mathcal{C}_1, \mathcal{C}_2$ be the chains that honest party P adopts at rounds r_1, r_2 when she starts the 2-for-1 PoW mining phase and when she extracts the public keys respectively. W.l.o.g, assume that all parties receive chains of length L_1, L_2 respectively, at the same round.

Due to the typicality of the execution, and the fact that $L_2/(1-\delta)\gamma \geq \kappa$, note that all parties will receive a chain of length greater than L_2, before round $\frac{L_2}{(1-\delta)\gamma}$. Hence, the protocol terminates after parties receive a chain of length at least L_2.

We first prove that all honest parties will output the same set of public keys. Due to the common prefix property the chains of all honest parties will be the same up to block $L_2 - k$ at the round they extract the public keys. Hence, the set of public keys they output is exactly the same. Moreover, all honest parties will compute transactions based on the same $L_1 - k$-th block, and thus they will all be valid.

Next, we are going to prove that the majority of these keys have been generated by honest parties. Since $L_1 - 2k \geq 4(l+1)$, it follows that the last $2k$ blocks of $\mathcal{C}_1$ have been mined after round $14(l+1)$, and thus the chain quality property holds for this sequence of blocks. Hence, in the last k blocks of $\mathcal{C}_1^{\lceil k}$ there exists at least one honest block. Moreover, this honest block must have been mined after round $r_1 - \frac{2k}{(1+\delta)f}$, otherwise more than $2k$ blocks would have been mined in less than $2k/(1+\delta)f$ rounds, which contradicts the typicality of the execution. This implies that the adversary cannot have started mining transactions before this round, as otherwise he would be able to predict the output of the hash function, and this event does not happen in a typical execution.

Similarly, it holds that there exists at least one honest block in the last k blocks of $\mathcal{C}_2^{\lceil k}$, and this block must have been mined after round $r_2 - \frac{2k}{(1+\delta)f}$, otherwise P could not have received a chain of length L_2 at round r_2. Hence, by the typicality of the execution, honest parties mine at least $X_{\texttt{trans}} = (r_2 - r_1 - \frac{2k}{(1+\delta)f})(1-\frac{\delta}{8})\alpha_2$ transactions from round r_1 up to round $r_2 - \frac{2k}{(1+\delta)f}$, while the adversary mines at most $Z_{\texttt{trans}} = (r_2 - r_1 + \frac{2k}{(1+\delta)f})(1+\frac{\delta}{9})\beta_2$ transactions from round $r_1 - \frac{2k}{(1+\delta)f}$ up to round r_2. Furthermore, using a similar argument as before, it holds that:

$$r_2 - r_1 + \frac{2k}{(1+\delta)f} \geq \frac{L_2 - L_1 + k}{(1+\delta)f} \Leftrightarrow r_2 - r_1 \geq \frac{2k}{(1+\delta)f}\cdot(1+\frac{10}{\delta}) \tag{3.4}$$

Hence, putting it all together:

$$\begin{aligned}
X_{\texttt{trans}} - Z_{\texttt{trans}} &\geq (r_2 - r_1 - \frac{2k}{(1+\delta)f})(1-\frac{\delta}{8})\alpha_2 - (r_2 - r_1 + \frac{2k}{(1+\delta)f})(1+\frac{\delta}{9})\beta_2 \\
&\geq (r_2 - r_1)((1-\frac{\delta}{8})\alpha_2 - (1+\frac{\delta}{9})\beta_2) - \frac{2k}{(1+\delta)f}((1-\frac{\delta}{8})\alpha_2 + (1+\frac{\delta}{9})\beta_2) \\
&\geq \frac{2k}{(1+\delta)f}\cdot(1+\frac{10}{\delta})((1-\frac{\delta}{8})\alpha_2 - (1+\frac{\delta}{9})\beta_2) - \frac{2k}{(1+\delta)f}((1-\frac{\delta}{8})\alpha_2 + (1+\frac{\delta}{9})\beta_2) \\
&\geq \frac{2k}{(1+\delta)f}\cdot[(1+\frac{10}{\delta})((1-\frac{\delta}{8})\alpha_2 - (1+\frac{\delta}{9})\beta_2) - ((1-\frac{\delta}{8})\alpha_2 + (1+\frac{\delta}{9})\beta_2)] \\
&> \frac{2k}{(1+\delta)f} > 0
\end{aligned}$$

Where the forth inequality is implied by Inequality 3.4, and the $6th$ one by the fact that $\frac{10}{\delta} > \frac{2(1+\frac{\delta}{9})\beta_2}{(1-\frac{\delta}{8})\alpha_2-(1+\frac{\delta}{9})\beta_2}$, which can be derived by the inequalities from Lemma 15. The theorem follows.

□

Note, that our protocol does not guarantee that all honest parties will get the same number of keys, or even just one. This is the reason why we can only tolerate a static adversary; if only a few keys are computed, an adaptive adversary can potentially corrupt all honest parties that hold them. In principle, we can tolerate an *adaptive* adversary followings the ideas found in [PS17]; the difficulty of finding a transaction D_2 is set very low, and parameters L_1, L_2 are changed accordingly to ensure that all parties compute approximately the same number of keys.

Consensus and other applications. Next, we describe how $\Pi_{\mathtt{PL}}^{\mathtt{PKI}}$ can be used in other contexts. First, a direct application of our protocol is in the context of *proof of stake* protocols. In this type of protocols, blocks are mined by randomly selecting stake holders with probability proportional to their stake. A typical requirement for bootstrapping such protocols (e.g. [KRDO17, KN12]), is that in the initial state of the economy the majority of the coins are controlled by honest parties. By assigning one coin to each public key produced by our protocol, we can efficiently and securely bootstrap a proof of stake protocol.

A more general application of $\Pi_{\mathtt{PL}}^{\mathtt{PKI}}$ is in solving consensus (aka Byzantine agreement) [PSL80, LSP82], with no trusted setup, and in a number of rounds independent of the number of parties. If parties submit transactions containing their input instead of public keys, it follows that by taking the majority of their output they are going to achieve Byzantine agreement. That is, everyone will agree on the same value (the Agreement property), and if all honest parties have the same input v, they are all going to output v (Validity). Note, that unlike the case of setting a PKI, we conjecture that this protocol can also tolerate adaptive attacks, as the adversary does not gain anything by corrupting a party that has already computed a transaction.

Finally, our protocol for the establishment of an honest-majority PKI enables the application of traditional Byzantine fault-tolerant techniques for ledger creation and maintenance based on "subcommittees" as opposed to mining (cf. [ACC$^+$17]) to permissionless networks. Instead of having arbitrary membership authorities, these committees can be elected using our protocol with the guarantee of an honest majority. Note that by changing the difficulty of the transaction-level PoW we can force the number of parties in the committee to be in a specific predefined interval.

Reducing the communication cost. While the round complexity of our protocol is independent of the number of parties, this does not hold for its communication cost, measured by the number of transmitted messages. The reason is that in the challenge-exchange phase, all parties have to diffuse their random challenges, thus increasing the communication cost of the protocol by an $O(n)$ factor.

Next, we describe an alternative design for the challenge-exchange phase, for which we *conjecture* that the number of *different* messages diffused[8] by honest parties is in-

[8]Note, that each diffusion requires sending the same message at least $O(n)$ times.

dependent of their number, and only depends on the security parameter and the precomputation time available to the adversary. We do this in the following way: instead of having all parties sent a random challenge in order to be sure that the genesis blocks that are later mined are fresh, we demand that each random challenge be accompanied by a PoW. This way, all honest parties will be sure that at least one honest challenge is generated with high probability every $O(\kappa)$ rounds. Moreover, honest parties will only diffuse random challenges that are tied to a PoW. Thus, the total number of different messages sent will be upper-bounded by the number of PoWs that the adversary and the honest parties combined have generated. Also, again different honest parties will have received the same block with at most one round difference. By combining the above ideas, we can again create a graded-agreement-type procedure for the genesis blocks and in the same way achieve consensus.

Acknowledgments

The author is grateful to Lisa Eckey, Sebastian Faust and Julian Loss, for helpful comments and discussion about this chapter.

Chapter 4

Consensus from Signatures of Work

4.1 Introduction

The consensus problem—reaching agreement distributedly in the presence of faults—has been extensively studied in the literature starting with the seminal work of Shostak, Pease and Lamport [PSL80, LSP82]. The problem formulation has a number of servers (parties) starting with an individual input which should agree at the end to a joint output that has to match the input in the case where all non-faulty servers happened to have the same input value. One of the critical measures of effectiveness for consensus protocols is maximizing their resilience to Byzantine faults, typically denoted by t. It is known that $t < n/2$ is necessary to achieve consensus, where n is the total number of parties, while protocols have been designed that reach that level of resilience assuming synchrony and a way to authenticate messages using digital signatures [DS83][1] (or "pseudosignatures" [PW92]). This result is known to be tight since lack of synchrony would imply $t < n/3$ [DLS88], while lack of a message authentication mechanism has a similar effect [Bor96].

Recently, with the advent of blockchain protocols like Bitcoin, the problem has experienced renewed interest from a much wider community of researchers and has seen its application expand to various novel settings, such as the so-called "permissionless" setting, where participation in the protocol is both unrestricted and unauthenticated. In fact, this setting was initially studied in [Oku05a, Oku05b], where it was shown that deterministic consensus algorithms are impossible for even a single

[1]Recall that the protocol in [DS83] tolerates an arbitrary number of Byzantine faults ($n > t$), but in the version of the problem of a single sender (a.k.a. "Byzantine Generals," or just broadcast); in the case of consensus, $t < n/2$ is necessary regardless of the resources available to the parties in the protocol execution (see, e.g., [Fit03, GK18]).

failure but that probabilistic consensus is still feasible by suitably adapting the protocols of [Ben83, FM97]. Nevertheless, the resulting protocol required exponentially many rounds in n.

The first efficient solutions for the consensus problem in the permissionless setting were formally shown to be possible utilizing an abstraction of the Bitcoin blockchain protocol in [GKL15], against adversaries controlling less than half of the computational power which, in a uniform configuration (meaning parties are endowed with the same computational power), corresponds to a number of Byzantine faults $t < n/2$ in the original setting. At a high level, these protocols (as well as the Bitcoin blockchain protocol itself) rely on a concept known as *proofs of work* (PoW), which, intuitively, enables one party to convince others that he has invested some computational effort for solving a given task. While being formulated a while back [DN93] and used for a variety of purposes—e,g, spam mitigation [DN93], sybil attacks [Dou02a], and denial of service protection [JB99, Bac02]—their role in the design of permissionless blockchain protocols [Nak08], is arguably their most impactful application.

In the context of permissionless blockchain protocols, the way a PoW-like primitive helps is by *slowing down* message generation for all parties indiscriminately, thus generating opportunities for honest parties to converge to a unique view under the assumption that the aggregate computational power of honest parties sufficiently exceeds that of the adversary. Now, while this intuition matches the more rigorous analyses of the Bitcoin protocol that have been carried out so far [GKL15, PSS17, GKL17, BMTZ17], these works have refrained from formally defining such enabling functionality as a stand-alone cryptographic primitive, and relied instead on the random oracle (RO) model [BR93] or similar idealized assumptions (cf. the $\mathcal{F}_{\text{TREE}}$ functionality in [PSS17]) to prove *directly* the properties of the blockchain protocol. The same is true for other provably secure PoW-based distributed protocols [AD15, KMS14, GKLP18].

The core of the hardness (or even impossibility [CGH04]) of implementing the assumed idealized resources is that they satisfy a strong independence property: Each bit output on a new query to the resource is independently sampled, even if the adversary is the one who is accessing the resource. This is indeed a very strong property, as it directly implies that the best way to compute a PoW for both an honest party and the adversary is brute force. Moreover, the same property is explicitly used to argue the security of the proposed consensus protocols in the PoW setting [GKL15, AD15], as we explain in detail later.

In this work we make progress in relaxing this assumption, by putting forth a formalization of a PoW-like primitive, which we call *signatures of work* (SoW). An SoW can be implemented in the RO model or by using $\mathcal{F}_{\text{TREE}}$, but the adversarial SoW computation process does *not* necessarily satisfy such strong guarantees as the ones mentioned above. Indeed, in contrast to previous approaches, only an upper bound on the rate at which the adversary generates SoWs is assumed. We then present a new permissionless consensus protocol based on SoWs that can be proven secure without relying on the strong independence property. The protocol utilizes a SoW-based blockchain and standard properties of the underlying hash function, and is secure assuming an honest majority of computational power. As a result, this protocol can be seen as an exemplar of how a permissionless signature-like primitive enables honest majority consensus in the same way that classical digital signatures imply honest-majority consensus protocols in the traditional setting.

Why *signatures* of work? We first provide some intuition behind the relevance of SoW as a useful primitive for the design of permissionless distributed protocols. Recall the main property of a digital signature in the design of classical consensus protocols: It enables parties to communicate to each other their protocol view and inputs at a certain stage of the protocol execution in a way that is transferable and non-repudiable. Indeed, Bob, upon receiving Alice's signed message, can show it to Charlie in a way that the latter is unequivocally convinced of the message's origin. It follows that Bob cannot modify Alice's messages, playing man-in-the-middle between Alice and Charlie, and thus Alice can be held accountable in case she provides conflicting views to the two parties. A SoW scheme provides a similar capability: Using a SoW, a party like Alice can invest effort into a specific protocol view and inputs, so that when Bob is presented with a SoW produced by Alice it will be infeasible for Alice to provide a conflicting view and inputs to Charlie, *unless she invests twice the effort*. Moreover, the above argument holds without establishing any set of identities among the parties, so for example Bob does not need to know he talks to Alice *per se* but rather to an arbitrary party that invested some effort with respect to a specific protocol view. Furthermore, exactly like digital signatures, SoWs can be chained recursively, enabling the parties to build on each other's protocol view.

While the above functionalities hint to the usefulness of SoWs in the distributed permissionless setting, formalizing and applying them properly is no simple task. Firstly, in contrast with classical signatures, there is no secret key involved in this primitive. This make sense, since in a permisionless setting signing messages using some kind of secret information is meaningless, as parties do no have any secret setup to begin with. Hence, if they are to sign any message, they should use some other kind of resource that only they have access to, such as their computational power. Secondly, in classical signatures, the exact time when the verification key becomes available to different parties is irrelevant; The key is only useful for verification, up to polynomial-time differences. In the context of SoWs, however, this time is of great importance. For example, allowing a party to learn the verification key, say, two days earlier than other parties, means that this party will be able to compute two days worth of signatures more than them. Hence, in contexts where *counting* the number of generated signatures matters, as is the case in blockchain protocols, great care should be taken on guaranteeing that the verification key is "fresh" enough for the relevant application.

Our results. Our contributions are are as follows:

1) Formalization of an SoW scheme. The syntax of an SoW scheme entails four algorithms: Public parameter generation, key generation, signing and verification—$\mathsf{PPub}, \mathsf{KeyGen}$, Sign and Verify, respectively. PPub is invoked on input 1^{λ}, where λ is the security parameter, and outputs public security parameters pp. KeyGen is invoked on input pp, and outputs a random verification key vk. Sign is invoked on input (pp, vk, msg, h), where msg is the message to be signed, and h is the *hardness* level of the signature generation. Expectedly, Verify is invoked on input (pp, vk, msg, h, σ), where σ is (possibly) an output of Sign. We require a SoW scheme to be:

- **Correct:** As in the case of classical signatures, we require that signatures produced by Sign should be accepted by the Verify algorithm.
- (t, α)**-Successful:** This property lower-bounds the probability that an honest signer will successfully produce a SoW in a certain number of steps t; α is a function of the hardness level h.
- t**-Verifiable:** The verifier should be able to verify a SoW in t steps. (Typically, t is a lot smaller than the time need to produce a signature.)
- **Moderately Unforgeable against Tampering and Chosen-Message Attacks** ((β, ε)-**MU-TCMA):** This property is akin to the property of *existential unforgeability under chosen-message attacks* of digital signatures (EU-CMA). It captures the fact that producing a *sequence* of SoWs, for chosen messages, does not provide an advantage to an adversary in terms of running time. Specifically, the chances to produce more than $\beta \cdot t$ SoWs in t steps (for any t) are less than ε.[2] Further, this

[2]Note that, unlike previous unforgeability definitions (e.g, [BR96]), this definition is parameterized by the *rate* β at which the adversary can produce signatures, instead of the number of steps it needs to compute one. We feel that this formulation is more appropriate for the moderate unforgeability game

should hold against an adversary able to *tamper* with the keys, and even in the presence of a Sign oracle.

– **Run-time independent:** This final property captures the setting where honest signers are potentially invoked on adversarial inputs and ensures that their running time enjoys some degree of independence. Specifically, the random variables defined as the running time of each Sign invocation is a set of almost independent random variables (cf. [AGHP92]). We stress that the adversarial signing algorithm may *not* satisfy this property.

As a "sanity check," we show in Section 4.5 that a SoW scheme can be easily designed and proven secure in the random oracle model (or by using $\mathcal{F}_{\text{TREE}}$), and hence in practice can be instantiated by a cryptographic hash function such as SHA-256.

2) Consensus from SoW. Next, we design a consensus protocol for an honest majority of computational power that can be reduced to the SoW primitive above. The core idea behind our new protocol is as follows. First, the parties build a blockchain using SoWs in a way reminiscent of the Bitcoin blockchain. Using SoWs we show how to emulate the Bitcoin backbone protocol [GKL15] by having parties compute a SoW in parallel, "on top" of the current view that incorporates the largest number of SoWs, i.e., the longest chain. However, in contrast with the consensus protocol of [GKL15], to generate a block, the parties include not only their input to the consensus protocol, but also the headers of "orphan" blocks that exist in forks stemming off their main chain and which have not been included so far, where the header of a block contains the hash of the previous block in the chain, the signature, the input to the consensus protocol, and a hash of the block's contents.

Using this mechanism, as shown in Figure 4.1, we prove that it is possible to reconstruct the whole tree of block headers from the blockchain contents, and thus in this way preserve all block headers produced by the honest parties. This ensures that the resulting ledger will reflect the number of parties and hence a consensus protocol may now be easily reduced to this blockchain protocol.

Our new consensus protocol relying on the SoW primitive in the setting where no PKI is available, exemplifies the contrast with consensus in the classical setting, relying on standard signatures and a PKI setup [DS83] (cf. [GK18]). It is worth noting that the only known blockchain-based provably secure and optimally resilient consensus protocol is given in [GKL15], using a technique called "2-for-1 PoW" where two PoW-based protocols can be run concurrently and create a blockchain where the number of

where the adversary tries to produce multiple signatures. For further details, see Definition 38.

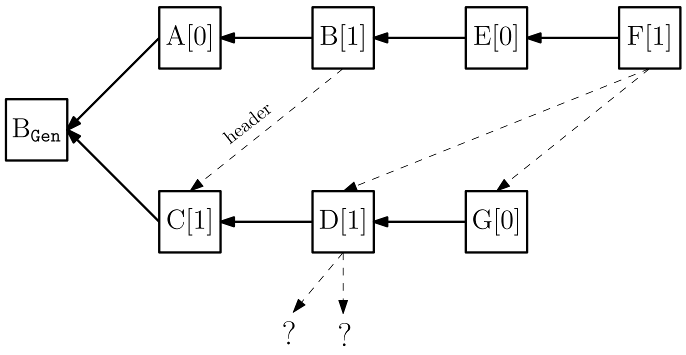


Figure 4.1: *The data structure maintained by the consensus protocol. Block F has consensus input* 1*, and includes the headers of blocks D and G, with input* 1 *and* 0*, respectively. Block D includes the headers of invalid blocks. This is not a problem, since any chain that contains D will be invalid and not selected by any party, while D's consensus input is correctly counted as a valid block header.*

honest-party contributions is proportional to their actual number, but which relies on the strong independence property of the RO model, discussed earlier, in a fundamental way. Indeed, in the RO model, each witness for a PoW can be rearranged in a certain way so as to obtain a test for a witness for *another* PoW in a way that is independent from the first solution. Our new protocol gets rid of this need. The only other (non-blockchain) PoW-based consensus protocol [AD15] also relies on the RO model.

As intermediate steps in our analysis, we first introduce an appropriate adaptation of the model of [GKL15] that allows for a standard model analysis and which may be of independent interest. We then recall the three basic properties of the blockchain data structure presented in [GKL15]: (strong) common prefix, chain quality and chain growth, and show how our SoW-based blockchain protocol satisfies them assuming, beyond the security of SoW, standard collision resistance from the underlying hash function that is used to "glue" the blocks together. This is achieved as follows: We first prove that using the MU-TCMA property and assuming the adversarial hashing power is suitably bounded, it is unlikely in any sufficiently long time window for the adversary to exceed the number of SoWs of the honest parties. Then, using the (t,α)-Successful and (β,ε)-MU-TCMA properties in conjunction with run-time independence, we establish that summations of running times of successive Sign invocations have the *variance* needed to ensure that "uniquely successful rounds" (i.e., rounds where exactly one of the honest parties produces a SoW) happen with high density in any sufficiently long time window. Using these last two core results, and under suitable constraints for the basic SoW parameters $\alpha, \beta, \varepsilon, h$ and number of parties n, we prove that the security of the Bitcoin backbone protocol implements a robust transaction ledger [GKL15]. Further, and as a sanity check, in Section 4.5, we argue that the results we get from our black-box analysis (and the RO-based SoW construction mentioned earlier), are similar to those from the random-oracle analysis of [GKL15].

Prior and related work. We have already mentioned above relevant related work regarding classical and blockchain-based consensus protocols. For a more exhaustive recent survey, refer to [GK18]. We also note that the focus of this chapter is the original consensus problem [PSL80, LSP82], and not so-called "ledger consensus" (sometimes referred to as "Nakamoto consensus"), which is an instance of the state machine replication problem [Sch90]; see also [CBPS10] for an overview of such protocols. The idea of referencing off-chain blocks has been considered early on in the ledger consensus literature (see, e.g., [LSZ15, SZ15, SLZ16, BHMN17]) as a way to obtain fairness, better throughput and faster confirmation times. Our novelty is that we leverage this technique along with the new SoW notion to build a provably secure consensus protocol, which, unlike prior results, is not based on the "2-for-1 PoW" technique described earlier.

There have been a number of attempts to formalize a proof of work (PoW) primitive that it is also sufficient to imply the security of a blockchain protocol. Nevertheless, such works were either informal [BCD$^+$14, Poe], or they did not produce a correctness proof for a blockchain or consensus protocol, focusing instead on other applications [BGJ$^+$16, AT17, BBBF18, BRSV18, GKP19]. We proceed to give a partial list of such considerations.

Dwork and Naor [DN93] first considered PoWs under the term "pricing functions," as a means of protection against spam e-mail. The main properties discussed in their work are amortization resistance, "moderate hardness" and the existence of trapdoors ("shortcuts" in their terms). Interestingly, among the three constructions described there, one of them is a partially "broken" signature scheme, i.e., while it is hard for an attacker to obtain the signing key, it is moderately hard to forge signatures.

In a different direction, Juels and Jacobsson [JJ99] and Back [Bac00, Bac02] use PoWs to construct electronic payment systems. In [JJ99] the authors consider the following properties: Amortization resistance, fast verification, and some special composability property which states that generating a PoW for some scheme may help in generating a PoW for another scheme. As acknowledged by the authors themselves, the definitions they provide are only sketches. In [Bac00, Bac02] another set of closely related properties is considered, including amortization and "trapdoor-freeness." Amortization refers to the ability to combine PoWs of lesser difficulty in order to construct PoWs of greater difficulty. "Trapdoor-freeness" dictates that the party which generates the initial parameters of the scheme should not be able to also generate a trapdoor regarding these parameters. Again the approach is not rigorous.

Stebila et al. in [SKR$^+$11], present PoW security definitions that capture amortization resistance, both in an interactive and in a non-interactive manner. In addition, they show that the Hashcash scheme satisfies their definition in the ROM. In contrast to out work, the focus of the paper is Denial of Service prevention in a client-server setting, and thus lacks many of the security properties we develop here, e.g., "honest prover" properties, adversarial precomputation in the hardness experiment.

Bitansky *et al.* [BGJ$^+$16] construct time-lock puzzles as well as PoW schemes from randomized encodings. Since the focus of their work is time-lock puzzles, the properties of PoW schemes—amortization resistance, moderate hardness and fast verification—are only briefly investigated, although they do instantiate a PoW scheme based on randomized encodings and the existence of non-amortizing languages in the worst case.

Another interesting approach is that of Ball *et al.* [BRSV18], who construct a PoW scheme from worst case assumptions. They base the security of their PoW on the worst case hardness of classical complexity theory problems, e.g., the Orthogonal Vectors problem. As the authors state, their focus is on PoW as a foundational cryptographic primitive, and not on its application on blockchain protocols. In any case, the PoW schemes they instantiate have non-negligible error, and thus cannot be directly used our context.

In [AT17], Alwen and Tackmann study moderately hard functions (MoHF), providing simulation based definitions for what they call "non-interactive proofs of effort" (niPoE), which—as explicitly acknowledged by the authors—cannot be used to analyze Bitcoin. The main impediment is that the adversary can only invoke the same MoHF only once per protocol session, while for the Bitcoin protocol multiple invocations of the same MoHF should be allowed.

In [BCD$^+$14] and [Poe], the concept of "dynamic membership multi-party signatures" (DMMS) is proposed to describe the underlying primitive used in Bitcoin. The author of the latter work also argues the security of Bitcoin, based on a DMMS scheme. While the properties described there bear some similarities to our work, the treatment is not formal.

More recently, an effort to formalize an intermediate PoW-like building block for the Bitcoin protocol was made in [PSS17]. The proposed ideal functionality, $\mathcal{F}^{p}_{\text{TREE}}$, keeps track of a tree of messages, which both the honest parties and the adversary can extend with probability p. The outcome of each such trial is independent of the others, even if it is made by the adversary. $\mathcal{F}^{p}_{\text{TREE}}$ satisfies the strong independence property mentioned before, and hence it is not suitable for the goals of this chapter. Moreover, we note that any protocol instantiating this functionality must necessarily be *interactive*, as two parties can use $\mathcal{F}^{p}_{\text{TREE}}$ to communicate at least one bit. Finally, in [PSS17], it was shown how to implement a transaction ledger, but not how to achieve consensus; the techniques introduced in this work can be adapted to implement a consensus protocol using $\mathcal{F}_{\text{TREE}}$.

Limitations and directions for future research. Our analysis is carried out in the synchronous setting. It is relatively straightforward to extend our results to the Δ-synchronous setting of [PSS17], by using the same techniques as in [GKL14] (Section 7). We leave as an open question extending our results to the variable difficulty setting of [GKL17]. The construction of an SoW based on better studied assumptions, e.g., the discrete logarithm or the RSA problem, or the collision resistance property of a hash function, or the problems presented in [BRSV18], is also an exciting direction for future research.

Organization of the paper. The basic computational model and definitions used by our constructions are presented in Section 4.2. Formal definition of the SoW primitive and its security properties are presented in Section 4.3. Section 4.4 is dedicated to applications of SoW: First, we introduce an appropriate model for our applications (Section 4.4.1). We then analyze the Bitcoin backbone protocol based on (and reducing its security to) SoW (Section 4.4.2), followed by the new blockchain-based consensus protocol (Section 4.4.3). Further, and as a “sanity check,” in Section 4.5 we show how to implement SoW in the RO model, or using $\mathcal{F}_{\text{TREE}}$.

4.2 Preliminaries

In this chapter we will follow a more concrete (“exact”) approach [BR96, BDJR97, GMPY11] to security evaluation rather than the asymptotic one. We will use functions t, ε, whose ranges are $\mathbb{N}, \mathbb{R}$, 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 fixed computational model, respectively. When we speak about running time this will include the execution time plus the length of the code (cf. [BL13]; 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.* [GMPY11]; refer to [GMPY11] for details on using interactive RAMs in a UC-like framework, as well as to Section 4.4.1. Given a RAM M, we will denote by $\mathsf{Steps}_M(1^\lambda, x)$ the random variable that corresponds to the number of steps of M given as input the security parameter 1^λ and x. We will say that M is t-bounded if it holds that $\Pr[\mathsf{Steps}_M(1^\lambda, x) \leq t(\lambda)] = 1$.

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

4.3 Signatures of Work

The main goal of this chapter is to implement consensus in the permissionless setting without relying on the strong independence property of the underlying computational resource. Towards that goal, in this section we introduce the *signature of work* (SoW) primitive. At a high level, a SoW enables one party to convince others that she has invested some computational power during some specific time interval and with respect to a "message." Next, we formalize this notion and present its desired security properties.

SoW syntax. Given a security parameter λ, let PP be the public parameter space, $HP \subseteq \mathbb{N}$ the hardness parameter space, K the key space, M the message space, and S the signature space. With foresight, the role of the key is to provide "freshness" for the signature computation, thus certifying that the signature was computed in the given time interval.

Definition 35. A SoW scheme consists of four algorithms $\mathsf{SoW} = (\mathsf{PPub}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where:

– $\mathsf{PPub}(1^\lambda)$ is a randomized algorithm that takes as input the security parameter λ, and returns a set of public parameters $pp \in PP$.

- $\mathsf{KeyGen}(pp)$ is a randomized algorithm that takes as input the public parameters pp, and returns a key $vk \in K$. (See Remark 3 below on the role of keys in SoW schemes.)
- $\mathsf{Sign}(pp, vk, msg, h)$ is a randomized algorithm that takes as input public parameters $pp \in PP$, a key $vk \in K$, a message $msg \in M$ and hardness parameter $h \in HP$, and returns a signature (of work) $\sigma \in S$.
- $\mathsf{Verify}(pp, vk, msg, h, \sigma)$ is a deterministic algorithm that takes as input public parameters $pp \in PP$, a key $vk \in K$, message $msg \in M$, hardness parameter $h \in HP$ and a signature $\sigma \in S$, and returns true or false to indicate the validity of the signature.

Remark 3. SoW schemes only have a public verification key. The role of this key is to guarantee that the computational work spent in order to create a signature of work is "fresh," i.e., executed during a specific time interval (say, from the time the key became known to the signer). In contrast, classical digital signatures also have a secret key that serves as a trapdoor to compute signatures. In the applications we consider, the existence of trapdoor information is not meaningful, and in fact may hurt the security of the respective constructions.

Security properties. Next, we present a number of security properties that we will require SoW schemes to satisfy. We start with the correctness property.

Definition 36. We say that a SoW scheme is *correct* if for every $\lambda \in \mathbb{N}, pp \in PP, vk \in K, h \in HP$, and $msg \in M$:

$$\Pr\Big[\mathsf{Verify}(pp, vk, msg, h, \mathsf{Sign}(pp, vk, msg, h)) = \mathsf{true}\Big] \geq 1 - \mathsf{negl}(\lambda).$$

Next, we require that the time to verify a signature be upper bounded.

Definition 37. We say that a SoW scheme is *t-verifiable*, if Verify takes time at most t (on all inputs).

Next, we capture the case of a malicious signer (resp., verifier) in the context of SoWs. In the first case, the adversary's objective is to compute a number of signatures a lot faster than an honest signer would, while in the second case it is to make the honest signer take too much time to generate a signature.

We deal with malicious signers first. We put forth an attack that we will use to express a class of adversaries that attempt to forge signatures faster than expected. Intuitively, this constitutes an attack against an honest verifier that may be trying to gauge a certain measure using the number of signatures. The game defining the attack is shown in Figure 4.2; we call the corresponding security property *Moderate Unforgeability against Tampering and Chosen Message Attack* (MU-TCMA). As in the security definitions of standard signatures (e.g., EU-CMA), we allow the adversary to have access to a signing oracle $\mathcal{S}$. Every time the oracle is queried, we assume that it runs the Sign procedure with uniformly sampled randomness. A subtle point in the modeling of security in the presence of such oracle is that $\mathcal{S}$ should also "leak" the number of steps it took for a query to be processed. In an actual execution while interacting with honest parties that are producing signatures, time is a side channel that may influence the adversarial strategy; in order to preserve the dependency on this side channel we will require from $\mathcal{S}$ to leak this information. We note that in the classical signatures literature, timing attacks have also been a serious consideration [Koc96].

In addition, we require that the key used by the adversary to construct signatures be fresh, i.e., we want to avoid situations where the adversary outputs signatures that he has precomputed a long time ago. We model this by providing the fresh key after the adversary has finished running his precomputation phase. Further, we allow the adversary to tamper with the key by manipulating it via tampering functions belonging to a family of functions $\mathcal{F}$.

Looking ahead, the tampering function in our applications will be related to a keyed hash function, where the key of the hash is part of a *common random string* (CRS). Hence, we choose to model functions in $\mathcal{F}$ to have two inputs: Σ (the CRS) and vk. Moreover, the output of the adversary is deemed invalid if he tampers vk with functions f_1, f_2 in such a way that $f_1(\Sigma, vk) = f_2(\Sigma, vk)$. Otherwise, the adversary could launch a generic attack that is unrelated to the SoW scheme, and produce signatures at twice the rate of an honest signer, as follows. The adversary first finds f_1, f_2 that have this property, and then computes signatures using the tampered key $f_1(\Sigma, vk)$. The trick is that each of them will also correspond to a signature with key $f_2(\Sigma, vk)$. Hence, he effectively can double the rate at which he produces signatures.

Formally, the adversary will have access to $\mathcal{S}(\cdot,\cdot)$, an SoW oracle that on input (vk', msg), where $vk' \in K$ and $msg \in M$, returns the pair (σ, t) where σ is the output of $\mathsf{Sign}(pp, vk', msg, h)$ and t is the number of steps taken by the Sign algorithm on these parameters. Function $\mathsf{Asked}(vk', msg, \sigma)$ is true if σ was the response of $\mathcal{S}$ to some query (vk', msg).

We are now ready to formulate the security property of *Moderate Unforgeability against Tampering and Chosen Message Attacks* (MU-TCMA). It has two parameters,

```
$\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A},\mathcal{F}}(1^\lambda, h, \ell)$
$\Sigma \leftarrow U_\lambda$; $pp \leftarrow \mathsf{PPub}(1^\lambda)$;                    *(Public parameters)*
$st \leftarrow \mathcal{A}_1(1^\lambda, \Sigma, pp)$;                                *(Precomputation)*
$vk \leftarrow \mathsf{KeyGen}(pp)$;                                       *(Verification key)*
$(f_i, msg_i, \sigma_i)_{i\in[\ell]} \leftarrow \mathcal{A}_2^{\mathcal{S}(\cdot,\cdot)}(1^\lambda, vk, st)$;                 *(SoW computation)*
return $\bigwedge_{i=1}^{\ell} \begin{pmatrix} \mathsf{Verify}(pp, f_i(\Sigma, vk), msg_i, \sigma_i) \wedge \neg\mathsf{Asked}(f_i(\Sigma, vk), msg_i, \sigma_i) \\ \wedge (f_i \in F_\lambda) \wedge (\forall j \in [\ell] : f_i(\Sigma, vk) = f_j(\Sigma, vk) \Rightarrow i = j) \end{pmatrix}$
```

Figure 4.2: *The Moderate Unforgeability against Tampering and Chosen-Message Attack (MU-TCMA) experiment for a SoW scheme.*

β and ε, and, informally, it states that no adversary $\mathcal{A}$ exists in the experiment of Figure 4.2 that takes at most t steps after receiving key vk and produces $\ell \geq \beta \cdot t$ signatures with probability better than ε. Note that in total we allow $\mathcal{A}$ to take any polynomial number of steps, i.e., the adversary is allowed to execute a precomputation stage that permits it to obtain an arbitrary number of signatures before learning vk. In the definition below, we allow β to depend on the hardness level h, and ε on h, t and $q_{\mathcal{S}}$, the number of queries the adversary makes to the signing oracle.

Definition 38. Let $\mathcal{F} = \{F_\lambda\}_{\lambda\in\mathbb{N}}$, where F_λ is a family of functions $f : \{0,1\}^\lambda \times K \to K$.[3] A SoW scheme is (β, ε)-*Moderately Unforgeable against Tampering and Chosen-Message Attacks* (MU-TCMA) with respect to tampering function class $\mathcal{F}$, if for any polynomially large t_1, t_2, any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ is t_1-bounded and $\mathcal{A}_2$ is t_2-bounded and makes at most $q_{\mathcal{S}}$ queries to oracle $\mathcal{S}$, for any $\lambda \in \mathbb{N}$, and any $h \in HP$, the probability of $\mathcal{A}$ winning in $\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A},\mathcal{F}}(1^\lambda, h, \lfloor \beta(h) \cdot t_2 \rfloor)$ (Figure 4.2) is less than $\varepsilon(h, t_2, q_{\mathcal{S}})$.

Remark 4. As mentioned in Section 4.1, unlike previous unforgeability definitions (e.g, [BR96]), Definition 38 is parameterized by the *rate* at which the adversary can produce signatures, instead of the number of steps it needs to compute one, which is more appropriate for the moderate unforgeability game where the adversary tries to produce multiple signatures.

In the MU-TCMA definition we are going to consider tampering functions classes that at the very least preserve the unpredictability of vk. Otherwise, the adversary can generically attack any SoW scheme by predicting the tampered key and precomputing

[3] K is the key space of the SoW scheme.

signatures. Formally, we will say that $\mathcal{F}$ is *computationally unpredictable* if the adversary, given the CRS Σ, cannot guess a value y that he will be able to "hit" when he gains access to vk through some $f \in \mathcal{F}$.

Definition 39. Let $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$, where F_λ is a family of functions $f : \{0,1\}^\lambda \times K \to K$. We say that $\mathcal{F}$ is *computationally unpredictable* with respect to a SoW scheme SoW, if for any PPT RAM $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and for any $\lambda \in \mathbb{N}$, it holds that:

$$\Pr_{\substack{pp \leftarrow \mathsf{PPub}(1^\lambda);\\ vk \leftarrow \mathsf{KeyGen}(pp);\\ \Sigma \leftarrow \mathcal{U}_\lambda}} \begin{bmatrix} (st, y) \leftarrow \mathcal{A}_1(1^\lambda, \Sigma, pp); f \leftarrow \mathcal{A}_2(1^\lambda, st, vk) : \\ f \in F_\lambda \wedge f(\Sigma, vk) = y \end{bmatrix} \leq \mathsf{negl}(\lambda).$$

Next, we consider the case of attacking an honest signer. Attacking an honest signer amounts to finding a certain set of keys over which the honest signer algorithm fails to produce SoWs sufficiently fast and regularly. We say that a SoW scheme is (t, α)-*successful* when the probability that the signer computes a signature in t steps is at least α.

Definition 40. We say that SoW scheme is (t, α)-*successful* if for any $\lambda \in \mathbb{N}$ and any $h \in HP$, $pp \in PP, vk \in K, msg \in M$, it holds that:

$$\Pr\left[\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, msg, h) \leq t\right] \geq \alpha(h).$$

Finally, in the same corrupt-verifier setting, we will require the signing time of honest signers to have some (limited) *independence*, which will be important for the applications we have in mind. This property, in combination with the efficiency and MU-TCMA properties, will prove crucial in ensuring that when multiple signers work together, the distribution of the number of them who succeed in producing a signature has some "good" variance and concentration properties.

Definition 41. We say that a SoW scheme has *almost-independent runtime* iff for any polynomial $p(\cdot)$, any $\lambda \in \mathbb{N}$, any $h \in HP$, there exists a set of identical and independently distributed random variables $\{Y_i\}_{i \in [p(\lambda)]}$ such that for any $pp \in PP, ((vk_i, m_i))_{i \in [p(\lambda)]} \in (K \times M)^{p(\lambda)}$ it holds that $\Delta[(\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_i, m_i, h))_i, (Y_i)_i] \leq \mathsf{negl}(\lambda)$.

Independence assumptions. As mentioned earlier, MU-TCMA does not enforce any independence assumption, and only bounds the probability that the rate at which the adversary computes SoWs is high. In contrast, the independent-runtime property does so, but only for honest signers. We remark that achieving such property is considerably easier for the honest case, as we can be sure that signers will use independently sampled coins if instructed; a guarantee that we cannot have for the adversary.

Parameters' range. Let SoW be a scheme that is $(t_{\mathsf{sign}}, \alpha)$-Successful. SoW trivially satisfies the MU-TCMA property for $\beta(h) > 1$, since the adversary does not have enough time to output the signatures it has computed. On the other hand, assuming $\varepsilon(h,t,q_S)$ is a negligible function of t, $\alpha(h)$ must be smaller than $\beta(h) \cdot t_{\mathsf{sign}}$, otherwise the expected number of SoWs computed by the Sign function would exceed that allowed by the MU-TCMA property. Hence, for optimal security, it should hold that $\alpha(h)$ is close to $\beta(h) \cdot t_{\mathsf{sign}}$.

Next, we turn to applications of our SoW primitive.

4.4 Applications of Signatures of Work

In this section we showcase applications of SoWs, the first one being implementing robust transaction ledgers: Using our primitive and standard properties of the underlying hash function, we establish the security of the Bitcoin backbone protocol [GKL15]. The second application is realizing consensus in the permissionless setting.

In both applications, we assume the existence of a SoW scheme with the security properties defined bellow.

Assumption 2 (SoW Assumption). For parameters $\beta, \varepsilon, t'_{\mathcal{H}}, \alpha$ and t_{ver} we assume that $\mathsf{SoW} = (\mathsf{PPub}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is:

- Correct;
- (β, ε)-MU-TCMA with respect to any computationally unpredictable tampering function class (cf. Definition 39);
- $(t'_{\mathcal{H}}, \alpha)$-successful;[4]
- almost run-time independent; and
- t_{ver}-verifiable,

where $\varepsilon(h,t,q_S) \in \mathsf{negl}(\beta(h) \cdot t)$. Moreover, we assume that the parameter spaces K, M, S of the scheme are equal to $\{0,1\}^{\log|K|}, \{0,1\}^*, \{0,1\}^{\log|S|}$, respectively.

We chose K, M, S as above due to the underlying hash-chain structure of the blockchain: The hash of each block acts as a key of the SoW scheme, thus the range $\{0,1\}^{\lambda}$ of the

[4]Parameter $t'_{\mathcal{H}}$ corresponds to a lower bound on the running time of honest parties in our protocol that we introduce in detail later.

hash function should match the key space of the SoW. It is conceivable that potentially different sets than $\{0,1\}^\lambda$ can be used, e.g, $\mathbb{Z}_n^*$, if the SoW key is an RSA group element. The hash function we are going to use in our protocol is also related to the selected SoW scheme by the fact that it is the basis of the computationally unpredictable key tampering function family (see Definition 39) available to the adversary.

We start with some pertinent details about the model that the two applications mentioned above will be analyzed under.

4.4.1 The Permissionless Model, Revisited

All security models proposed for the analysis of PoW-based blockchain protocols [GKL15, PSS17, BMTZ17] rely on bounding the number of queries to an idealized functionality to model limited computational resources. In contrast, we do not wish to restrict the way the adversary accesses the computational resource, and thus we model limited computational resources in a more general manner, i.e., by limiting the exact number of steps parties take. Next, we present a revised version of the model of Section 2.2 that captures our considerations.

For the reasons explained in Section 4.2, we substitute IRAMs for ITMs in the computational model of Section 2.2. Everything else, except the part about the existence of a q-bounded RO, is exactly the same, i.e., communication, synchrony, precomputation, and trusted setup modeling.

Next, we consider the complications in the modeling due to the analysis of Bitcoin in the concrete security setting. Both in [GKL15] and [PSS17] a modified version of the standard simulation-based paradigm of [Can00] is followed, where there exist both a malicious environment and a malicious adversary. In addition, the SoW scheme (called PoW in [GKL15, PSS17]) 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 SoW scheme 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 signatures, 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 [Can00], which we build on. Lastly, another issue arising is that if the adversary is able to send, say, θ messages in each round, it can force each honest party to take $\theta \cdot t_{\mathsf{ver}}$ extra steps per round. If we don't bound θ, then the adversary will be able to launch a DOS attack and make honest parties spend all their resources.[5]

In order to address 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 42. 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 $t_{\mathcal{A}}$;[6]
- the adversary sends at most θ messages per round.

Definition 43. 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 $Q(\textsc{view}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}})$ is false and the execution is $(t_{\mathcal{A}}, \theta)$-good is negligible in λ.

Finally, as per Section 2.2, we assume that we are in a setting with a fresh CRS and against a static adversary.

4.4.2 Public Transaction Ledger

Next, we take a reduction approach to the underlying cryptographic primitive—SoW, as defined in Section 4.3—to prove the security of the Bitcoin backbone protocol [GKL15]. We start with a description of the protocol based on SoW, and then continue with the security proof.

The Bitcoin Backbone protocol. The Bitcoin backbone protocol [GKL15], parameterized by functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$, is an abstraction of the Bitcoin protocol. First, we introduce some notation needed to understand the description of the algorithms, and then cast the protocol making use of our SoW primitive.

We will use the terms *block* and *chain* to refer to tuples of the form $\langle s, x, \sigma \rangle$ and sequences of such tuples, respectively. The rightmost (resp. leftmost) block of chain

[5]This problem is extensively discussed in [AD15], Section 3.4.

[6]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.

$\mathcal{C}$ is denoted by $\text{head}(\mathcal{C})$ (resp. $\text{tail}(\mathcal{C})$). Each block contains a seed, data, and a signature denoted by s, x, σ, respectively. As mentioned, all parties have access to a CRS at the beginning of the execution that contains: the public parameter pp of the SoW scheme, a verification key vk generated by $\mathsf{KeyGen}(pp)$, and the key k of the hash functions H, G used later. We will refer to $\langle 0^\lambda, pp||vk||k, 0^\lambda \rangle$ as the *genesis* block $B_{\texttt{Gen}}$. A chain $\mathcal{C} = B_1 \ldots B_m$ is *valid* with respect to the CRS if and only if (i) B_1 is the genesis block, (ii) for any two consecutive blocks $\langle s_i, x_i, \sigma_i \rangle, \langle s_{i+1}, x_{i+1}, \sigma_{i+1} \rangle$ it holds that $H_k(s_i||G_k(x_i)||\sigma_i) = s_{i+1}$, (iii) each block, besides B_{Gen}, contains a valid SoW, i.e., $\mathsf{Verify}(pp, s_i, x_i, \sigma_i) = \mathsf{true}$, and (iv) the content validation predicate $\mathrm{V}(\langle x_1, \ldots, x_m \rangle)$ outputs true. We call $H_k(s_i||G_k(x_i)||\sigma_i)$ the *hash of block B_i* and denote it by $H_k(B_i)$. Moreover, we define $H(\mathcal{C})$ to be equal to the hash of the head of chain $\mathcal{C}$.

At each round, each party chooses the longest valid chain amongst the ones it has received and tries to extend it by computing (mining) another valid block. If it succeeds, it diffuses the new block to the network. In more detail, each party will run the Sign procedure, with the message parameter being determined by the input contribution function $\mathrm{I}(\cdot)$, and the key parameter being the hash of the last block. We assume that the hardness parameter h is fixed for all executions. 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. Each honest party runs for at most $t_{\mathcal{H}}$ steps per round.

Algorithm 7 The *signature of work* function, parameterized by pp, h and hash functions H, G. The input is $(x, \mathcal{C})$.

```
1: function sow($x, \mathcal{C}$)
2:     $s \leftarrow H(\text{head}(\mathcal{C}))$
3:     $\sigma \leftarrow \mathsf{Sign}(pp, s, x, h)$          ▷ Run the prover of the SoW scheme.
4:     $B \leftarrow \varepsilon$
5:     if $\sigma \neq \perp$ then
6:         $B \leftarrow \langle s, x, \sigma \rangle$
7:     $\mathcal{C} \leftarrow \mathcal{C}B$                                  ▷ Extend chain
8:     return $\mathcal{C}$
```

We summarize the modifications with respect to the original [GKL15] protocol: In Algorithm 6 (signature of work function) the Sign function of the underlying SoW

Algorithm 8 The *chain validation predicate*, parameterized by $pp, h, B_{\texttt{Gen}}$, the hash functions H, G, and the *input validation predicate* $V(\cdot)$. The input is $\mathcal{C}$.

```
1:  function validate($\mathcal{C}$)
2:      $b \leftarrow \mathrm{V}(\mathbf{x}_{\mathcal{C}}) \wedge (\mathrm{tail}(\mathcal{C}) = B_{\texttt{Gen}})$      ▷ $\mathbf{x}_{\mathcal{C}}$ describes the contents of chain $\mathcal{C}$.
3:      if $b = \mathrm{True}$ then      ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
4:          $s' \leftarrow H(\mathrm{head}(\mathcal{C}))$
5:          while $(\mathcal{C} \neq B_{\texttt{Gen}}) \wedge (b = \mathsf{true})$ do
6:              $\langle s, x, \sigma \rangle \leftarrow \mathrm{head}(\mathcal{C})$
7:              if $(\mathsf{Verify}(pp, s, x, h, \sigma) = \mathsf{true}) \wedge (H(\mathrm{head}(\mathcal{C})) = s')$ then
8:                  $s' \leftarrow s$      ▷ Retain hash value
9:                  $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$      ▷ Remove the head from $\mathcal{C}$
10:             else
11:                 $b \leftarrow \mathrm{False}$
12:     return $b$
```

scheme is invoked for a limited number of steps so that the total number of steps of the invoking party does not exceed the $t_{\mathcal{H}}$ bound per round; in Algorithm 4 (chain validation predicate) the Verify predicate is replaced with a call to the Verify algorithm of the SoW scheme; and in Algorithm 1 (backbone protocol) we assume that the honest parties start the execution with a "genesis" block. We leave Algorithm 5 intact.

In order to turn the backbone protocol into a protocol realizing a public transaction ledger suitable definitions were given for functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ in [GKL15], also presented in Table 3.2. We change these definitions slightly as shown in Table 4.1, to ensure two things: Firstly, that the data contained in the hash chain is encoded with a suffix-free code; this is important to ensure that no collisions occur [BJL17] as we show later. And, secondly, to ensure that any block created by an honest party contains sufficient entropy, thus the adversary will not be able to use blocks that it has precomputed to extend this block. We call the resulting protocol $\Pi_{\mathsf{PL}}^{\texttt{SoW}}$.

Security proof. We now prove that $\Pi_{\mathsf{PL}}^{\texttt{SoW}}$ implements a robust public transaction ledger assuming the underlying SoW scheme satisfies Assumption 2 for appropriate parameters related to the running time of honest parties and the adversary. First, we formalize this relation.

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 Sign and Verify calls. By care-

Algorithm 9 The Bitcoin backbone protocol, parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$.

```
1:  $\mathcal{C} \leftarrow B_{\mathsf{Gen}}$                                    ▷ Initialize $\mathcal{C}$ to the genesis block.
2:  $st \leftarrow \varepsilon$
3:  $round \leftarrow 0$
4:  while TRUE do
5:      $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{any chain } \mathcal{C}' \text{ found in } \textsc{Receive}())$
6:      $\langle st, x \rangle \leftarrow \mathrm{I}(st, \tilde{\mathcal{C}}, round, \textsc{Input}(), \textsc{Receive}())$     ▷ Determine the $x$-value.
7:      $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{sow}(x, \tilde{\mathcal{C}})$
8:      if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ then
9:          $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
10:         DIFFUSE($\mathcal{C}$)
11:     $round \leftarrow round + 1$
12:     if INPUT() contains READ then
13:         write $\mathrm{R}(\mathbf{x}_{\mathcal{C}})$ to OUTPUT()
```

fully analyzing the backbone protocol one can extract an upper bound on this value.[7] To aid our presentation, we will use $t'_{\mathcal{A}}$ and $t'_{\mathcal{H}}$ to denote: (i) the time needed by a RAM machine to simulate one round in the execution of the Bitcoin protocol, without taking into account calls made to the Sign routine by the honest parties, and (ii) the minimum number of steps that an honest party takes running the Sign routine per round, respectively.

$$t'_{\mathcal{A}} = t_{\mathcal{A}} + n \cdot t_{\mathsf{bb}} + \theta t_{\mathsf{ver}} \quad \text{and} \quad t'_{\mathcal{H}} = t_{\mathcal{H}} - t_{\mathsf{bb}} - \theta t_{\mathsf{ver}}$$

It holds that at least $n - t$ (non-corrupted) parties will run the Sign routine for at least $t'_{\mathcal{H}}$ steps at every round.

In previous works [GKL15, GKLP18, PSS17], the security assumptions regarding the computational power of the parties participating in the protocol were twofold: (1) The total running time of honest parties per round should exceed that of the adversary, and (2) the rate at which parties produce blocks at each round should be bounded. More realistically, in our approach the running time of the adversary and the running time of honest parties do not have the same *quality*, as the adversary may use a superior

[7]Note that t_{bb} depends on the running time of three external functions: $\mathrm{V}(\cdot), \mathrm{I}(\cdot)$ and $\mathrm{R}(\cdot)$. 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 10 The function that finds the "best" chain, parameterized by function $\max(\cdot)$. The input is $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$.

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

Content validation predicate $\mathrm{V}(\cdot)$	$\mathrm{V}(\cdot)$ is true if its input $\langle x_1, \ldots, x_m \rangle$ is a valid ledger, i.e., it is in $\mathcal{L}$, and each x_i starts with a neutral transaction of the form $r\|\|i$, where r is a string of length $\log\|K\|$ and i is the "height" of the respective block.
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 $\mathsf{KeyGen}(pp)\|\|\|\mathcal{C}\|$.

Table 4.1: *The instantiation of functions* $\mathrm{I}(\cdot), \mathrm{V}(\cdot), \mathrm{R}(\cdot)$ *for protocol* $\Pi_{\mathsf{PL}}^{\mathsf{SoW}}$.

signing algorithm. To take this into account, we additionally need to assume that the efficiency of the adversarial signing algorithm, i.e., β, is close to that of the honest parties. Finally, note that if SoW is close to optimal, i.e., $\alpha(h) \approx \beta t'_{\mathcal{H}}$, and the block generation rate is a lot less than 1, our assumption holds as long as the honest parties control the majority of the computational power.

We now state the computational power assumption formally. The second and the third conditions are similar to the ones already found in previous works, while the first one is the new condition we introduce regarding the underlying computational primitive.

Assumption 3. There exist $\delta_{\mathsf{SoW}}, \delta_{\mathsf{Steps}}, \delta \in (0,1)$, such that for sufficiently large $\lambda \in \mathbb{N}$, there exists an $h \in HP$, such that:

– $\alpha(h) \geq (1 - \delta_{\mathsf{SoW}})\beta(h)t'_{\mathcal{H}} > \mathsf{negl}(\lambda)$ (signatures generation rate gap)

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

We will assume from now on, that the hardness parameter used in our protocols, is one satisfying the above conditions.

Remark 5. The better the adversarial signing algorithm may be compared to the honest one, the closer δ_{SoW} is to 0, while the closer the number of adversarial steps $t'_{\mathcal{A}}$ are to that of the honest parties, the closer δ_{Steps} is to 0. Assumption 3 implies, in a quantitative manner, that the better the adversarial signing algorithm, the smaller the computational power of the adversary we can tolerate.

Next, we focus on the hash functions used by Bitcoin, and the necessary security assumptions to avoid cycles in the blockchains. In previous chapters, this was avoided by taking advantage of the unpredictability of the RO, see Definition 11. Instead, here we want to avoid relying on idealized assumptions, and thus we are going to take advantage of the collision resistance property of the underlying hash function. First, note that in the actual implementation of Bitcoin an unkeyed hash function is used, namely, a double invocation of SHA-256. In previous analyses of the protocol this was modeled as a random oracle. We choose to model it in a strictly weaker way, as a keyed hash function family $\mathcal{H}$ that is *collision resistant* (Definition 112); the CRS we have already assumed will contain the key of our hash function. Given security parameter λ and hash key $k \in K'$, we assume that the corresponding hash function is of the form:

$$H_k : \{0,1\}^{\log|K|+\lambda+\log|S|} \to \{0,1\}^{\log|K|}$$

where K, S are the key and signature sets of SoW. Moreover, as depicted in Figure 4.3, the protocol makes use of another hash function G to compress the input x of each block, which may be of arbitrary size. In our analysis we will require G to be collision resistant. It is well known (see, e.g., [Dam89, BJL17]) that given a fixed-length collision-resistant hash function family, we can construct an arbitrary-length collision-resistant hash function family. To aid readability, we will sometimes omit the keys of both functions (as we already do in the description of the protocol). Furthermore, observe that the hash structure of any blockchain (depicted in Figure 4.3) is similar to the Merkle-Damgaard transform [Dam89]:

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

where the fixed-length hash function family used is always assumed to be $\mathcal{H}$. 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 $\mathcal{C} = B_1, \ldots, B_m$, where $B_i = \langle s_i, x_i, \sigma_i \rangle$, it holds that for any $j \in [m]$: $H_k(\text{head}(\mathcal{C})) = \mathsf{MD}_k(H_k(B_j), (G_k(x_i)||\sigma_i)_{i \in \{j+1,\ldots,m\}})$.

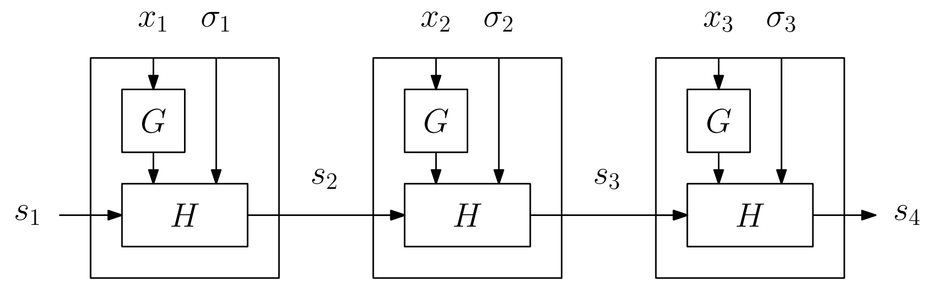


Figure 4.3: *The hash structure of the blocks in the Bitcoin protocol.*

Lemma 44. *Let $\mathcal{H}, \mathcal{G}$ be collision-resistant hash function families. The probability that any PPT RAM $\mathcal{A}$, given B_{Gen}, can find two distinct valid chains $\mathcal{C}_1, \mathcal{C}_2$ such that $H(\mathcal{C}_1) = H(\mathcal{C}_2)$, is negligible in λ.*

Proof. Let $\mathcal{C}_1 = B_{\texttt{Gen}}, B_1, \ldots, B_{|\mathcal{C}_1|}$, $\mathcal{C}_2 = B_{\texttt{Gen}}, B'_1, \ldots, B'_{|\mathcal{C}_2|}$, $m_1 = (G(x_i)||\sigma_i)_{i \in [|\mathcal{C}_1|]}$ and $m_2 = (G(x'_i)||\sigma'_i)_{i \in [|\mathcal{C}_2|]}$, where x_i, σ_i are the data and the signature of block B_i. 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 $\mathsf{MD}(H(B_{\texttt{Gen}}), m_1) = \mathsf{MD}(H(B_{\texttt{Gen}}), m_2)$.

We will construct an adversary $\mathcal{A}'$ that breaks the collision resistance of H also with non-negligible probability. We take 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 $x_{|\mathcal{C}_1|}, x'_{|\mathcal{C}_2|}$ (cf. Table 4.1), it follows that $x_{|\mathcal{C}_1|} \neq x'_{|\mathcal{C}_2|}$ and with overwhelming probability $G(x_{|\mathcal{C}_1|}) \neq G(x'_{|\mathcal{C}_2|})$, which in turn implies that $B_{|\mathcal{C}_1|} \neq B'_{|\mathcal{C}_2|}$. Since $H(\text{head}(\mathcal{C}_1)) = H(\text{head}(\mathcal{C}_2))$, 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 $\mathsf{MD}(H(\mathsf{Gen}), ((G(x_i)||\sigma_i))_{i \in [\ell-1]}) \neq \mathsf{MD}(H(\mathsf{Gen}), ((G(x'_i)||\sigma'_i))_{i \in [\ell-1]})$ or $(G(x_\ell), \sigma_\ell) \neq (G(x'_\ell), \sigma'_\ell)$, and thus a collision has been found. The lemma follows.

□

Lemma 44 implies that the insertion and copy properties [GKL15] (see Definition 12), that have to do with the way blocks are connected, do not occur with overwhelming probability in λ.[8]

Corollary 45. *Let $\mathcal{H}, \mathcal{G}$ be collision-resistant hash function families. Then, for any PPT $\mathcal{A}, \mathcal{Z}$ no insertions or copies occur in* $\mathrm{VIEW}^{t,n}_{\Pi^{\mathsf{SoW}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$ *with probability* $1-\mathsf{negl}(\lambda)$.

We proceed to the main part of the analysis. We follow the notation and definitions of Section 3.3.2, for: successful and uniquely successful rounds, honest, adversarial and u.s. blocks, random variables $X(\cdot), Y(\cdot)$. Moreover, with respect to some block B computed by an honest party P at some round r, let $Z^P_r(S)$ denote the number of distinct blocks diffused by the adversary during S that have B as their ancestor; note that honest parties compute at most one block per round. If P is corrupted or did not compute any block at r, let $Z^P_r(S)=0$. Define $X^P_r(S)$ similarly.

Lower bounds on the expected rate of successful and uniquely successful rounds are defined as follows:

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

We will formally show that this is the case later, in Lemma 49. As in previous chapters we show that the rate of uniquely successful rounds outperforms the rate at which the adversary produces signatures.

Lemma 46. *Assume a SoW scheme that complies with Assumptions 2 and 3. It holds that* $\gamma \geq (1+\delta)\beta t'_{\mathcal{A}}$.

Proof. It holds that:

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

where we have first used Bernouli's inequality, and then the three conditions from Assumption 3. The last inequality follows from the fact that $\frac{\delta_{\mathsf{Steps}}-\delta_{\mathsf{SoW}}}{2} \geq \delta$. $\square$

Next, we prove that the adversary cannot mine blocks that extend an honest block created recently at a very high rate with probability better than that of breaking the MU-TCMA property. For a summary of our notation we refer to Table 4.2.

Lemma 47. *Let $\mathcal{H}, \mathcal{G}$ be collision-resistant hash function families, and assume a SoW scheme that complies with Assumptions 2 and 3. For any set of consecutive rounds S, for any party P, and any round $i \in S$, the probability that $Z^P_i(S) \geq \lfloor \beta t'_{\mathcal{A}}|S|\rfloor$ is at most* $\varepsilon(h, t'_{\mathcal{A}}\cdot|S|, n\cdot|S|)$.

[8] A third property, called "prediction," also introduced in [GKL15], is not needed in our proof as it is captured by the fact that SoW is MU-TCMA secure even in the presence of adversarial precomputation.

λ :	security parameter
n :	number of parties
$t_{\mathcal{H}}$:	number of steps per round per honest party
$t_{\mathcal{A}}$:	total number of adversarial steps per round
$t'_{\mathcal{H}}$:	lower bound on number of Sign steps per round per honest party
$t'_{\mathcal{A}}$:	round simulation cost, excluding honest Sign calls
θ :	upper bound on the number of messages sent by the adversary per round
β :	upper bound on SoW computation rate per step
γ :	lower bound on the rate of uniquely successful rounds
f :	lower bound on the rate of successful rounds
δ_{SoW} :	adversarial advantage in SoW computation rate
δ_{Steps} :	honest advantage in number of steps per round
δ :	lower bound on $\delta_{\mathsf{Steps}} - \delta_{\mathsf{SoW}}$, upper bound on block generation rate

Table 4.2: *The parameters of our analysis. Parameters* $\lambda, n, t_{\mathcal{H}}, t_{\mathcal{A}}, t'_{\mathcal{H}}, t'_{\mathcal{A}}, \theta \in \mathbb{N}$, *while* $\beta, \gamma, f, \delta_{\mathsf{SoW}}, \delta_{\mathsf{Steps}}, \delta \in \mathbb{R}$.

Proof. W.l.o.g., assume that i is the first round of $S = \{i'|i \leq i' < i+s\}$ and party P is honest and mined a block B at round i. Let E be the event where in $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$ the adversary has mined at least $\lfloor \beta t'_{\mathcal{A}} s \rfloor$ blocks until round $i+s$ that descend B. For the sake of contradiction, assume that the lemma does not hold, and thus the probability that E holds is greater than $\varepsilon(h, t'_{\mathcal{A}} \cdot s, n \cdot s)$. Using $\mathcal{A}$, we will construct an adversary $\mathcal{A}'$ that wins the MU-TCMA game with probability greater than that. $\mathcal{A}'$ is going to run internally $\mathcal{A}$ and $\mathcal{Z}$, while at the same time perfectly simulating the view of honest parties using the signing oracle that he has in his disposal on the MU-TCMA game. This way, the view of $\mathcal{A}, \mathcal{Z}$ will be indistinguishable both in the real and the simulated runs, and thus the probability that E happens will be the same in both cases. A sketch of the reduction is given at Figure 4.4.

We are going to describe the two stages of $\mathcal{A}'$ separately, i.e. before and after obtaining vk. First, $\mathcal{A}'_1$ creates the genesis block and sets the fixed length hash key and the SoW public parameters to be Σ and pp, respectively. Then, he 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. He can do this since he 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 indistinguishable compared to the real

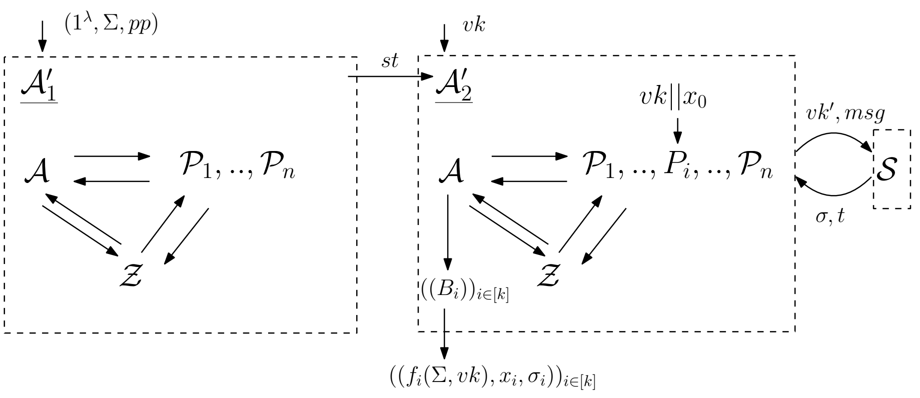

$(1^\lambda, \Sigma, pp)$
$\mathcal{A}'_1$
$\mathcal{A}$
$\mathcal{P}_1, .., \mathcal{P}_n$
$\mathcal{Z}$
st
vk
$\mathcal{A}'_2$
$vk||x_0$
$\mathcal{P}_1, .., P_i, .., \mathcal{P}_n$
$((B_i))_{i\in[k]}$
vk', msg
$\mathcal{S}$
σ, t
$((f_i(\Sigma, vk), x_i, \sigma_i))_{i\in[k]}$

Figure 4.4: *The figure depicts a schematic of the reduction from the Bitcoin backbone to the MU-TCMA game of Lemma 47.*

one.

For the second stage, $\mathcal{A}'_2$, 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. Moreover, $\mathcal{A}'_2$ is again going to simulate honest parties behavior, from round i until round $i+s$, but in a different way. Instead of running the Sign algorithm for each non-corrupted honest party at every round, it makes a query to the signing oracle $\mathcal{S}$ with the respective parameters. Then, it checks if the honest party succeeded in making a signature in this round by comparing the number of steps needed to make this signature to the number of steps available to the party at this round. Hence, $\mathcal{A}'_2$ has to do n queries to the signing oracle per round. The adversary can also send up to θ messages per round to honest parties which they have to verify, thus inducing an additional $\theta \cdot t_{\mathsf{ver}}$ overhead in the simulation. Note that $\mathcal{A}'_2$ has to run the verification procedure only once per message.

Continuing with the description of $\mathcal{A}'_2$, as shown in Figure 4.4, it takes as input a key vk generated from $\mathsf{KeyGen}(pp)$. We should somehow relate vk to the blocks the internal adversary is going to produce. In our reduction, this is achieved by: (i) relating the block B that party P generates at round i with vk through the input contribution function $I(\cdot)$, and (ii) by the fact that the seed of all blocks that have B as an ancestor is related to $H(B)$. In more detail, at round i, $\mathcal{A}'_2$ will use vk in the neutral transaction included in $I(\cdot)$ for P; denote by $vk||x_0$ the output of I for P at this round. If P is successful at this round and mines a block $B = \langle s_0, vk||x_0, \sigma_0\rangle$ (this can be simulated using $\mathcal{S}$), then any block $B' = \langle s, x, \sigma \rangle$ descending B will be related to it as follows:

$$\begin{aligned} s &= \mathsf{MD}(H_\Sigma(B), (G_\Sigma(x_i)||\sigma_i)_i) \\ &= \mathsf{MD}(H_\Sigma(s||G_\Sigma(vk||x_0)||\sigma_0), (G_\Sigma(x_i)||\sigma_i)_i) \\ &\overset{\text{def}}{=} f_{(s,\{x_i,\sigma_i\}_i)}(\Sigma, vk) \end{aligned}$$

for some $((x_i, \sigma_i))_i$ due to Fact 1. Observe, that the seed of B' is a function of Σ and vk, as required by the MU-TCMA game. In fact the tampering function class we will consider is going to be exactly the set of all these functions f. More formally, let C be the set of sequences $((x_i, \sigma_i))_i$ that correspond to a valid chain in the way described before. Then, a member of the tampering function class we will be considering is defined as follows:

$$F_\lambda = \{f_{s,a}\}_{s \in \{0,1\}^\lambda, a \in C}$$

We show next, as required by our assumption regarding the SoW scheme, that $\mathcal{F}$ is computationally unpredictable.

Claim 4. *$\mathcal{F}$ is computationally unpredictable.*

Proof. For the sake of contradiction, assume that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the computational unpredictability property of $\mathcal{F}$. This implies that

$$\Pr_{\substack{pp \leftarrow \mathsf{PPub}(1^\lambda); \\ vk \leftarrow \mathsf{KeyGen}(pp); \\ \Sigma \leftarrow \mathcal{U}_\lambda;}} \begin{bmatrix} (st, y) \leftarrow \mathcal{A}_1(1^\lambda, \Sigma, pp); \\ f \leftarrow \mathcal{A}_2(1^\lambda, st, vk) : \\ f \in F_\lambda \wedge f(\Sigma, vk) = y \end{bmatrix}$$

is non-negligible. W.l.o.g, also assume that no PPT adversary can guess vk given pp, as this easily implies that no function family $\mathcal{F}$ can be computationally unpredictable, and the SoW scheme is trivially insecure as described in Section 4.3.

We are going to describe an adversary $\mathcal{A}'$ that uses $\mathcal{A}$ to break the collision resistance property of H. Given Σ, $\mathcal{A}'$ first runs $\mathcal{A}_1(\Sigma, pp)$ and obtains a prediction y and state st. Next, $\mathcal{A}'$ randomly samples vk_1, vk_2 using KeyGen and runs $\mathcal{A}_2$ twice on inputs st, vk_1 and st, vk_2 respectively. By an application of the splitting lemma we can show that with non-negligible probability $\mathcal{A}_2$ will output (not necessarily different) functions f_1, f_2 such that $y = f_1(\Sigma, vk_1) = f_2(\Sigma, vk_2)$. As noted earlier, this corresponds to equality of the hashes of two chains. Moreover, $vk_1 \neq vk_2$ with overwhelming probability in λ, as otherwise the adversary would be able to guess vk by simply running $\mathsf{KeyGen}(pp)$. Hence, due to the collision resistance of G, the two chains start with two different blocks with overwhelming probability. Using similar techniques as in Lemma 44, we can show that $\mathcal{A}'$ can find a collision in H using f_1, f_2, vk_1, vk_2 with non-negligible probability in λ, which is a contradiction. $\dashv$

Since $\mathcal{A}$ and $\mathcal{Z}$ cannot distinguish between the bitcoin execution and the one we described above, E will occur with probability at least $\varepsilon(h, t'_{\mathcal{A}}s, ns)$, i.e. $\mathcal{A}$ will compute at least $\lfloor \beta t'_{\mathcal{A}} s \rfloor$ blocks starting from round i and up to round $i+s$ that descend B. Note, that these blocks are also valid signatures, whose keys are of the form $f(\Sigma, vk)$, for (possibly different) f's. Moreover, the event that the adversary outputs different f_i, f_j such that $f_i(\Sigma, vk) = f_j(\Sigma, vk)$, corresponds to finding chains $\mathcal{C}_1, \mathcal{C}_2$ such that $H(\mathcal{C}_1) = H(\mathcal{C}_2)$. By Lemma 44, this happens with negligible probability. Hence, $\mathcal{A}'$ will win the MU-TCMA game with respect to tampering function class $\mathcal{F}$ with probability greater than $\varepsilon(h, t'_{\mathcal{A}}s, ns)$, while being $s \cdot (t_{\mathcal{A}} + \theta \cdot t_{\mathsf{ver}} + t_{\mathsf{bb}} \cdot n) = s \cdot t'_{\mathcal{A}}$-bounded and having made at most ns queries to the signing oracle, which is a contradiction to our initial assumption.

$\square$

Note that we can do exactly the same reduction without using the oracle to simulate the signing procedure of the honest parties. Then, the total running time of the second stage of $\mathcal{A}'$ is on the worst case $s \cdot (t_{\mathcal{A}} + nt_{\mathcal{H}})$-bounded and hence the probability it can win is $\varepsilon(h, s \cdot (t_{\mathcal{A}} + nt_{\mathcal{H}}), 0)$. We derive the following bound on the total number of blocks produced by both honest and malicious parties during a certain number of rounds.

Corollary 48. *Let $\mathcal{H}, \mathcal{G}$ be collision-resistant hash function families, and assume a SoW scheme that complies with Assumptions 2 and 3. For any set of consecutive rounds S, for any party P, and any round $i \in S$, the probability that $Z_i^P(S) + X_i^P(S) \geq \lfloor \beta(t_{\mathcal{A}} + nt_{\mathcal{H}}) \cdot |S| \rfloor$ is less than $\varepsilon(h, |S| \cdot (t_{\mathcal{A}} + nt_{\mathcal{H}}), 0)$.*

Next, we prove lower bounds on the rate of successful and uniquely successful rounds in any big enough interval of rounds. Our proof crucially depends on the run-time independence property of the SoW scheme. Specifically, the property implies that for some set of rounds the sum of the Bernoulli random variables of the event that a round is (resp. uniquely) successful concentrate around the mean. Moreover, using the MU-TCMA and Successful properties we can compute a lower bound on this mean, and thus lower-bound the rate of uniquely successful rounds with good probability.

Lemma 49. *Assume a SoW scheme that complies with Assumptions 2 and 3. For any set of consecutive rounds S, with $|S| \geq \frac{\lambda}{\gamma\delta^2}$, the following two events occur with negligible probability in λ:*

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

Proof. W.l.o.g., let $S = \{1,\dots,s\}$. Our proof strategy will be to first prove the results of the lemma in an "ideal" execution where honest parties behavior is simulated using the distribution defined in the runtime independence property. Due to the fact that the random variables defined in this property are i.i.d., analyzing the distributions of successful and uniquely successful rounds is a lot easier. We finish the proof by showing that these two distributions are statistically indistinguishable in the "real" and "ideal" executions.

More formally, we start by describing the two experiments. First, let $\{L_{i,j}\}_{(i,j)\in[s]\times[n]}$ denote the set of i.i.d. random variables which are guaranteed to exist by the runtime independence property. Let $E_{\texttt{real}}(1^\lambda)$, $E_{\texttt{ideal}}(1^\lambda)$ be two random experiments. $E_{\texttt{real}}(1^\lambda)$ simulates the protocol execution we have defined, and at the end of the experiment the number of successful and uniquely successful rounds that happened during S is output, while $E_{\texttt{ideal}}(1^\lambda)$ is defined in the same way, except that whenever it is the turn of an honest party P_i at round $j \in S$ to run the Sign routine, it instead samples a value from $L_{i,j}$. If this value is larger than the available number of steps it has to run Sign, it acts as if it did not find a new signature in this round. Otherwise, it samples a signature according to the distribution induced by the Sign routine for its parameters pp, vk, m, h in this round, conditioned on the fact that the routine should take at most the available number of steps; the sampling part may take many more steps than the available steps the party has for this round.

We start our analysis by showing an upper and a lower bound for the random variables in $\{L_{i,j}\}_{(i,j)\in[s]\times[n]}$. By the runtime independence property, it holds that for any i, j, pp, vk, m: $\Delta[L_{i,j}, \mathsf{Steps}_{\mathsf{Sign}}(pp, vk, m, h)] \leq \mathsf{negl}(\lambda)$. Moreover, for any predicate ϕ on the image of $L_{i,j}$, it holds that $|\Pr[\phi(L_{i,j})] - \Pr[\phi(\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, m, h))]| \leq \mathsf{negl}(\lambda)$, as otherwise we would be able to use ϕ to distinguish between the two random variables with non-negligible probability, which is a contradiction. For $\phi(x) \overset{def}{=} x \leq t'_{\mathcal{H}}$, and due to the Successful property, it follows that $\Pr[L_{i,j} \leq t'_{\mathcal{H}}] \geq \alpha(h) - \mathsf{negl}(\lambda)$. Next, we prove an upper bound on $\Pr[L_{i,j} \leq t_{\mathcal{H}}]$.

Claim 5. *For all $(i,j) \in [s]\times[n] : \Pr[L_{i,j} \leq t_{\mathcal{H}}] \leq \beta t_{\mathcal{H}} + \mathsf{negl}(\lambda)$.*

Proof of Claim. Assume that the claim does not hold, and thus there exist some i, j such that $\Pr[L_{i,j} \leq t_{\mathcal{H}}] - \beta t_{\mathcal{H}}$ is non-negligible. From the discussion above, for every pp, vk, m, random variable $L_{i,j}$ is indistinguishable from $\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, m, h)$, hence $\Pr[\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, m, h) \leq t_{\mathcal{H}}] - \beta t_{\mathcal{H}}$ is also non-negligible. We are going to show that this cannot hold, as it would violate the MU-TCMA property.

Assume an attacker $\mathcal{A}$ that on input pp, vk, h, picks $k = \lceil \lambda/\beta \rceil$ different messages $m_1, \ldots, m_k$ and runs then Sign routine for each of the messages for $t_{\mathcal{H}}$ steps. Let random variable W_i be equal to 1, iff for the i-th invocation of Sign it holds that $\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, m_i, h) \leq t_{\mathcal{H}}$, and 0 otherwise. By the discussion above, it holds that $\mathbb{E}[W_i] - \beta t_{\mathcal{H}}$ is non-negligible, and from the linearity of expectation $\mathbb{E}[\sum_{i=1}^{k} W_i] - k\beta t_{\mathcal{H}}$ is also non-negligible. Note that $\sum_{i=1}^{k} W_i$ corresponds to the number of SoWs that $\mathcal{A}$ will compute in $kt_{\mathcal{H}}$ steps. On the other hand, for any adversary that runs for $kt_{\mathcal{H}}$ steps, it holds that it computes less than $\lfloor \beta k t_{\mathcal{H}} \rfloor$ SoWs, with overwhelming probability on $\beta k t_{\mathcal{H}}$, thus also on λ. In turn this implies that on expectation any adversary will compute less than $\lfloor \beta k t_{\mathcal{H}} \rfloor + \mathsf{negl}(\lambda)$ SoWs. This contradicts the fact that for $\mathcal{A}$ it holds that $\mathbb{E}[\sum_{i=1}^{k} W_i] - k\beta t_{\mathcal{H}}$ is non-negligible, and thus the claim follows. $\dashv$

We proceed to analyze the probability of successful and uniquely successful rounds occurring in $E_{\mathtt{ideal}}(1^{\lambda})$. Let random variables $\hat{X}(\cdot), \hat{Y}(\cdot)$ be defined similarly to $X(\cdot), Y(\cdot)$ for $E_{\mathtt{ideal}}(1^{\lambda})$. We define an additional two random variables $X'(\cdot), Y'(\cdot)$ that will be helpful in our analysis. $X'(\{i\})$ is equal to 1, where i is some round of the execution, if there exists some party, among the first $n - t$ honest parties that are activated at round i, such that $L_{i,j} \leq t'_{\mathcal{H}}$. Note that $X'(\{i\}) = 1$ implies that $\hat{X}(\{i\}) = 1$. Further, if we define $X'(S) = \sum_{i \in S} X'(\{i\})$, it follows that $\hat{X}(S) \geq X'(S)$. Next, let $Y'(\{i\})$ be equal to 1, if there exists a unique party, among the first $n - t$ honest parties that are activated at round i, such that $L_{i,j} \leq t'_{\mathcal{H}}$, and for all other $n - 1$ parties it holds that $L_{i,j} > t_{\mathcal{H}}$. Again, it holds that $\hat{Y}(S) \geq Y'(S)$. Note that due to the way $X'(\cdot), Y'(\cdot)$ are defined, and the fact that $\{L_{i,j}\}_{(i,j) \in [s] \times [n]}$ are i.i.d random variables, these two random variables do *not* depend on the behavior of the adversary and thus are easier to analyze. We proceed to analyze their expectations.

Claim 6. *For any $i \in S$, it holds that $\mathbb{E}[Y'(\{i\})] \geq \gamma$ and $\mathbb{E}[X'(\{i\})] \geq f$.*

Proof of Claim.

$$\begin{aligned}\mathbb{E}[Y'(\{i\})] =&\Pr[Y'(\{i\})=1]\\ =&\Pr[\bigvee_{j\in[n-t]}(L_{i,j}\le t'_{\mathcal{H}}\wedge\bigwedge_{m\in[n]\setminus\{j\}}L_{i,m}>t_{\mathcal{H}})]\\ =&\sum_{j\in[n-t]}\Pr[L_{i,j}\le t'_{\mathcal{H}}]\cdot\prod_{m\in[n]\setminus\{j\}}\Pr[L_{i,m}>t_{\mathcal{H}}]\\ \ge&(n-t)\cdot\alpha(h)\cdot(1-\beta t_{\mathcal{H}})^{n-1}=\gamma\end{aligned}$$

where the second equality follows from the fact that the events are mutually exclusive, and the last inequality follows from the bounds established earlier about $L_{i,j}$.

$$\begin{aligned}\mathbb{E}[X'(\{i\})] =&\Pr[X'(\{i\})=1]=\Pr[\bigvee_{j\in[n-t]}L_{i,j}\le t'_{\mathcal{H}}]\\ =&1-\Pr[\bigwedge_{j\in[n-t]}L_{i,j}>t'_{\mathcal{H}}]\\ =&1-\prod_{j\in[n-t]}\Pr[L_{i,j}>t'_{\mathcal{H}}]\\ \ge&1-(1-\alpha(h))^{n-t}=f\end{aligned}$$

where the last inequality follows as before. ⊣

By the linearity of expectation we have that $\mathbb{E}[Y'(S)]\ge\gamma|S|$ and $\mathbb{E}[X'(S)]\ge f|S|$. Moreover, it easy to see that $\{Y'(\{i\})\}_{i\in S}$ are independent, and thus we can apply the Chernoff Bound:

$$\Pr[Y'(S)\le(1-\frac{\delta}{4})\gamma|S|]\le\Pr[Y'(S)\le(1-\frac{\delta}{4})\mathbb{E}[Y'(S)]]\le e^{-\Omega(\delta^2\gamma|S|)}$$

Similarly, we can show that $\Pr[X'(S)\le(1-\frac{\delta}{4})f|S|]\le e^{-\Omega(\delta^2 f|S|)}$. Since $\hat{X}(S)\ge X'(S)$ and $\hat{Y}(S)\ge Y'(S)$, the same bounds hold for $\hat{X}(S),\hat{Y}(S)$.

We finish our proof by arguing that the two probabilistic inequalities we proved should also hold for $E_{\texttt{real}}(1^\lambda)$, and thus for our protocol execution, with negligible difference in probability. We will argue by a hybrid argument that the outputs of experiments $E_{\texttt{real}}(1^\lambda),E_{\texttt{ideal}}(1^\lambda)$ are statistically indistinguishable.

Let experiment $E_1(1^\lambda)$ be the same as $E_{\texttt{real}}(1^\lambda)$, with the only difference that party P_1 at round 1 acts as in the $E_{\texttt{ideal}}(1^\lambda)$ experiment, i.e., samples a value from $L_{1,1}$ instead of actually running the Sign routine with parameters $pp,vk_{1,1},m_{1,1},h$ to see if it succeeded, and computes a signature accordingly. For the sake of contradiction, assume that the outputs of $E_{\texttt{real}}(1^\lambda)$ and $E_1(1^\lambda)$ are not statistically indistinguishable.[9] Then, there exists an unbounded distinguisher D that can distinguish the output of the two experiments with non-negligible probability, i.e.,

$$|\Pr[D(E_1(1^\lambda)) = 1] - \Pr[D(E_{\texttt{real}}(1^\lambda)) = 1]| > \mathsf{negl}(\lambda)$$

where the probabilities are defined over the coins of the two experiments. Let random variable $\mathcal{R}$ denote the partial execution up to the point that party P_1 at round 1 is about to run the Sign routine in either of the two experiments; partial executions are fully specified by the value of the CRS and the coins used by $\mathcal{A}, \mathcal{Z}$ and all parties up to that point. Note, that this random variable is well defined and is equal in both experiments. By an averaging argument, it holds that:

$$\begin{aligned}
&|\Pr[D(E_1(1^\lambda)) = 1] - \Pr[D(E_{\texttt{real}}(1^\lambda)) = 1]| \\
&= |\sum_{R \in \text{supp}(\mathcal{R})} \Pr[R \leftarrow \mathcal{R}] \cdot (\Pr[D(E_1(1^\lambda)) = 1|R] - \Pr[D(E_{\texttt{real}}(1^\lambda)) = 1|R])| \\
&\leq \max_{R \in \text{supp}(\mathcal{R})} \{|\Pr[D(E_1(1^\lambda)) = 1|R] - \Pr[D(E_{\texttt{real}}(1^\lambda)) = 1|R]|\}
\end{aligned}$$

where supp$(\mathcal{R})$ is the support of $\mathcal{R}$. Hence, there exists some specific partial execution R such that the probability that D distinguishes between the two experiments conditioned on R is non-negligible. Moreover, R determines a unique set of parameters $pp, vk_{1,1}, m_{1,1}$ that party P_1 is going to use in its invocation of the Sign routine.

We can then use D to construct another distinguisher D' for the random variables $L_{1,1}$ and $\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_{1,1}, m_{1,1}, h)$. D' takes as input some value x, and works as follows: It simulates the real execution according to R up to the point where party P_1 at round 1 is going to run the Sign routine. Then, if x is larger that the available time to run the Sign routine, it simulates the behavior of party P_1 as if it failed to find a signature for this round. Otherwise, it generates a signature as specified in the ideal experiment. It continues by simulating the rest of the execution as in the real experiment, computing the number of successful and uniquely successful rounds, and running D on this input. D' outputs the value returned from D. If x is distributed as random variable $L_{1,1}$ (resp. $\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_{1,1}, m_{1,1}, h)$), then D' outputs 1 with the same probability that D outputs 1 on input from experiment $E_1(1^\lambda)$ (resp. $E_{\texttt{real}}(1^\lambda)$) conditioned on R. Note, that while in the $\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_{1,1}, m_{1,1}, h)$ case the actions of party P_1 are performed in a different way than in experiment $E_{\texttt{real}}(1^\lambda)$ (the Sign

[9] For simplicity, we say here that the random variables are statistically indistinguishable instead of the random variable ensembles. The relevant ensembles can be easily deduced.

routine is invoked more than once), the value input to D follows exactly the same distribution as the output of experiment $E_{\texttt{real}}(1^\lambda)$ conditioned on R. It follows that:

$$|\Pr[D'(L_{1,1}) = 1] - \Pr[D'(\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_{1,1}, m_{1,1}, h)) = 1]|$$
$$= |\Pr[D(E_1(1^\lambda)) = 1|R] - \Pr[D(E_{\texttt{real}}(1^\lambda)) = 1|R]| > \mathsf{negl}(\lambda)$$

D' succeeds with non-negligible probability in distinguishing the two random variables, which is a contradiction to the runtime independence property. Thus the outputs of experiments $E_{\texttt{real}}(1^\lambda)$, $E_1(1^\lambda)$ should be statistically indistinguishable.

We define $E_i(1^\lambda)$ recursively, by letting it be the same as $E_{i-1}(1^\lambda)$, except that the i-th honest invocation of Sign is simulated as in $E_{\texttt{ideal}}$. Then, by a similar argument as above we can show that $E_{i-1}(1^\lambda), E_i(1^\lambda)$ are indistinguishable. Note now that it holds that $E_{|S|\cdot n}(1^\lambda) \equiv E_{\texttt{ideal}}(1^\lambda)$. Therefore, $E_{\texttt{real}}(1^\lambda) \approx E_1(1^\lambda) \approx \ldots \approx E_{\texttt{ideal}}(1^\lambda)$, which by transitivity implies that $E_{\texttt{real}}(1^\lambda) \approx E_{\texttt{ideal}}(1^\lambda)$. By our previous analysis of $\hat{X}(S), \hat{Y}(S)$, the lemma follows with negligible difference in probability.

□

We are now ready to define the set of *typical executions* for this setting. However, here we will need to adapt the definition due to the difficulties associated with performing a reduction to the security of the SoW scheme.

Definition 50. [Typical execution] An execution is *typical* if and only if $\lambda \geq 9/\delta$ and for any set S of consecutive rounds with $|S| \geq \frac{2\lambda}{\gamma\delta^2}$, the following hold:

1. $Y(S) > (1 - \frac{\delta}{4})\gamma|S|$ and $X(S) > (1 - \frac{\delta}{4})f|S|$;
2. for any party P at any round i in S, $Z_i^P(S) < \frac{\gamma}{1+\delta} \cdot |S|$ and $Z_i^P(S) + X_i^P(S) < \beta(t_{\mathcal{A}} + nt_{\mathcal{H}}) \cdot |S|$; and
3. no insertions and no copies occurred.

Theorem 51. *Let $\mathcal{H}, \mathcal{G}$ be collision-resistant hash function families, and assume a SoW scheme that complies with Assumptions 2 and 3. An execution is typical with overwhelming probability in λ.*

Proof. In order for an execution to not be typical, one of the three points of Definition 50 must not hold with non-negligible probability for some big enough set of rounds. Point 3 is implied by Corollary 45. For a specific set of rounds S, where $|S| \geq \frac{2\lambda}{\gamma\delta^2}$, point 1 is implied by Lemma 49 with overwhelming probability in λ. Regarding point 2, by an application of Lemma 47 for $t'_{\mathcal{A}} = \frac{\gamma}{(1+\delta)\beta}$, it follows that $Z_i^P(S) <$

$\lfloor \frac{\gamma}{1+\delta} \cdot |S| \rfloor \leq \frac{\gamma}{1+\delta} \cdot |S|$ with probability $\mathsf{negl}(\beta \cdot t'_{\mathcal{A}} \cdot |S|)$, where $\beta t'_{\mathcal{A}} \cdot |S| \geq \beta \frac{\gamma}{(1+\delta)\beta} \cdot \frac{2\lambda}{\gamma\delta^2} = \Omega(\lambda)$. Note, that due to Assumption 3 and Lemma 46, necessarily $t'_{\mathcal{A}} \leq \frac{\gamma}{(1+\delta)\beta}$. Similarly, Corollary 48 implies that $Z_i^P(S) + X_i^P(S) < \beta(t_{\mathcal{A}} + nt_{\mathcal{H}}) \cdot |S|$ with overwhelming probability in λ. The latter part follows from inequality $\beta t_{\mathcal{H}} \geq \alpha$ argued in Lemma 49 and the definition of γ, since $\beta(t_{\mathcal{A}} + nt_{\mathcal{H}}) \cdot \frac{2\lambda}{\gamma\delta^2} \geq \frac{2\lambda}{\delta} = \Omega(\lambda)$. Hence, point 2 also follows with overwhelming probability in λ. Finally, we can bound the probability that an execution is not typical by applying the union bound on the negation of these events over all sets of consecutive rounds of sufficiently large size, where the probability of each event occurring is negligible in λ.

□

Next, taking advantage of the Assumption 3, we show that the rate at which the adversary computes blocks in a typical execution, is bounded by the rate at which uniquely successful rounds occur. With foresight, we note that this relation is going to be at the core of our security proof.

Lemma 52. *Assume a typical execution. For any set of consecutive rounds $S = \{i, \ldots, j\}$, where $|S| \geq \frac{2\lambda}{\gamma\delta^2}$, and for any party P and round r in S, it holds that $(1 - \frac{\delta}{4})Y(S \setminus \{i\}) > Z_r^P(S)$.*

Proof. It holds that:

$$(1 - \frac{\delta}{4})Y(S \setminus \{i\}) > (1 - \frac{\delta}{4})^2\gamma(|S| - 1) \geq \frac{\gamma|S|}{1+\delta} > Z_r^P(S)$$

where, the first and last inequalities follow from the assumption that the execution is typical, while the middle one follows from the fact that $|S| \geq 9/\delta \geq (1 - \frac{1}{(1-\delta/4)^2(1+\delta)})^{-1}$.

□

We can now use the machinery built in [GKL15] to prove the common prefix, chain quality and chain growth properties, and eventually Persistence and Liveness, with only minor changes.

Higher level properties. Next, we describe the minor changes one has to do after proving the typical execution theorem with respect to the analysis of [GKL15], in order to prove the security of the protocol in our model. We only give brief proof sketches of lemmas or theorems from [GKL15] that are exactly the same for our own setting.

Lemma 53. *(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 $\ell + X(\{r, \ldots, s-1\})$.*

Proof. The main idea of the proof of this lemma, is that after each successful round at least one honest party will receive a chain that is at least one block longer than the chain it had. Hence, from this round on all parties will mine chains that are at least that long, and the lemma follows. □

Theorem 54. *(Chain-Growth). Assume a typical execution. The chain-growth property holds with parameters* $\tau = (1-\frac{\delta}{4})f$ *and* $s \geq \frac{2\lambda}{\gamma\delta^2}$.

Proof. Let S be any set of at least s consecutive rounds. Then, since the execution is typical: $X(S) \geq (1-\frac{\delta}{4})f \cdot |S| \geq \tau \cdot |S|$. By Lemma 53, 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 55. *Assume a typical execution. Let B be some honest or the genesis block. Any sequence of $k \geq \frac{2\lambda}{\gamma\delta}$ consecutive blocks in some chain $\mathcal{C}$, where the first block in the sequence directly descends B, have been computed in at least k/δ rounds, starting from the round that B was computed.*

Proof. W.l.o.g, let B be an honest block computed by party P at round i. First, note that due to Assumption 3, it holds that $\beta(t_{\mathcal{A}}+nt_{\mathcal{H}}) < \delta$. For the sake of contradiction, assume that for some $k \geq \frac{2\lambda}{\gamma\delta}$, there is a set of rounds S', such that $|S'| < k/\delta$ and at least k blocks that descend block B have been computed during S'. This implies that there is a set of rounds S, where $|S| \geq \frac{2\lambda}{\gamma\delta^2}$, such that $X_i^P(S)+Z_i^P(S) \geq k \geq |S|\delta > |S|\beta(t_{\mathcal{A}}+nt_{\mathcal{H}})$. This contradicts the typicality of the execution, and the lemma follows. □

Lemma 56. *(Common-prefix Lemma). Assume a typical execution. 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 $k \geq \frac{2\lambda}{\gamma\delta}$.*

Proof. In Lemma 47, instead of bounding the number of blocks mined by the adversary in a set of rounds, we bound the number of blocks mined by the adversary with respect to some specific honest block. Using also Lemma 55, the proof is exactly the same as in [GKL15]. 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 57. *(Common-prefix). Assume a typical execution. The common-prefix property holds with parameter* $k \geq \frac{2\lambda}{\gamma\delta}$.

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

Theorem 58. *(Chain-Quality). Assume a typical execution. The chain-quality property holds with parameter $\mu < 1 - \delta/4$ and $\ell \geq \frac{2\lambda}{\gamma\delta}$.*

Proof. The main idea of the proof is the following: If the theorem does not hold, and due to the chain growth property, a large enough number of adversarial blocks will have been mined in a set rounds that satisfies the properties of Definition 50. However, the rate at which the adversary generates blocks is upper bounded, which leads to a contradiction. □

Finally, the Persistence and Liveness properties follow from the three basic properties, albeit with different parameters than in [GKL15].

Lemma 59. *(Persistence). Assume a typical execution. It holds that Π_{PL} satisfies Persistence for $k \geq \frac{2\lambda}{\gamma\delta}$.*

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 60. *(Liveness). Assume a typical execution. It holds that Π_{PL} satisfies Liveness with parameters $u = \frac{2k}{(1-\frac{\delta}{4})f}$ rounds and $k \geq \frac{2\lambda}{\gamma\delta}$.*

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. □

Theorem 61. *Assuming the existence of a collision-resistant hash function and a SoW scheme that complies with Assumptions 2 and 3, protocol $\Pi_{\mathsf{PL}}^{\mathsf{SoW}}$ implements a robust public transaction ledger with parameters $u = \frac{2k}{(1-\frac{\delta}{4})f}$ and $k \geq \frac{2\lambda}{\gamma\delta}$ except with negligible probability in λ.*

As a "sanity check," we show in Section 4.5 that the Bitcoin SoW scheme we outline there is secure both in the random oracle and the $\mathcal{F}_{\text{TREE}}$ model [PSS17] according to our definitions; moreover, according to the security parameters we obtain for the scheme, it is implied that the security guarantees we get from our black-box analysis of the Bitcoin backbone based on this scheme are similar to those proved in [GKL15, PSS17].

4.4.3 Consensus

In this section we show how to achieve consensus (a.k.a. Byzantine agreement [PSL80, LSP82]) under exactly the same assumptions used for proving the security of the Bitcoin backbone protocol in Section 4.4.2.

As mentioned earlier, in [GKL15] consensus is achieved under the Honest Majority Assumption by using a proof-of-work construction in a *non-black-box* way, through a mining technique called "2-for-1 PoWs." In more detail, the technique shows how miners can compute proofs of work for two different PoW schemes at the cost of one, while at the same time ensuring that their resources cannot be used in favor of one of the two schemes. However, the security proof for the resulting protocol crucially relies on the fact that each of the bits of the strings output by the random oracle are independently sampled, and thus goes again our stated goal of designing a SoW scheme that does not rely on such a strong independence assumption.

Here, we get rid of this requirement, by showing how blockchain-based consensus can be achieved by only using the SoW security properties we have defined, directly, and without the extra non-black-box machinery used in [GKL15]. The protocol is based on the Bitcoin backbone protocol, and formally specified by providing adequate definitions for the $\mathrm{V}, \mathrm{R}, \mathrm{I}$ functions presented in Section 4.4.2.

First, we define some additional notation and terminology that will be used in the remainder of the section. We will use the terms "input" and "vote" interchangeably, referring to the parties' input to the consensus problem. We will use $header(\langle s, x||vote, \sigma\rangle)$ to denote the "compressed" version of block $\langle s, x||vote, \sigma\rangle$[10], equal to $\langle s, G(x)||vote, \sigma\rangle$. Note, that as defined, the header of any block is of a fixed size. We also extend the definition of our hash function H as applied to headers of blocks. The hash of the header of some block B will be equal to the hash of B, i.e., $H((header(B)) = H(B) = H(s, G(x)||vote, \sigma)$ (note that the header of B provides all the information needed to calculate the hash of B).

We now present a high-level description of the protocol. The basic idea is that during block mining, parties are going to include in their blocks not only their own votes, but also headers of other blocks that they have seen and that *are not* part of their chain. Then, after a predetermined number of rounds, the parties will count the votes "referenced" in a prefix of their chain, including the votes found in the headers of the blocks referenced. In this way, they can take advantage of the robust transaction ledger

[10]We augment the block content x with a vote bit. This does not change the results of the analysis of the previous section.

built in Section 4.4.2. The Persistence property implies that the honest parties will all agree on *which* votes should be counted, while the Liveness property guarantees that the majority of the counted votes come from honest parties.

The reader may wonder about the reason behind honest parties including in their blocks also headers of other blocks that they have seen but that are not part of their chain. It's because, as shown in [GKL15], the adversary is able to add more blocks in the main chain than his ratio of mining power (e.g., using a selfish-mining attack). This does not hold if the honest parties are able to also count off-chain blocks as our protocol does.

Algorithm 11 The *content validation predicate*. The input is the contents of the blocks of some chain.

```
function V($\langle x_1, \ldots, x_m \rangle$)
    $D \leftarrow$ new AVL()                                  ▷ Create a new (empty) AVL tree.
    $D.add(H(B_{\mathsf{Gen}}))$                    ▷ Add the hash of the genesis block on the tree.
    for $i = 1, ..., m$ do
        $queue \leftarrow$ references$(x_i)$                  ▷ Add all block references in a queue.
        $\langle r||height \rangle \leftarrow queue.top()$
        if $height \neq i$ then
            return False                          ▷ Check for the correct block "height".
        while $queue \neq \emptyset$ do
            $\langle s, G(x)||vote, w \rangle \leftarrow queue.top()$
            if $((D.exists(s)) \wedge \mathsf{Verify}(s, G(x)||vote, h, w))$ then
                $D.add(H(\langle s, G(x)||vote, w \rangle))$         ▷ Add new entry on the tree.
                $queue.pop()$
            else
                return False                             ▷ If not, the chain is invalid.
    return True
```

A main technical challenge is to be able to add the block references without making the honest parties' chains grow too large, and at the same time to ensure that the number of honest votes exceeds the adversarial ones. To overcome this challenge, we modify the Sign algorithm so that it is run on the header of the block, i.e., $\mathsf{Sign}(pp, s, G(x)||vote, h)$ and $\mathsf{Verify}(pp, s, G(x)||vote, h, \sigma)$, respectively. This way we are able to verify the validity of a block as a SoW and determine the block's vote by only knowing its header. These are exactly the properties we need for the consensus application.

Moreover, we should be able to tell whether the referenced blocks are “fresh”; that is, the adversary should not be able to reference blocks that it has precomputed and are not related to the genesis block. We achieve this by requiring blockchain contents to have a special structure in order to be considered valid by the content validation predicate $\mathrm{V}(\cdot)$ (Algorithm 11). A chain will be *valid* when the referenced blocks on every prefix of the chain form a *tree* that has the genesis block at its root. In order to check this efficiently, we require that the block headers listed in each block are ordered, so that each entry extends some block header found in previous entries of the same or parent blocks.

In more detail, to efficiently check for membership in the hash tree, in line 2 of Algorithm 11 we use an `AVL` tree. (Any other data structure supporting efficient updates and search would also work.) In line 5, the referenced blocks are extracted and pushed into a queue. We note that during this process it is checked that: (i) the contents of the block have a correct format, i.e., a vote field and list of block headers, (ii) each header in the list is a valid SoW and extends a chain starting from the genesis block, and (iii) that the first reference includes a string r and the height of the block as required in the security analysis of Section 4.4.2 and described in Table 4.1.

Content validation predicate $\mathrm{V}(\cdot)$	As defined in Algorithm 11.
Chain reading function $\mathrm{R}(\cdot)$	$\mathrm{R}(\cdot)$ outputs the majority of the votes found in the block headers of the first M blocks of the selected chain.
Input contribution function $\mathrm{I}(\cdot)$	The input function $\mathrm{I}(\cdot)$ maintains state of which blocks have been received, and outputs an input value x that contains (i) the headers of all valid blocks that extend the genesis and are not mentioned in the chain $\mathcal{C}$ that the party is currently extending, (ii) a neutral transaction of the form $\mathsf{KeyGen}(pp)\|\|\mathcal{C}\|$, and (iii) the party's input (i.e., 0 or 1).

Table 4.3: *The instantiation of functions* $\mathrm{I}(\cdot), \mathrm{V}(\cdot), \mathrm{R}(\cdot)$ *for protocol* $\Pi^{\mathtt{SoW}}_{\mathsf{BA}}$.

The algorithm runs for L rounds, after which it outputs the majority of the votes found in a prefix of the selected chain, of a predetermined length M. We call the resulting protocol $\Pi^{\mathtt{SoW}}_{\mathsf{BA}}$ (“BA” for Byzantine agreement). A description of the consensus protocol (specifically, the V, R, I functions) is presented in Table 4.3, and also recall the example in Figure 4.1. Note that all parties terminate the protocol simultaneously.

Theorem 62. *Assuming the existence of a collision-resistant hash function and a SoW scheme that complies with Assumptions 2 and 3. Protocol $\Pi_{\mathsf{BA}}^{\mathsf{SoW}}$ solves consensus in $O(\frac{\lambda}{\gamma^3\delta})$ rounds with overwhelming probability in λ.*

Proof. We are going to show that protocol $\Pi_{\mathsf{BA}}^{\mathsf{SoW}}$, parameterized with $k = \frac{2\lambda}{\gamma\delta}$, $M = k + \frac{8k}{\gamma}$ number of blocks, and $L = \frac{M+k}{(1-\frac{\delta}{4})\gamma}$ number of rounds solves consensus with overwhelming probability in λ. Our analysis uses many of the intermediate lemmas established for the proof of Theorem 61.

We start, by proving that the Agreement property holds. First, note that our definition of $V(\cdot)$ guarantees that if an honest party accepts a chain as valid, all other parties are also going to accept it as valid, since the validation predicate only depends on the chain that is being validated. Assume that an execution is δ-typical. Since $L \geq \frac{2\lambda}{\gamma\delta}$, after L rounds: (i) due to chain growth the chains of all honest parties will have length at least $M+k$ blocks, and (ii) due to the common prefix property they will all agree on the first M blocks of their chains. Hence, all honest parties will decide on their output values based on the "votes" mentioned in the same blocks, and thus they will all agree on the same value.

Regarding Validity, we are going to show that the majority of the counted "votes," i.e., from blocks and block headers found in blocks $B_1, \ldots, B_M$ of the common prefix, have been mined by honest parties. Due to the chain quality property at least one block from $B_{M-k}, \ldots, B_M$ is honest. Assume that the last honest block in this chain has been diffused in the network at round r. Since $M - k \geq \frac{8k}{\gamma} \geq k$, by an application of Lemma 55 starting from the genesis block, it holds that $r \geq \frac{8k}{\gamma\delta}$. Let $r' = r + \frac{2k}{(1-\frac{\delta}{4})\gamma}$. At round r', by the chain growth property all parties will have chains of length at least $M+k$, and by the common-prefix property all blocks up to the M-th position will be fixed for the rest of the execution. Hence, the last adversarial block in $B_1, \ldots, B_M$ must have been computed before round r'.

It remains to show that for $S_1 = \{1, \ldots, r\}$ and $S_2 = \{1, \ldots, r'\}$ it holds that $Z(S_2) < X(S_1)$, where random variable $Z(S_2)$ corresponds to the number of blocks diffused by the adversary during S_2 that extend the genesis block.

$$Z(S_2) < \frac{1}{1+\delta}\gamma r' \leq (1-\frac{\delta}{4})(1-\frac{\delta}{4})\gamma r' \leq (1-\frac{\delta}{4})\gamma r < X(S_1).$$

The first and the last inequalities hold due to the fact that the execution is typical. The fourth inequality follows by the size of r. The theorem follows since the majority of the referenced blocks in the chain agreed upon, have been mined by honest parties.

Concluding, notice that the total size of any chain is bounded by the total number of blocks mined, since each block's header is mentioned at most once in a single chain. Hence, in s rounds of a typical execution a chain has size in the order of $O(s \cdot \delta \cdot \lambda)$ bits. $\square$

4.5 SoW Constructions from Idealized Assumptions

In this section, and as a sanity check, we outline a SoW scheme that is secure both in the random oracle and the $\mathcal{F}_{\text{TREE}}$ model [PSS17]. Moreover, according to the security parameters we obtain for the scheme, the security guarantees we get from our black-box analysis of the Bitcoin backbone are similar to those proved in [GKL15, PSS17].

SoW in the RO model. Our first step is to show that the SoW scheme used in the Bitcoin protocol (call it $\mathtt{BSOW}$) is secure in the random oracle model according to our definitions.

In a nutshell, Bitcoin's Sign algorithm tries to find a block header with a small hash. The main components of the header are as follows: (i) the hash of the header of the previous block, (ii) the hash of the root of the Merkle tree of the transactions that are going to be added to Bitcoin's ledger, including the *randomly* created coinbase transaction, and (iii) a counter. The algorithm works by first fetching available transactions from the network, then computing a random public key that will be used for the coinbase transaction, and then iteratively incrementing the counter and calculating the hash of the header of the block until a small value is found. Casting this in our terms, the key is the hash of the previous block, which by itself depends on Bitcoin's genesis block, while the transactions received by the network, as well as the coinbase transaction, constitute the message. It is important to note that it is not possible to consider the key to be the coinbase transaction, as there is no guarantee it has any entropy when produced by an adversarial signer. To abstract the randomized nature of the signing procedure, which in the actual implementation is captured by the coinbase transaction, we hash *msg* together with a randomly generated string. This should be part of the signature in our SoW syntax since it is produced by the signing process and is necessary for verification. Similarly, the counter is also part of the signature produced by the signing process. $\mathtt{BSOW}$, a simplified version of the scheme described above with the transaction semantics omitted for simplicity, is presented in Figure 12.

Remark 6. In the Bitcoin implementation, the hash of the root of the Merkle tree of the transactions is not "*salted*." This means that if we consider the adversary to be non-uniform, she could get collisions for free in her advice string and use them to compute two SoWs at the cost of one. This would be problematic for our MU-TCMA security game. Thus, in order to strengthen the security of the scheme, we choose to also include the key in the hash of the message.

Algorithm 12 Bitcoin's SoW implementation on the ROM. H and G are modeled as ROs.

```
function PPub($1^\lambda$)
    return $1^\lambda$

function KeyGen($pp = 1^\lambda$)
    $vk \leftarrow \mathcal{U}_\lambda$
    return $vk$

function Sign($pp = 1^\lambda, vk, msg, h$)
    while $true$ do
        $\sigma_1 \leftarrow \mathcal{U}_\lambda$
        $dig \leftarrow G(vk, \sigma_1, msg)$
        for $\sigma_2 = 0^\lambda|_2$ to $1^\lambda|_2$ do
            if $(H(vk, dig, \sigma_2) < 2^\lambda - h)$ then
                return $(\sigma_1, \sigma_2)$

function Verify($pp = 1^\lambda, vk, msg, h, \sigma = (\sigma_1, \sigma_2)$)
    return $(H(vk, G(vk, \sigma_1, msg), \sigma_2) < 2^\lambda - h)$
```

We assume that H and G are idealized hash functions, i.e., our analysis is in the random oracle model.

Theorem 63. *If H, G are modeled as Random Oracles, then Algorithm 12 implements a SoW scheme, with hardness space $HP = [2^\lambda - \lambda]$, that is:*

- *correct;*
- *$O(\lambda)$-verifiable;*
- *$(t_{sign}, 1 - \left(\frac{h}{2^\lambda}\right)^{t_{sign}})$-successful, for any polynomially large t_{sign};*

- *run-time independent;*
- $((1+\sigma)(1-\frac{h}{2^\lambda}), e^{-\frac{\beta(h)t\sigma^2}{6}} + \mathsf{negl}(\lambda))$*-MU-TCMA secure w.r.t. any computationally unpredictable function family $\mathcal{F}$, for any* $\sigma \in (0,1)$.

Proof. Let $p_h = 1 - \frac{h}{2^\lambda}$ be the probability that a query to the random oracle returns a value less than $2^\lambda - h$, and let $q_{\mathcal{H}}$ be the number of queries the adversary makes to the RO. We consider each property in turn.

Correct. The probability that the Sign algorithm cannot find any signature for the given parameters is upper bounded by the probability that 2^λ different queries to the RO return a value greater or equal than $2^\lambda - h$. This probability is upper bounded by:

$$(\frac{h}{2^\lambda})^{\lambda 2^\lambda} \leq (\frac{2^\lambda - \lambda}{2^\lambda})^{2^\lambda} \leq (1 - \frac{\lambda}{2^\lambda})^{2^\lambda} \leq e^{-\lambda}$$

The correctness property follows.

MU-TCMA. Let $\ell = \beta(h)t$. W.l.o.g., assume $\ell \geq 1$, since if $\ell < 1$ there exists a trivial adversary that wins with probability 1. First, we show that for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{A}'$ that succeeds in winning $\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A}',\mathcal{F}}$ (Figure 4.2) with almost the same time complexity and probability that $\mathcal{A}$ wins, without using the signing oracle $\mathcal{S}$. $\mathcal{A}'$ is going to run $\mathcal{A}$ internally, and all calls made by $\mathcal{A}$ to $\mathcal{S}$ are going to be simulated, i.e., assuming $\mathcal{A}$ queries $\mathcal{S}$ with values (k, msg), $\mathcal{A}'$ will respond with some number t' sampled from the time distribution of $\mathcal{S}$ (t' can be efficiently sampled from a geometric distribution, since queries are i.i.d Bernoulli trials) and some random signature $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_2 < t'$. Moreover, it will sample a uniformly random element from $[2^\lambda - h - 1]$, and store all these information in some efficient data structure that allows for search in logarithmic time. Any calls made by the adversary afterwards to the RO that are related to (k, msg) will be answered accordingly: If $\mathcal{A}_2$ queries the RO with some string $k||G(k, \sigma_1, msg)||\sigma_2'$, where $\sigma_2' = \sigma_2$, then $\mathcal{A}'$ will respond with the stored value for the related $\mathcal{S}$ query that is less than $2^\lambda - h$. Otherwise if $\sigma_2' < \sigma_2$, if this is the first time this query is performed he samples uniformly a value from $[2^\lambda - h, 2^\lambda]$, store it, and return it to $\mathcal{A}$ to this and any subsequent identical query. Note, that since σ_1 is chosen at random and RO is unpredictable, the probability that $\mathcal{A}$ has queried the RO with a string of this format before querying $\mathcal{S}$ is negligible. Hence, the view of $\mathcal{A}$ in both experiments is computationally indistinguishable, and he will output ℓ valid SoWs with respect to the simulated view with the same probability that he wins in the real experiment.

We next have to show that $\mathcal{A}'$ can use the output of $\mathcal{A}$ to win in the real experiment. The only case this will not happen, is if the output of $\mathcal{A}$ contains a SoW related to the queries asked to (the simulated) oracle $\mathcal{S}$, and thus it does not correspond to a winning output for $\mathcal{A}'$, i.e., $\mathcal{A}'$ has set the value of this SoW to be small enough, while this does not necessarily hold for the actual RO. This implies, that there exists a SoW on the output of $\mathcal{A}$ of the form $\langle f, msg, (\sigma_1, \sigma_2)\rangle$ and a query $\langle (k, msg'), (\sigma_1', \sigma_2')\rangle$ on $\mathcal{S}$, such that $f(\Sigma, vk)||G(f(\Sigma, vk), \sigma_1, msg)||\sigma_2 = k||G(k, \sigma_1', msg')||\sigma_2'$. In order for this to be a winning output for $\mathcal{A}$, it must not correspond to the asked query, and thus it should hold that either $msg \neq msg'$ or $\sigma_1 \neq \sigma_1'$. This implies that the adversary has found a collision in G, which only happens with negligible probability in λ. Hence, $\mathcal{A}'$ will win $\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A}',\mathcal{F}}$ with the same probability (minus some negligible term in λ) as $\mathcal{A}$. Moreover, the overhead incurred to $\mathcal{A}'$'s running time will be only logarithmic on λ i.e. $\mathcal{A}'$ can simulate the t steps taken by $\mathcal{A}_2$ in time $t' = t \cdot (1 + \Omega(\log(\lambda)))$; he has to maintain a heap of $poly(\lambda)$ queries made to $\mathcal{S}$ and the RO. Importantly, note that in any case $\mathcal{A}'$ queries the real RO at most t times.

Next, we analyze the probability that $\mathcal{A}'$ wins in the MU-TCMA experiment. Let A_ℓ be the event where $\mathcal{A}'$ asks t queries the RO after receiving vk, and receives at least ℓ responses that have value less than $2^\lambda - h$. Let random variable X be equal to the number of these responses that are less than $2^\lambda - h$. Since the queries are i.i.d. Bernoulli random variables with probability of success p_h, we can use the Chernoff bound to bound the probability of A_ℓ. For any $\sigma \in (0,1)$, since $\ell = \beta t \geq (1+\sigma)p_h t$, it follows that:

$$\begin{aligned}\Pr[A_\ell] = \Pr[X \geq \ell] \leq & \Pr[X \geq (1+\sigma)p_h t] \\ = & \Pr[X \geq (1+\sigma)\mathbb{E}[X]] \leq e^{-\frac{\mathbb{E}[X]\sigma^2}{3}} \leq e^{-\frac{(1+\sigma)p_h t \sigma^2}{6}}.\end{aligned}$$

Let B be the event where $\mathcal{A}_2'$ outputs f, m, σ such that $f \in F_\lambda$ and there exists a query made to the random oracle by $\mathcal{A}_1'$ of the form $f(\Sigma, vk)||x||\sigma$, for some $x \in \{0,1\}^*$. We will show that B happens with only negligible probability in λ. For the sake of contradiction, assume that B happens with non-negligible probability. Then, we can use $\mathcal{A}'$ to break the computational unpredictability of $\mathcal{F}$. Let $\mathcal{A}'' = (\mathcal{A}_1'', \mathcal{A}_2'')$ be the attacker in the computational unpredictability game. $\mathcal{A}_1''$ on input $(1^\lambda, \Sigma, pp)$ will first run $\mathcal{A}_1'(1^\lambda, \Sigma, pp)$. It will output $st' = st||y$, where st is the output of $\mathcal{A}_1$, and y the prefix of a random query that $\mathcal{A}_1'$ made to the RO with length equal to the size of a verification key. Then, $\mathcal{A}_2''$ on input (st', vk) will run $\mathcal{A}_2'(1^\lambda, vk, st)$, until it halts and possibly outputs a number of SoWs. Since $\mathcal{A}'$ is a PPT algorithm, the number of queries made to the RO is at most polynomial in number. Hence, with non-negligible probability B will occur and y will be the prefix of the RO query that matches the key of the SoW output by $\mathcal{A}'$. This violates the computational unpredictability property, and hence B only occurs with negligible probability.

Let C be the event where the adversary wins and outputs two distinct SoWs that correspond to the same query to the RO. This implies that the adversary can find a collision on G. In time $L = t' + t_{\mathsf{pre}}$ polynomial in λ, the probability that $\mathcal{A}'$ finds a collision is $\binom{L}{2}2^{-\lambda+1} = e^{-\Omega(\lambda)} = \mathsf{negl}(\lambda)$.

Finally, note that if A_ℓ, B, C do not occur, it is implied that $\mathcal{A}'$ will lose in the MU-TCMA experiment. Thus:

$$\begin{aligned}
\Pr[\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A},\mathcal{F}}(1^\lambda, h, \ell) = 1] &= \Pr[\mathsf{Exp}^{\text{MU-TCMA}}_{\mathcal{A}',\mathcal{F}}(1^\lambda, h, \ell) = 1] \\
&\le \Pr[A_\ell \vee B \vee C] \\
&\le \Pr[A_\ell] + \Pr[B] + \Pr[C] \\
&\le \Pr[A_\ell] + \mathsf{negl}(\lambda) \\
&\le e^{-\frac{(1+\sigma)p_h t \sigma^2}{6}} + \mathsf{negl}(\lambda)
\end{aligned}$$

where we have used the union bound for the third inequality.

Now, by our initial argument, it follows that if there exists an adversary $\mathcal{A}$ that is t bounded and breaks the MU-TCMA property with probability greater than $e^{-\frac{(1+\sigma)p_h t \sigma^2}{6}} + \mu(\lambda)$, where $\mu(\cdot)$ is a non-negligible function, then there exists an adversary $\mathcal{A}'$ that makes at most t queries to the RO, and no queries to $\mathcal{S}$, and breaks the MU-CMA property with the same probability. This is a contradiction to the inequality proven above, and the desired property follows.

Verifiability. Assuming H and G take constant time, verification takes time $c_{\mathsf{ver}}\lambda$, for some small constant c_{ver} which can be easily computed by careful inspection of the verification protocol.

Successful. For any $\lambda \in \mathbb{N}$, any polynomially large $t \in \mathbb{N}$, $pp \in PP$, $vk \in K$, $msg \in M$ and $h \in HP$, it follows that[11]:

$$\begin{aligned}
\Pr\left[\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, msg, h) \le t\right] &\ge 1 - \Pr\left[\mathsf{Steps}_{\mathsf{Sign}}(pp, vk, msg, h) > t\right] \\
&\ge 1 - (1 - p_h)^t
\end{aligned}$$

Independence. Fix some polynomial $p'(\cdot)$, λ, h, and some $pp \in PP$, $((vk_i, m_i))_{i \in [p'(\lambda)]} \in (K \times M)^{p'(\lambda)}$. Let $\{Y_i\}_{i \in [p'(\lambda)]}$ be the same as $\{X_i\}_{i \in [p'(\lambda)]} = \{\mathsf{Steps}_{\mathsf{Sign}}(pp, vk_i, m_i, h)\}_{i \in [p'(\lambda)]}$

[11]For simplicity, here we ignore the cost of computing σ_1 and dig, which in any real application will be a lot smaller than the cost of the main loop of Sign.

with the only difference being that the random oracles H, G are replaced with the uniformly random sampling of an element in their range, i.e., every time Sign is called and H or G are invoked, a random output is returned. Note, that $\{Y_i\}_i$ does not depend on $pp, ((vk_i, m_i))_i$, and only depends on the coins of the sampler, as the independence property requires. Moreover, the random variables in $\{Y_i\}_i$ are mutually independent, since their output only depends on their own local coins.

Regarding the second property, let E be the event that all σ_1 sampled are different among all the invocations of Sign, and that no collisions occurs in G. Note, that since $p'(\cdot)$ is a polynomial, E happens with overwhelming probability in λ. Moreover, it holds that $\Pr[\{X_i\}_i = z|E]$ is equal to $\Pr[\{Y_i\}_i = z|E]$, for any z, since the random oracle behaves exactly as the sampler we have replaced it with conditioned on E, as no query is repeated. Therefore, it holds that for any $z \in \mathbb{N}^{p'(\lambda)}$

$$\begin{aligned}
&\Pr[\{X_i\}_i = z] - \Pr[\{Y_i\}_i = z] = \\
&= \Pr[\{X_i\}_i = z|E]\Pr[E] + \Pr[\{X_i\}_i = z|\neg E]\Pr[\neg E] \\
&\quad - \Pr[\{Y_i\}_i = z|E]\Pr[E] - \Pr[\{Y_i\}_i = z|\neg E]\Pr[\neg E] \\
&= (\Pr[\{X_i\}_i = z|\neg E] - \Pr[\{Y_i\}_i = z|\neg E])\Pr[\neg E]
\end{aligned}$$

It follows that the two distributions are $\mathsf{negl}(\lambda)$-close, as shown next:

$$\begin{aligned}
2\Delta[\{X_i\}_i, \{Y_i\}_i] &= \sum_z |\Pr[\{X_i\}_i = z] - \Pr[\{Y_i\}_i = z]| \\
&\leq \sum_z |(\Pr[\{X_i\}_i = z|\neg E] - \Pr[\{Y_i\}_i = z|\neg E])\Pr[\neg E]| \\
&\leq \Pr[\neg E] \sum_z |(\Pr[\{X_i\}_i = z|\neg E] - \Pr[\{Y_i\}_i = z|\neg E])| \\
&\leq \mathsf{negl}(\lambda)(\sum_z \Pr[\{X_i\}_i = z|\neg E] + \sum_z \Pr[\{Y_i\}_i = z|\neg E]) \leq \mathsf{negl}(\lambda)
\end{aligned}$$

The last inequality follows from the fact that each of the sums should be less or equal to 1, as the events described are disjoint, and their union covers the entire sample space. □

SoW in the $\mathcal{F}_{\text{TREE}}$ model. Next, we argue that we can use the $\mathcal{F}^p_{\text{TREE}}$ functionality from [PSS17] to realize a secure SoW scheme.

We start by reciting the $\mathcal{F}^p_{\text{TREE}}$ description. $\mathcal{F}^p_{\text{TREE}}$ is parametrized by some hardness parameter p and keeps track of records on a tree. Initially, the tree contains $\perp$. On invocation of $extend((m_1, \ldots, m_{\ell-1}), m)$, $\mathcal{F}^p_{\text{TREE}}$ checks if $(m_1, \ldots, m_{\ell-1})$ is a valid path on the tree, and if yes with probability p extends this path by m and returns 1.

Otherwise, returns 0. On invocation of $ver(m_1,\dots,m_\ell)$, $\mathcal{F}^p_{\text{TREE}}$ returns 1 if $m_1,\dots,m_\ell$ is a valid path recorded earlier, and 0 otherwise. For simplicity, here we assume that *extend* and *ver* take one computational step each.

The SoW protocol is constructed in a similar way as in the ROM; we only have to replace the "$H(\dots) < 2^\lambda - h$" checks in Sign and Verify, with invocations to *extend* and *ver* on $\mathcal{F}^{(2^\lambda-h)/2^\lambda}_{\text{TREE}}$, respectively (see Algorithm 13). We consider a slightly stronger version of $\mathcal{F}_{\text{TREE}}$, where the hardness parameter p is a parameter of the function calls; SoW schemes allow for different hardness levels, while the original $\mathcal{F}_{\text{TREE}}$ functionality has a fixed hardness level.

Algorithm 13 Bitcoin's SoW implementation based on functionality $\mathcal{F}_{\text{TREE}}$. We omit functions PPub, KeyGen which are exactly the same as in Algorithm 12, while differences in Sign, Verify are depicted in red.

```
1: function Sign($pp = (1^\lambda), vk, msg, h$)
2:     while $true$ do
3:         $\sigma_1 \leftarrow \mathcal{U}_\lambda$
4:         for $\sigma_2 = 0^\lambda|_2$ to $1^\lambda|_2$ do
5:             if $(\mathcal{F}^{(2^\lambda-h)/2^\lambda}_{\text{TREE}}.extend(\perp, vk||\sigma_1||msg||\sigma_2) = 1)$ then
6:                 return $(\sigma_1, \sigma_2)$
7:
8: function Verify($pp = (1^\lambda), vk, msg, h, \sigma = (\sigma_1, \sigma_2)$)
9:     return $\mathcal{F}^{(2^\lambda-h)/2^\lambda}_{\text{TREE}}.ver(\perp, vk||\sigma_1||msg||\sigma_2)$
```

The proof that Algorithm 13 implements a secure SoW scheme follows the same arguments as the proof of Theorem 63. We omit the details.

Corollary 64. *If $\mathcal{G}$ is a collision resistant hash function family, and we are in the $\mathcal{F}_{\text{TREE}}$-hybrid world, then Algorithm 13 implements a SoW scheme as in Theorem 63.*

Comparison with the results of [GKL15]. Next, we argue that if `BSOW` is used to instantiate the SoW scheme of Theorem 61, the resulting ledger will achieve similar parameters to the ones obtained in the analysis of [GKL15]. In order to be able to use this theorem, Assumptions 2 and 3 must be satisfied. Note, that we still have to assume that honest parties collectively take more steps than the adversary at each round, as this is an assumption that has to do mainly[12] with our computational model, and not the

[12]Mainly, because $t'_{\mathcal{H}}$ and $t'_{\mathcal{A}}$ also depend on t_{bb}, θ and t_{ver}.

SoW scheme.

Theorem 65. *Assume that $(n-t)t'_{\mathcal{H}}(1-\delta_{\mathsf{Steps}}) \geq t'_{\mathcal{A}}$, for some $\delta_{\mathsf{Steps}} \in (0,1)$. If G,H are modeled as a Random Oracle, then there exists an $h \in HP = [2^{\lambda}-\lambda]$, such that Algorithm 12 (*`BSOW`*) satisfies Assumptions 2 and 3.*

Proof. We have already proved in Theorem 63 that `BSOW` satisfies all properties described in Assumption 2. It remains to argue about Assumption 3.

Set $\delta = \delta_{\mathsf{Steps}}/4$ and $\delta_{\mathsf{SoW}} = \delta_{\mathsf{Steps}}/2$. It follows that $\frac{\delta_{\mathsf{Steps}}-\delta_{\mathsf{SoW}}}{2} \geq \delta$. For simplicity, assume that $\sigma \ll 1$ in the definition of $\beta(h)$. Let $p_h = 1 - h/2^{\lambda}$. It holds that:

$$\begin{aligned}\alpha(h) \geq (1-\delta_{\mathsf{SoW}})\beta(h)t'_{\mathcal{H}} &\Leftrightarrow 1-(1-p_h)^{t'_{\mathcal{H}}} \geq (1-\delta_{\mathsf{SoW}})p_h t'_{\mathcal{H}} \\ &\Leftrightarrow (1-p_h)^{t'_{\mathcal{H}}} + (1-\delta_{\mathsf{SoW}})p_h t'_{\mathcal{H}} \leq 1 \\ &\Leftarrow e^{-p_h t'_{\mathcal{H}}} + (1-\delta_{\mathsf{SoW}})p_h t'_{\mathcal{H}} \leq 1\end{aligned}$$

Now, let $f(u) = e^{-u} + (1-\delta_{\mathsf{SoW}})u$. It holds that $f(0)=1$, $\frac{df}{du} = -e^{-u} + (1-\delta_{\mathsf{SoW}})$, $\frac{df}{du}(\ln((1-\delta_{\mathsf{SoW}})^{-1})) = 0$, and $\frac{df}{du}$ is strictly increasing as u grows. Since $\ln((1-\delta_{\mathsf{SoW}})^{-1}) > 0$, it is implied that f is decreasing in $(0, \ln((1-\delta_{\mathsf{SoW}})^{-1})]$, and thus for any $u < \ln((1-\delta_{\mathsf{SoW}})^{-1})$, $f(u) < 1$. Hence, for $p_h \leq \frac{\ln((1-\delta_{\mathsf{SoW}})^{-1})}{t'_{\mathcal{H}}}$, it follows that $\alpha(h) \geq (1-\delta_{\mathsf{SoW}})\beta t'_{\mathcal{H}}$, for any $t'_{\mathcal{H}}$.

Hence, if we set $p_h \leq \min\{\frac{\ln((1-\delta_{\mathsf{SoW}})^{-1})}{t'_{\mathcal{H}}}, \delta/(t'_{\mathcal{A}} + nt_{\mathcal{H}})\}$, the preconditions of Theorem 61 for `BSOW` are satisfied. Note, that we assume that λ is large enough so that (i) $\lambda > 9/\delta$, (ii) there exists an $h \in HP$ so that p_h satisfies the inequality above. □

A second issue is that the analysis of Theorem 61 is performed under the assumption that we are in the standard model, and thus the outputs of the routines of the SoW scheme depend only on the parameters they are invoked with and the coins independently sampled by the routines. This condition is violated for `BSOW`, since its outputs also depend on the shared random coins of the RO. To address this issue we adopt slightly different versions of the SoW security definitions that are specific to the idealized model used. In more detail, the idealized-model complete, successful and almost-runtime independent properties should still hold if we fix any polynomial number of entries of the RO to arbitrary values before the relevant experiments are performed. These properties can be proven in the same way as in Theorem 63, since the probability that an honestly performed RO query coincides with the fixed set of RO entries is negligible. Hence, the following corollary is implied from Theorem 61.

Corollary 66. *Assume protocol $\Pi_{\mathsf{PL}}^{\mathtt{SoW}}$ uses the SoW scheme of Algorithm 12, and that $(n-t)t'_{\mathcal{H}}(1-\delta_{\mathsf{Steps}}) \geq t'_{\mathcal{A}}$, for some $\delta_{\mathsf{Steps}} \in (0,1)$. Then, there exists a hardness parameter h, such that protocol $\Pi_{\mathsf{PL}}^{\mathtt{SoW}}$ implements a robust public transaction ledger with parameters $u = \frac{2k}{(1-\frac{\delta}{4})f}$ and $k \geq \frac{2\lambda}{\gamma\delta}$ in the ROM, except with negligible probability in λ.*

Next, we argue that the parameters obtained from this theorem are comparable to those obtained in [GKL15]. We focus on the γ, f quantities introduced in the previous subsection. These quantities are important since γ is related to the rate of uniquely successful rounds, while f is related to how fast blocks are produced. Replacing with the parameters proved for `BPOW`, for $\sigma \ll 1$, γ and f are equal to:

$$\gamma = (n-t)\cdot(1-\left(\frac{h}{2\lambda}\right)^{t'_{\mathcal{H}}})\cdot(1-t_{\mathcal{H}}(1-\frac{h}{2\lambda}))^{n-1}, f = 1-\left(\frac{h}{2\lambda}\right)^{t'_{\mathcal{H}}(n-t)}$$

Both γ and f appear in [GKL15] and are well approximated by our results. Hence, $\Pi_{\mathsf{PL}}^{\mathtt{BSoW}}$ implements a robust transaction ledger with overwhelming probability in λ and with bounds comparable to those in [GKL15].

Chapter 5

Blockchain Security and Iterated Search Problems

5.1 Introduction

Blockchain protocols, introduced by Nakamoto [Nak08], 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 [DN93] initially as a spam protection mechanism, and subsequently found applications in other domains such as Sybil attack resilience [Dou02a] and denial of service protection [JB99, Bac02], prior to their application to the domain of distributed consensus hinted at early on by Aspnes *et al.* [AJK05].

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

Despite the evolution of our understanding of the PoW primitive, as exemplified in recent works (e.g., [AT17, BRSV18, BGJ$^+$16]), there has been no definitive analysis of the primitive in the context of blockchain protocol security. Intuitively, PoWs are useful in the consensus setting because they make message passing (moderately) hard and hence generate stochastic opportunities for the parties running the protocol to unify their view of the current state of the system. This fundamentally relies on an assumption about the aggregate computational power of the honest parties, but not on their actual number, in relation to the computational power of the parties that may deviate from the protocol ("Byzantine")—a hallmark of the peer-to-peer setting where Bitcoin is designed for. Despite the fact that the Bitcoin blockchain has been analyzed formally [GKL15, PSS17, GKL17, BMTZ17], the required PoW properties have not been identified and the analysis has been carried out in the random oracle (RO) model [BR93]. The same is true for a wide variety of other protocols in the space, including [AD15, KMS14, GKLP18].

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.* [CGH04] showed that there exist implementations of signatures and encryption schemes that are secure in the RO model but insecure for any implementation of the RO in the standard model; Nielsen [Nie02] proved that efficient non-committing encryption has no instantiation in the standard model but a straightforward implementation in the RO model, while Goldwasser and Kalai [GK03] showed that the Fiat-Shamir heuristic [FS86] does not necessarily imply a secure digital signature, which is in contrast with the result by Pointcheval and Stern [PS96] in the RO model. It follows that it is critical to discover security arguments for blockchain protocols that do not rely on the RO model. Note that we are looking for arguments as opposed to proofs since it is easy to observe that some computational assumption would still be needed for deriving the security of a blockchain protocol (recall that blockchain security cannot be inferred information theoretically as it fundamentally requires at minimum the collision resistance of the underlying hash function). Further, the formalization of non-idealized, concrete hash function assumptions sufficient to prove security of Bitcoin and related protocols has been identified as an important open question [CCRR18].

Towards identifying "good" cryptographic assumptions, Naor [Nao03] introduced a framework for classifying cryptographic hardness assumptions and identified the notion of a *falsifiable assumption* which was further studied by Gentry and Wichs [GW11]. In the latter formulation, a falsifiable assumption is one that can be expressed as a game between an adversary and a challenger, and where the challenger can efficiently determine (and output) when the adversary wins the game. As a main result they show that succinct non-interactive arguments (SNARGs), which exist in the RO model, are impossible to construct in the standard model under any falsifiable assumption. The above highlights the main open question that motivates our work:

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

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

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

We then note that a related but distinct notion of hardness, *sequential* (i.e., non-parallelizable) iterated hardness, implies the existence of a hard ISP. This notion has been considered as early as [RSW96], mainly in the domains of timed-release cryptography [BN00] and protocol fairness [GMPY11], and recently formalized in [BBBF18] under the term iterated sequential functions. In addition, a number of candidate hard problems have been proposed, including squaring a group element of a composite moduli [RSW96], hashing, and computing the modular square root of an element on a prime order group [LW15]. Nevertheless, and importantily, we prove that the blockchain protocol which is based on the ISP that results from the underlying iterated sequential problem is *insecure*. The fundamental issue is that it does not allow for parallelization, which, as our negative result shows, is crucial for proving the security of any (Bitcoin-like) blockchain protocol. This also highlights that, even though necessary, iterated hardness is by itself insufficient to prove the security of this type of protocols.

Motivated by the above we identify a set of additional properties that epitomize what we call an *enhanced* ISP, from which we can provably derive the security of a blockchain protocol. The enhanced properties are as follows. First, an ISP is (t,α)-*successful* when the number of steps of the solving algorithm is below t with probability at least α. The ISP is *next-problem simulatable* if the output of the successor algorithm applied on a witness w corresponding to an instance x can be simulated independently of x and the same is the case for the running time of the solver. Finally, an ISP is *witness-malleable* if, given a witness w for a problem instance x, it is possible to sample an alternative witness whose resulting distribution via the successor algorithm is computationally indistinguishable to the output of the successor over a random witness produced by the solving algorithm.

Armed with the above definitions we show a novel blockchain protocol whose security can be reduced to the hardness of the underlying enhanced ISP. We note that the main technical difficulty of our blockchain security proof is to construct a reduction that breaks the underlying iterated hardness assumption given a common-prefix attack to the blockchain protocol. This is achieved by taking advantage of zero-knowledge proofs with efficient simulation and the ability of the reduction to efficiently extract a sequence of iterated witnesses despite the fact that the attacker may not produce consecutive blocks.

We perform our analysis in the static setting with synchronous rounds as in [GKL15], 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. We conjecture that our results can be extended to the Δ-synchronous setting of

[PSS17].

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

Related work. The notion of "correlation intractability" (CI) [CCRR18] bears some ressemblance to our ISP notion. There are, however, two main differences. First, CI only bounds the success probability in solving a single challenge, while our notion talks about (and fundamentally requires) multiple instances. Second, while CI talks about any sparse relation, our definition talks about a single non-sparse relation. Exploring further connections between the two notions is an interesting direction.

In Chapter 4, the question we pose here is partially answered, as a set of sufficient falsifiable conditions for implementing a ledger and the consensus problem in the permissionless setting are presented. However, the only known implementation of "signatures of work" (SoW), the concept introduced there as the basic underlying assumption, is in the RO model. It is unknown (and an interesting open question) whether SoWs can be realized under simpler falsifiable assumptions like the ones we consider here.

Regarding our hash-based ISP construction, the concept of collision resistant randomness extractors has been considered before: Dodis in [Dod05] proposes a construction of a (statistical) collision-resistant randomness extractor, assuming the existence of claw-free permutations. In order for collision resistance to hold, it is required that among different invocations of the extractor, the seed is fixed. Naturally, randomness extraction is only guaranteed for sources which are independent of the seed. This is problematic in our case, since the seed is part of the description of the hash function and should be public. Thus, we cannot ensure that the adversarially influenced source distribution will be independent of the seed, and cannot use this construction in our setting.

Zero-knowledge proofs have been previously considered in the study of cryptocurrencies mainly to guarantee privacy [KKKZ18, BCG$^+$14]. Here, we make use of

NIZKs to hide block witnesses, i.e., in the consensus layer. A concern with NIZKs in the permissionless setting is how to generate the trusted CRS needed for such proofs without any party learning the simulator trapdoor guaranteed to exist by ZK security [GKM$^+$18, BSCG$^+$15]. We consider strengthening the CRS generation procedure with techniques developed in this line of work to be an interesting open question.

We remind the reader, that a list of previous attempts to define the security of PoW schemes, as well as their inadequacy for our research question, is also presented in Section 4.1.

Limitations and directions for future research. We leave as an open question the extension of our results to the dynamic setting of [GKL17], as well as matching the 50% threshold on adversarial computational power of the Bitcoin blockchain which can be shown in the RO model. As in Chapter 4, reducing the hardness of our hash-based ISP construction to better studied assumptions remains an interesting open problem. An equally important direction is the study of the ISP-hardness of our construction from a cryptanalytic point of view.

Organization of the chapter. The basic computational model and definitions used by our constructions are presented in Section 5.2. 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 5.3. 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 5.4.

5.2 Preliminaries

In this chapter we follow the concrete approach discussed in Section 4.2, and perform our analysis in the revised permissionless model presented in Section 4.4.1. We let λ denote the security parameter. We consider static adversaries, but we conjecture that our results hold also for adaptive adversaries if we allow erasures.

Robust public transaction ledgers. We adopt a slightly stronger version of the Liveness property, introduced in [PSS17]. The main difference from Definition 5, is that the transactions that are guaranteed to become stable after the u round interval, are the ones provided to at least one of the honest parties at some round, as opposed to all of the honest parties and every round.

Definition 67. A protocol Π implements a *robust public transaction ledger* in the synchronous setting, if after some round r_0 the following two properties are satisfied:

- *Persistence:* as in Definition 5.
- *Liveness:* Parameterized by $u,k \in \mathbb{N}$ (the "wait time" and "depth" parameters, resp.), for every u consecutive rounds, there exists a round and an honest party, such that the transactions given as input to that party at this round that are either (i) issued by Txgen, or (ii) are neutral, will be reported by all honest parties as k-stable at the end of this round interval.

5.3 Necessary Conditions for PoW-based Blockchain Protocols

In this section we study the necessary conditions for the security of PoW-based blockchain protocols, focusing on (an abstraction of) Bitcoin. Our aim will be to distill what are the necessary properties the underlying moderately hard problem should satisfy. In the Bitcoin protocol this problem is related to a real-world hash function, namely, SHA-256. We abstract the problem in Section 5.3.1, where we introduce the notion of *hard iterated search problems*. Then, in Section 5.3.2 we describe a generalized Bitcoin-like protocol based on such problems, and show that both the existence of such problems is necessary to prove the security of Bitcoin, and yet it is not sufficient.

5.3.1 Iterated search problems

Next, we introduce a class of problems expressive enough to abstract Bitcoin's underlying computational problem. The straightforward requirements are the ability to find a witness for a problem statement, and to verify that a witness is correct, matching the block mining and block verification procedures, respectively. In addition, we require the ability to generate a new problem statement from a valid statement/witness pair. This captures the fact that in Bitcoin the problem that a miner solves depends on a previous block (i.e., a statement/witness pair). This concept has appeared before in the study of iterated sequential functions, whose name we borrow from. Syntactically, the key difference here is that in each iteration we are not evaluating a function, but instead we are solving a search problem with possibly many witnesses. Moreover, we will later argue that iterated sequential functions are *not* the correct abstractions for the underlying computational problem, as they allow for an attack against the protocol. We proceed to give a formal definition.

An *iterated search problem* (ISP) I specifies a collection $(I_\lambda)_{\lambda \in \mathbb{N}}$ of distributions.[1] For every value of the security parameter $\lambda \geq 0$, I_λ is a probability distribution of *instance descriptions*. An instance description Λ specifies (1) finite, non-empty sets X, W, and (2) a binary relation $R \subset X \times W$. We write $\Lambda[X, W, R]$ to indicate that the instance Λ specifies X, W and R as above.

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

- *Verification* algorithm $V_\Lambda(x, w)$: A deterministic algorithm that takes as input a problem statement x, and a witness w and outputs 1 if $(x, w) \in R$, and 0 otherwise.
- *Successor* algorithm $S_\Lambda(x, w)$: A deterministic algorithm that takes as input a problem statement[2] x, and a valid witness w and outputs a new instance $x' \in X$.
- *Solving* algorithm $M_\Lambda(x, k)$: A probabilistic algorithm that takes as input a problem statement x, and a number $k \in \mathbb{N}^+$ and outputs a sequence of k witnesses $(w_i)_{i \in [k]}$.

In the sequel, we will omit writing Λ as a parameter of V, S, M when it is clear from the context. As an example, we present Bitcoin's underlying computational problem captured as an ISP.

Construction 1. Let T be a protocol parameter that has to do with how hard it is to solve a problem instance.[3] Then:

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

[1] Here we follow the notation used to define *subset membership problems* in [CS02]. We remark that no other connection exists between the two works.

[2] We could formalize S more generally, to take as input a sequence of problem statements. However, for our exposition the current formulation suffices. Note, that a more general definition would be needed for the variable difficulty case [GKL17], which we do not study here, where the next block's difficulty depends on the whole chain.

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

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

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

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

We assume that M is *correct*, i.e., for $(w_i)_{i\in[k]} \leftarrow M(x,k)$, it holds that $(x,(w_i)_{i\in[k]}) \in R$ with overwhelming probability in λ.

Finally, in the same spirit as Boneh *et al.* [BBBF18]'s definition of an iterated sequential function (cf. Definition 117), we define the notion of a *hard* iterated search problem. Our definition is parameterized by t, δ and k_0, all functions of λ which we omit for brevity. Unlike the former definition, we take in account the *total* number of steps instead of only the sequential ones, and we require the error probability to be negligible after at least k_0 witnesses have been found instead of one. In that sense, our notion relaxes the strict convergence criterion of [BBBF18]. Finally, we note that is possible to define the error probabilities concretely. For simplicity, we chose not to do so, as the current formulation still suffices for our goals.

Definition 68. An ISP $I = (V,M,S)$ is (t,δ,k_0)-*hard* iff it holds that:

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

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

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

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

Remark 7. While in this section we focus on the hardness of an ISP, later we are going to require more properties for it to be useful, including fast verification and the ability to parallelize the solver. If we did not require any extra properties, we could design trivially hard ISPs, for example based on the hardness of outputting long sequences of 1 in the output register.

5.3.2 An ISP-based Bitcoin protocol

Next, we describe an adaptation of the Bitcoin public ledger protocol [GKL15] with respect to an ISP $I = (M, V, S)$; we call the resulting protocol $\Pi^{\mathsf{isp}}_{\mathsf{PL}}(I)$. We give a high-level overview of the protocol, mostly focusing on the points where it differs from that in [GKL15]; we refer the reader to [GKL15] or Section 4.4.2 for a detailed exposition of the original protocol.

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

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

Remark 8. In the ISP definition, we do not explicitly model the ability of the solver M to encode information in the witnesses it computes. The reason for this is to better highlight the hardness properties of the ISP, instead of its use as an encoding mechanism. However, in order for Bitcoin to implement a transaction ledger, it is necessary that M implements such a mechanism. Otherwise, honest parties will not be able to add to the chain the transactions given to them by the environment, and the Liveness property would not hold. Hence, to keep the ISP definition uncluttered, and still be able to argue about the necessary hardness conditions it should satisfy, we opt for a weak encoding property for I: We assume that M encodes a random neutral transaction on the witness it outputs. More formally, we assume that for any input $x \in X$, the contents of block $\langle x, M(x, 1) \rangle$ extracted by $R(\cdot)$, are distributed uniformly on the neutral transaction space (remember that this space is super-polynomial in size). This corresponds to an execution of the Bitcoin backbone protocol where honest parties only generate random transactions locally, e.g., coinbase transactions. This change is w.l.o.g., as the necessary conditions derived for this restricted scenario should also hold in the general case.

5.3.3 Iterated hardness is necessary

We now analyze the necessary conditions that Bitcoin's underlying ISP should satisfy in order for Bitcoin to implement a ledger. In particular, we show that the ISP used in Bitcoin must be hard (Definition 68). In more detail, we prove that no adversary, that is suitably computationally bounded, should be able to find witnesses at a rate much higher than that of the honest parties. Other, somewhat secondary properties are necessary to state our subsequent result. We start with those.

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

Definition 69. An ISP $I = (V, M, S)$ is *collision resistant* iff for all PPT $\mathcal{A}$ and $\lambda \in \mathbb{N}$, it holds that

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

Additionally, we assume that there exists an upper bound t_{ver} on the time it takes to evaluate the verification algorithm V, such that $t_{\mathsf{ver}} \cdot \theta$ is insignificant compared to $t_{\mathcal{H}}$, where θ is the upper bound on the messages send per round and $t_{\mathcal{H}}$ is the number of steps each honest party takes per round. Intuitively, the time to verify that a witness is valid, must be a lot smaller than the available time honest parties have to compute one. 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 I encodes random neutral transactions, is collision resistant, and $t_{\mathsf{ver}} \cdot \theta$ is insignificant compared to $t_{\mathcal{H}}$. We are now ready to state the necessary hardness condition regarding ISPs. The main idea is that if I is not a hard-ISP for appropriate parameters, any adversary whose computational power is a constant fraction of that of the honest parties can extend its private chain faster than the rate at which honest parties extend their own chains. Hence, it can create a long enough fork and break Persistence.

Theorem 70. *Let $n, t, t_{\mathcal{H}}, t_{\mathcal{A}}$ such that $t_{\mathcal{A}} = c \cdot (n-t)t_{\mathcal{H}}$, for some $c \in (\frac{1}{n-t}, 1)$. If $\Pi_{\mathsf{PL}}^{\mathsf{isp}}(I)$ satisfies Persistence and Liveness with parameters u, k, where $I = (V, M, S)$ is an ISP, then there exists an ISP $I' = (V, M', S)$ that is $(\frac{u}{k} \cdot (n-t)t_{\mathcal{H}}, 1 - \frac{c}{2u}, 2k+2)$-hard.*

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

Let $\delta = 1 - \frac{c}{2u}$, and, for the sake of contradiction, assume that I' is not $(\frac{u}{k} \cdot (n-t)t_{\mathcal{H}}, \delta, 2k+2)$-hard. First, we will show that the first part of Definition 68 is satisfied. Assume an adversary in the Bitcoin execution that stays inactive. Due to Liveness, every u rounds honest parties add a new transaction k blocks deep in some chain. Repeating the same argument $\frac{m}{k}$ times, and since honest parties extend always the longest chain, it follows that after $\frac{m}{k} \cdot u$ rounds honest parties will have added m new blocks to the blockchain with overwhelming probability on λ. Equivalently, M' computes a sequence of m consecutive blocks in $m \cdot \frac{u}{k} \cdot (n-t)t_{\mathcal{H}}$ steps ($\frac{m}{k} \cdot u$ rounds), for any polynomially large $m \geq 2k+2$. Hence, the first part of the hardness definition is satisfied. From our assumption, it follows that the second part should not hold. This implies that there exists an adversary $\mathcal{A}$ that for infinitely many λ and $k_1 \geq 2k+2$ computes more blocks that the hardness definition allows, with non-negligible probability. We are going to use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ that creates a long enough chain privately, and breaks Persistence.

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

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

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

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

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

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

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

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

It is easy to show now, that if E_4 happens, Persistence breaks. We take two cases: In the first one, there exists at least one honest party that has a smaller chain than the one computed by $\mathcal{A}''$. It follows that in the next round, that this party will receive the chain of $\mathcal{A}''$, it will declare a transaction that no other honest party has in its chain as stable, which breaks Persistence. In the second case, all honest parties compute chains of exactly the same size as $\mathcal{A}''$. This implies, that if you take any two of them, they will have declared transactions as stable, that are not part of the other party's chain. Hence, $\mathcal{A}''$ breaks Persistence with non-negligible probability, which is a contradiction. The theorem follows.

$\square$

Note that as c approaches 1, the ISP should be secure against stronger adversaries. This is in line with our intuition: The better the security of the resulting protocol is, the harder the requirements from the ISP are. One reason that our bounds may seem rather weak, is that the assumptions about the distribution of blocks produced by honest parties are weak, e.g., some party may be able to compute multiple blocks per round, with some non-negligible probability. If we further assume, that block production is strongly concentrated, we can establish better bounds about the hardness of the resulting ISP. We note that the ISP construction we propose in Section 5.4.3 satisfies this property. In any case, our theorem is sufficient to ensure at least the existence of hard ISPs.

With foresight, we conclude this subsection by stating a stronger hardness property that allows the adversary some precomputation time. This will be useful in the next section, where we present and prove secure a protocol based on an enhanced ISP. We note, that a more convoluted version of this property, where the problem statement x is sampled from a distribution that has to do with the solving algorithm M, can be proven necessary in a similar way as Theorem 70; our results in Section 5.4 apply for both definitions.

Definition 71. An ISP $I = (V, M, S)$ is (t, δ, k_0)-*hard against precomputation* iff I is (t, δ, k_0)-hard (Definition 68) and for any PPT RAM $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\lambda \in \mathbb{N}$, and all polynomially large $k \geq k_0$, it holds that

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

Remark 9. Our analysis, with minor differences, also applies to other well-known PoW-based blockchain protocols. For example, in the GHOST protocol [SZ15] parties choose their chain by successively picking the "heaviest" subtree of blocks, starting from the genesis block. By similar arguments as above, it is implied that it is necessary for the ISP to be hard with respect to an upper bound on the rate at which the honest parties generate blocks (cf. the chain growth rate in Bitcoin). While this may sound as an improvement, it opens up the space for other attacks against the ISP; for example, if computing a new witness for a specific statement becomes easy after computing the first one, the adversary can create a large enough "tree" of blocks and break Persistence essentially at the cost of computing a single witness. This is not the case for Bitcoin, that can withstand such an attack.

Remark 10. The notion of a hard-ISP is related to the MU-TCMA property of a SoW scheme, as presented in Chapter 4. Specifically, assuming a SoW scheme that satisfies Assumptions 2 and 3 with respect to the tampering class defined in Lemma 47, one can define a hard-ISP. However, as we will discuss later in Section 5.4, the other direction does not necessarily hold, as a hard-ISP may be malleable in the sense that despite being hard for the adversary to compute a chain of witnesses, but may be easy to compute several witnesses for a single problem statement if he already knows one.

5.3.4 Iterated sequential functions are not sufficient

In this section we explore the relation of iterated sequential functions (cf. Definition 117) to ISPs and the Bitcoin protocol. On the bright side, we show that iterated sequential functions imply the existence of hard ISPs. Then, we deal with the question of whether Bitcoin can be solely based on such functions, and show that this is not the case.

In more detail, we show that the existence of iterated sequential functions that can be evaluated using a constant number of processors, implies that of hard ISPs, for appropriate parameters. Most interestingly, if we assume the existence of a $(\hat{t}, \mathsf{negl}(\lambda))$-ISF where the standard solver requires very small parallelization, as it is assumed for SHA-256 [BBBF18], then the resulting ISP is $(\hat{t}, \mathsf{negl}(\lambda), 1)$-hard, i.e., almost optimally hard.

Lemma 72. *If there exist a $(\hat{t}, \delta)$-iterated sequential function, which can be evaluated in time $\hat{t} \in poly(\lambda)$ using $c = O(1)$ parallel processors, then there exists a $(c \cdot \hat{t}, 1 - \frac{1-\delta}{c}, 1)$-hard ISP.*

Proof. Let $\{f_\lambda : X_\lambda \to X_\lambda\}_{\lambda \in \mathbb{N}}$ be a $(\hat{t}, \delta)$-iterated sequential function. Let I be the following ISP: $R_\lambda(x, w) = \{(x, f_\lambda(x)) | x \in X_\lambda\}$, S_λ is equal to the identity function, $I_\lambda = \{[X_\lambda, X_\lambda, R_\lambda]\}$, and M is the standard solver of f_λ. By the ISF definition, M on input x computes $f_\lambda(x)$ in $\hat{t}$ sequential steps with parallelization c, i.e, $c\hat{t}$ steps in total. On the other hand, for any $k \geq 1$, the adversary takes at least $(1-\delta)k\hat{t} = (1-(1-\frac{1-\delta}{c}))kc\hat{t}$ (sequential) steps to compute $f_\lambda^k(x)$. Hence, I is a $(c \cdot \hat{t}, 1 - \frac{1-\delta}{c}, 1)$-hard ISP. □

The first problem we encounter when trying to base Bitcoin on the ISP implied by an ISF, is that ISFs are functions, i.e., for a single problem statement there exists only one possible witness. This in turn means that it is impossible to encode information in the witness, which as argued in the previous section, is necessary to prove security. This aspect, however, is not the only problem. Assume that an *iterated sequential relation* (ISR), the obvious generalization of ISFs to relations with similar security guarantees exists (Definitions 73 and 74), and that it satisfies all the necessary conditions described in the previous section—i.e., collision resistance, fast verification and the ability to encode a random transaction to the witness. Even then, we can show that if we base our protocol to an ISR, it will be insecure. The main reason being that the solving algorithm must be *parallelizable*—exactly the opposite of what sequential functions or relations guarantee.

Definition 73. $R : X \times W$ is a (t, ε)-*sequential relation* for $\lambda = O(\log(|X|))$, if the following conditions hold:

1. There exists an algorithm that for all $x \in X$ computes a witness for R in parallel time t using $\mathsf{poly}(\log(t), \lambda)$ processors.

2. For all $\mathcal{A}$ that run in parallel time strictly less than $(1-\varepsilon)\cdot t$ with $\mathsf{poly}(t,\lambda)$ processors:
$$\Pr[(x,w_{\mathcal{A}})\in R|w_{\mathcal{A}}\leftarrow\mathcal{A}(\lambda,x),x\leftarrow X]<\mathsf{negl}(\lambda).$$

Definition 74. Let $G : X\times X$ be a relation that satisfies (t,ε)-sequentiality. A relation $R : X\times X^k$ defined as $(x_0,(x_i)_{i\in[k]})\in R$ iff $\bigwedge_{i=1}^{k}(x_{i-1},x_i)\in G$ is called an *iterated sequential relation* (with round relation G), if for all $k=2^{o(\lambda)}$, R is (kt,ε)-sequential.

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

We show next, that under the assumption that all machines in our computational model take sequential steps with approximately the same rate and have some small amount of parallelization available, the (ideal) perfectly secure ISR-based Bitcoin protocol in insecure. Formally, similarly to [BBBF18], we slightly modify our model and assume that the machines of honest parties in our execution are parallel RAMs that have c processors, for $c=O(1)$, while the adversary is free to choose the number of processors its machine has, as long as the rate of sequential steps $\mathsf{seq}_{\mathcal{A}}$ it takes per round, is close to that of honest parties $\mathsf{seq}_{\mathcal{H}}$, i.e, $\mathsf{seq}_{\mathcal{A}}\leq(1+\varepsilon)\cdot\mathsf{seq}_{\mathcal{H}}$, for some $\varepsilon\in[0,1)$. The total number of steps each party takes per round is the product of the number of processor it uses times the number of sequential steps it takes. We call this the (c,ε)-*sequential model.*

Theorem 75. *Let R be an $(\hat{t},0)$-iterated sequential relation which can be evaluated in time $\hat{t}\in poly(\lambda)$ with $c=O(1)$ parallel processors, and I be the implied ISP instance as in Lemma 72. For $t_{\mathcal{H}}$,$t_{\mathcal{A}}$ and any polynomial $p(\cdot)$, such that $t_{\mathcal{A}}=t_{\mathcal{H}}(1+\frac{1}{p(\lambda)})$, $\Pi_{\mathsf{PL}}^{\mathsf{isp}}(I)$ does not satisfy Persistence for any polynomially large k in the $(c,\frac{1}{p(\lambda)})$-sequential model.*

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

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

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

Hence, the adversary with overwhelming probability computes a chain of k' blocks faster than the honest parties for some polynomially large k'. Note, that the adversary can also tell exactly when the honest parties are going to compute a chain of the same length, and thus break Persistence for any k as in Theorem 70. The theorem follows. □

Another domain where our observation may be of great interest is that of "useful" PoW blockchain protocol, i.e., protocols where parties create PoWs by solving real-life problems. Namely, if all parties that run a Bitcoin-like blockchain try to solve the same useful problem, and this problem is not amendable to parallelization, an attack as above may be a real security threat.

5.4 Sufficient Conditions and a Provably Secure ISP-based Blockchain

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

5.4.1 Enhanced ISPs

First, we present a set of extra ISP properties, in addition to hardness, that are specific to the use of an ISP in blockchain applications. We note that the properties' specifics are not necessary for the description of our protocol, hence the eager reader can directly proceed to Section 5.4.2.

The first property, has to do with establishing an upper bound t on the running time of the verification algorithm V, for the reasons explained in Section 5.3.

Definition 76. An ISP $I = (V, M, S)$ is *t-verifiable* iff algorithm V takes time at most t (on all inputs).

In general, attacking an honest solver amounts to finding a certain set of inputs over which the honest solving algorithm fails to produce witnesses sufficiently fast. In order to combat this attack, we introduce the following property: We say that an ISP I is (t,α)-*successful* when the probability that M[4] computes a witness in t steps is at least α.

Definition 77. An ISP $I = (V, M, S)$ is (t,α)-*successful* iff for $\lambda \in \mathbb{N}$, $\Lambda[X,W,R] \in I_\lambda$, and for all $x \in X$ it holds that:

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

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

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

Definition 78. An ISP $I = (V, M, S)$ is *t-next-problem simulatable* iff there exists a t-bounded RAM Ψ such that for any PPT RAM D, any $\lambda \in \mathbb{N}$, any $z \in \{0,1\}^{poly(\lambda)}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $x \in X$, it holds that

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

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

Definition 79. An ISP $I = (V, M, S)$ is *t-witness malleable* iff there exists a t-bounded RAM Φ such that for any PPT RAM D, any $\lambda \in \mathbb{N}$, any $z \in \{0,1\}^{poly(\lambda)}$, any $\Lambda[X, W, R] \in I_\lambda$, and any $(x, w) \in R$, it holds that $(x, \Phi(x, w)) \in R$, and

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

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

Definition 80. An ISP $I = (V, M, S)$ is $(t_{\mathsf{ver}}, t_{\mathsf{succ}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$*-enhanced* iff it is correct, t_{ver}-verifiable, $(t_{\mathsf{succ}}, \alpha)$-successful, t_{nps}-next-problem simulatable, t_{mal}-witness malleable, and $(t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-hard against precomputation.

Ranges of parameters. An ISP scheme with trivial parameters is of limited use in a distributed environment; for example, if $\delta_{\mathsf{hard}} \ll 1$ or $t_{\mathsf{hard}} \ll t_{\mathsf{ver}}$. Here we describe the parameters' ranges that make for a non-trivial enhanced ISP. First off, and ignoring negligible terms, one can show that $\alpha \leq \frac{t_{\mathsf{succ}}}{(1-\delta_{\mathsf{hard}})t_{\mathsf{hard}}}$ (see Lemma 86). On the other hand, the successful property always holds for $\alpha = 0$. Therefore, for a non-trivial ISP scheme it should hold that α is close to $\frac{t_{\mathsf{succ}}}{(1-\delta_{\mathsf{hard}})t_{\mathsf{hard}}}$. Moreover, in order to avoid attacks as the one against sequentially secure primitives, described in Section 5.3.4, and to ensure that our scheme is parallelizable, we will require α to be smaller than 1.

Next, and as already discussed in Section 5.3, to avoid denial of service attacks, $\theta \cdot t_{\mathsf{ver}}$ must be sufficiently small compared to t_{hard}, the running time of the solving algorithm M. Finally, t_{mal} should be a lot smaller than t_{hard}, otherwise M can be used as a trivial simulator. We note, that the security of the protocol that we present later will rely on the fact that an enhanced ISP scheme with favorable parameters exists, mainly reflected in Assumption 5 (Section 5.4.2).

Comparison with SoW. The security properties we have discussed in this chapter bear similarities with the ones we introduced in Chapter 4. Firstly, the verifiability and successfulness properties are almost identical. Regarding almost-independent runtime, the other honest prover property explored in Chapter 4, here we take a computational approach and require that the runtime simulator in the next-problem simulatability property is efficient, does not depend on the specific problem instance, and is *computationally* indistinguishable from the honest prover runtime distribution. Finally, regarding hardness, while the MU-TCMA property of Chapter 4 implies some kind of hardness when problems are composed in a "chain" structure, as it is the case for ISP hardness, it excludes the possibility of malleable witnesses/signatures, a critical property of our ISP construction. On the other hand, the fact that ISP witnesses may be malleable, implies that any implementation of an SoW scheme from an enhanced ISP problem that retains malleability is going to be insecure.

Remark 11. Gentry and Wichs [GW11] define as falsifiable the cryptographic assumptions that can be expressed as a game between an efficient challenger and an adversary. We note that all assumptions that constitute an enhanced ISP are falsifiable in this sense, with two caveats: First, due to the concrete security approach our work takes, the challenger should take as input the number of steps of the adversary. Second, in most of the introduced properties, we quantify over all instance descriptions Λ and statements x, which is not immediate to express in the framework of [GW11]. We could instead first provide as input to the adversary randomly sampled Λ and x, and then have him provide us a statement x' and a sequence of witnesses $(w_i)_i$ such that $S(x,(w_i)_i) = x'$, for which the properties should hold. For simplicity, here we choose the former version of the definitions. However, we note that the proof techniques we use later can be easily adapted to handle the latter.

5.4.2 The provably secure ISP-based protocol

The main challenge that our protocol has to overcome, is that while its security is based on iterated hardness (Definition 71), it operates in a setting where the adversary can also take advantage of the work of honest parties. This includes, the adversary learning the information leaked by the honestly produced blocks, as well as, honest parties directly working on the chain it is extending. In contrast, the hard ISP experiment does not directly provide any guarantees about these cases; the adversary does not receive any externally computed witnesses.

Towards this end, blocks in our protocol, instead of exposing the relevant witness computed, contain a proof of knowledge (PoK) of such a valid witness through a non-interactive zero-knowledge (NIZK) proof. At first, the fact that we use NIZK proofs for a language that is moderately hard may seem counter-intuitive, due to the fact that a trivial simulator and extractor exist for the zero-knowledge and soundness properties, since computing a new witness for a given statement takes polynomial time. Instead, following our general approach, we make concrete assumptions regarding the efficiency of both the simulator and the extractor. Informally, we require that the time it takes to simulate a proof or extract a witness is a lot smaller than the time it takes for honest parties to compute a witness (see Assumption 5).

In addition, to further reduce the security guarantees required by the ISP, in our protocol the hash chain structure of blocks is decoupled from the underlying computational problem. Finally, the protocol adopts the longest-chain selection rule, which as we will see later allows it to operate even if the witnesses of the ISP are malleable (cf. Remark 9).

Protocol description. Next, we are going to describe our new protocol. Our protocol, $\Pi^{\mathsf{new}}_{\mathsf{PL}}$, uses as building blocks three cryptographic primitives: An enhanced ISP $I = (M,V,S)$, a collision-resistant hash function family $\mathcal{H}$, and a robust NIZK (Definition 114) protocol $\Pi_{\mathsf{NIZK}} = (q,\mathsf{P},\mathsf{V},\mathsf{S} = (\mathsf{S}_1,\mathsf{S}_2),\mathsf{E})$ for the language[5]

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

where $\Lambda[X,W,R]$ is an ISP instance of I. Π_{NIZK} also supports labels, which we denote as a superscript on P and V. The initialization of these primitives happens through the CRS all parties share at the start of the execution, which contains: An instance description $\Lambda[X,W,R]$, a statement $x_{\mathtt{Gen}}$, the description of a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ and the NIZK reference string Ω, each randomly sampled from $I_\lambda, X, \mathcal{H}, \{0,1\}^{q(\lambda)}$, respectively. Moreover, as in [GKL15], our protocol is parameterized by functions $\mathrm{V}(\cdot),\mathrm{R}(\cdot),\mathrm{I}(\cdot)$ that capture higher-level applications (such as Bitcoin).

Next, we introduce some notation needed to understand the description of the algorithms (similarly to Section 5.3.2). We use the terms block and chain to refer to tuples of the form $\langle s,m,x,\pi\rangle \in \{0,1\}^\lambda \times \{0,1\}^* \times X \times \{0,1\}^{\mathsf{poly}(\lambda)}$, and sequences of such tuples, respectively. The rightmost (resp., leftmost) block of chain $\mathcal{C}$ is denoted by head$(\mathcal{C})$ (resp., tail$(\mathcal{C})$). Each block contains the hash of the pre-

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

vious block s, a message m, the next problem x to be solved, and a NIZK proof π. We denote by $B_{\texttt{Gen}} = \langle 0^\lambda, 0^\lambda, x_{\texttt{Gen}}, 0^\lambda \rangle$ 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 $B_{\texttt{Gen}}$; (ii) the contents of the chain $\mathbf{m}_{\mathcal{C}} = (m_1, \ldots, m_k)$ are valid according to the chain validation predicate V, i.e., $\mathrm{V}(\mathbf{m}_{\mathcal{C}})$ is true; (iii) $s_{i+1} = H(s_i, m_i, x_i, i)$[6] for all $i \in [k]$, and (iv) $\mathsf{V}^{s_{i+1}}((\Lambda, x_{i-1}, x_i), \pi_i)$ is true for all $i \in [k] \setminus \{1\}$ (see Algorithm 14). We call $H(s_i, m_i, x_i, i)$ the *hash of block B_i* and denote it by $H(B_i)$, and define $H(\mathcal{C}) \overset{def}{=} H(\text{head}(\mathcal{C}))$. We will consider two valid blocks or chains as *equal*, if all their parts match, except possibly the NIZK proofs.

Algorithm 14 The validate procedure, parameterized by $B_{\texttt{Gen}}$, the hash function $H(\cdot)$, the *chain validation predicate* $V(\cdot)$, and the verification algorithm V of Π_{NIZK}. The input is $\mathcal{C}$.

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

At each round, each party chooses the longest valid chain among the ones it has received (as in Algorithm 10) 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

[6]We include a fixed length (λ-bit) encoding of the height of the block in the hash on purpose. This way, the contents of the hash chain form a suffix-free code [BJL17], which in turn implies collision resistance.

run the solver M on the problem x defined in the last block $\langle s,m,x,\pi\rangle$ of the chosen chain $\mathcal{C}$. If it succeeds on finding a witness w, it will then compute a NIZK proof that it knows a witness w such that $(x,w)\in R$ and $S(x,w) == x'$, for some $x'\in X$. The proof should also have a label $H(H(head(\mathcal{C})),m',x',|\mathcal{C}|+1)$, where m' is the output of the input contribution function $\mathrm{I}(\cdot)$; see Algorithm 15. Then, the party diffuses the extended chain to the network. Finally, if the party is queried by the environment, it outputs $\mathrm{R}(\mathcal{C})$, where $\mathcal{C}$ is the chain selected by the party; the chain reading function $\mathrm{R}(\cdot)$ interprets $\mathcal{C}$ differently depending on the higher-level application running on top of the backbone protocol. The main function of the protocol is the same as Algorithm 9 of Chapter 4, except instead of sow, procedure pow of Algorithm 15 is invoked when a new block is to be mined. We assume that all honest parties take the same number of steps $t_{\mathcal{H}}$ per round.

Algorithm 15 The *proof of work* function. The input is $(m',\mathcal{C})$.

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

In order to turn the above protocol into a protocol realizing a public transaction ledger, we define functions $\mathrm{V}(\cdot),\mathrm{R}(\cdot),\mathrm{I}(\cdot)$ exactly as in [GKL15], see Table 3.2 at Section 3.4. We denote the new public ledger protocol by $\Pi^{\mathsf{new}}_{\mathsf{PL}}$.

Security analysis. In this section, we prove that $\Pi^{\mathsf{new}}_{\mathsf{PL}}$ implements a robust public transaction ledger. The first assumption we are going to make is that the underlying ISP I is enhanced, and that the runtimes of the procedures of the NIZK system are suitably bounded.

Assumption 4 (ISP Assumption). For parameters $t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}}, t_{\mathsf{P}}$, $t_{\mathsf{V}}, t_{\mathsf{S}}$, and t_{E} we assume that:

- ISP I is $(t_{\mathsf{ver}}, t'_{\mathcal{H}}, \alpha, t_{\mathsf{nps}}, t_{\mathsf{mal}}, t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$-enhanced;[7]
- running the prover (resp., verifier, simulator, extractor) of Π_{NIZK} takes t_{P} (resp. $t_{\mathsf{V}}, t_{\mathsf{S}}, t_{\mathsf{E}}$) steps.

Next, we introduce some additional notation necessary to formalize our second assumption, that has to do with the computational power of the honest parties and the adversary. For brevity, and to better connect our analysis to previous work [GKL15, PSS17] and the previous chapters, we denote by $\beta = \frac{1}{(1-\delta_{\mathsf{hard}})\cdot t_{\mathsf{hard}}}$, the upper bound on the rate at which the adversary can compute witnesses in the iterated hardness game. As in Chapter 4, we introduce two variables, $t'_{\mathcal{H}}$ and $t'_{\mathcal{A}}$, that have to do with the effectiveness of honest parties and any adversary against our protocol in producing witnesses for I. $t'_{\mathcal{H}}$ is a lower bound on the number of steps each honest party takes per round running M. It thus holds that in any round at least $n-t$ parties will run M for at least $t'_{\mathcal{H}}$ steps. $t'_{\mathcal{A}}$ denotes the maximum time needed by a RAM machine to simulate the adversary, the environment and the honest parties in one round of the protocol execution, without taking into account calls made to M by the latter, and with the addition of one invocation of the NIZK extractor. They amount to:

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

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

We are now ready to state our main computational assumption regarding the honest parties and the adversary. As in Chapter 4, besides assuming that the total number of steps the honest parties take per round exceed those of the adversary, and that the total block generation rate is bounded, we have to additionally assume that the efficiency of the solving algorithm M used by honest parties is comparable to that of the adversary; i.e, as explained earlier, α should be comparable to $\beta t'_{\mathcal{H}}$. In our formalization, that we present next, the first condition in our assumption corresponds to the observation we just made, while the other two correspond to adaptations of the older assumptions. To avoid confusion, we cast most of our analysis based on the δ parameter. Furthermore, note that under optimal conditions – i.e., δ_{ISP} close to 0 and $t_{\mathsf{P}}, t_{\mathsf{V}}, t_{\mathsf{E}}, t_{\mathsf{S}}, t_{\mathsf{nps}}, t_{\mathsf{mal}}$ a lot smaller than $t_{\mathcal{H}}$ – our assumption allows for an adversary that controls up to $1/3$ of the total computational power available (vs. $1/2$ in the RO model).

Assumption 5. There exist $\delta_{\mathsf{ISP}}, \delta_{\mathsf{Steps}}$ and $\delta \in (0,1)$, such that for sufficiently large $\lambda \in \mathbb{N}$:

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

λ :	security parameter
n :	number of parties
t :	number of parties corrupted
$t_{\mathcal{H}}$:	number of steps per round per honest party
$t_{\mathcal{A}}$:	total number of adversarial steps per round
$t'_{\mathcal{H}}$:	lower bound on number of steps running M per round per honest party
$t'_{\mathcal{A}}$:	round simulation cost, excluding honest calls to M
θ :	upper bound on the number of messages sent by the adversary per round
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 :	upper bound on the total block generation rate
δ_{Steps} :	honest parties' advantage on number of steps
δ_{ISP} :	adversary's advantage on ISP witness computation rate
k_{hard} :	convergence parameter of ISP hardness

Table 5.1: *The parameters in our analysis:* $\lambda, n, t, t_{\mathcal{H}}, t_{\mathcal{A}}, t'_{\mathcal{H}}, t'_{\mathcal{A}}, \theta, k_{hard}$ *are in* $\mathbb{N}$, $f, \gamma, \beta, \delta, \delta_{\mathsf{Steps}}, \delta_{\mathsf{ISP}}$ *are in* $\mathbb{R}$.

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

Exactly as in Chapter 4, using a collision resistant hash function to connect subsequent blocks in a chain, allows us to establish certain properties about its structure. We provide only the relevant lemmas, as the analysis is the same as in the previous chapter.

Lemma 81. *Let $\mathcal{H}$ be a collision-resistant hash function family. The probability that any PPT RAM $\mathcal{A}$, given B_{Gen}, can find two distinct valid chains $\mathcal{C}_1, \mathcal{C}_2$ such that $H(\mathcal{C}_1) = H(\mathcal{C}_2)$, is negligible in λ.*

Corollary 82. *Let $\mathcal{H}$ be a collision-resistant hash function family. Then, for any PPT $\mathcal{A}, \mathcal{Z}$ no insertions or copies (Definition 12) occur in* $\mathrm{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$ *with probability* $1-\mathsf{negl}(\lambda)$.

We proceed to the main part of the analysis. We follow the notation and definitions of Section 3.3.2, for: successful and uniquely successful rounds, honest, adversarial and u.s. blocks, random variables $X(\cdot), Y(\cdot)$. Moreover, with respect to some block B computed by an honest party P at some round r, let $Z_r^P(R)$ denote the maximum number of distinct blocks diffused by the adversary during R that have B as their ancestor and lie on the same chain; note that honest parties compute at most one block per round. If P is corrupted or did not compute any block at r, let $Z_r^P(R) = 0$. Define $X_r^P(R)$ similarly.

Lower bounds on the expected rate of successful and uniquely successful rounds are defined as in Chapter 4:

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

We will formally show that this is the case later, in Lemma 86. As in previous chapters we show that the rate of uniquely successful rounds outperforms the rate at which the adversary produces witnesses. Unlike previous analyses, we require γ to be sufficiently bigger than two times $\beta \cdot t'_{\mathcal{A}}$, which follows from the fact that in Assumption 5 the honest parties take at least double the steps the adversary takes per round.

Lemma 83. *Assume an ISP that complies with Assumptions 4 and 5. It holds that* $\gamma \geq 2(1+\delta)\beta t'_{\mathcal{A}}$.

Proof. For γ it holds that:

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

where we have first used Bernouli's inequality, and then the three conditions from Assumption 5. The last inequality follows from the fact that $\frac{\delta_{\mathsf{Steps}}-\delta_{\mathsf{ISP}}}{2} \geq \delta$. □

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

Proof. W.l.o.g., let i be the first round of $R = \{i' | i \leq i' < i+s\}$, and let E be the event where in $\text{VIEW}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$ party P at round i mined a block B, and the adversary mined at least $\beta t'_{\mathcal{A}} s$ blocks until round $i+s$ that extend B and are part of a single chain. For the sake of contradiction, assume that the lemma does not hold, and thus $\Pr[E]$ is non-negligible. Using $\mathcal{A}$, we will construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that breaks the iterative hardness against precomputation (Definition 71) of I with non-negligible probability.

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

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

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

$\mathcal{A}'_2$ takes as input a problem statement x sampled from X, as in Definition 71. It should somehow introduce x to the simulated protocol execution, without the adversary noticing any difference that could help him distinguish from the real execution. Let $B_0 = \langle s_0, m_0, x_0, \pi_0 \rangle$ be the head of chain $\mathcal{C}$ that party P is extending at round i, and m_1 the block input it produced for this round using the input contribution function $\mathrm{I}(\cdot)$. $\mathcal{A}'_2$ is first going to run M on input x for the amount of steps available to P. If it is successful and produces some witnesses w, it will diffuse the following block:

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

where the last component is a simulated NIZK proof for the statement $(x_0, S(x,w))$.

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

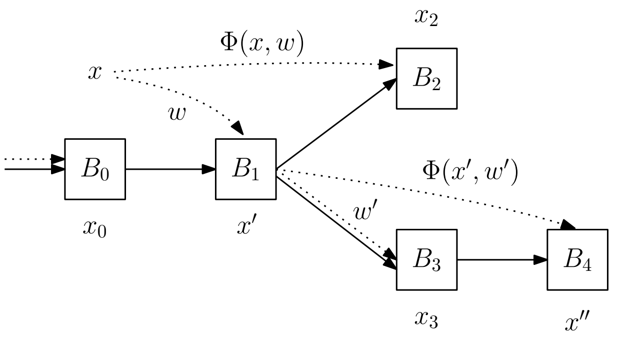


Figure 5.1: *A possible scenario according to Lemma 84. The blocks have been generated in order B_0, B_1, B_2, B_3, B_4, with B_3 being the only adversarial block. The cases where a valid witness is either known or can be extracted, and a NIZK proof has either been computed or simulated for the depicted transitions, correspond to the dotted and normal arrows, respectively.*

$\mathcal{A}_2'$ will follow a more complex strategy to simulate the rest of the honest parties invocations. For each honest party, it will run the next-problem simulator $\Psi(1^\lambda)$ and check if the numbers of steps output is less than the number of steps available on this invocation. If they are not, $\mathcal{A}_2'$ will proceed by just updating the state of this party for the round. Otherwise, it will simulate its behavior when being successful, as follows: Let block $B^* = \langle s^*, m^*, x^*, \pi^* \rangle$ be the head for the chain $\mathcal{C}^*$ the honest party was trying to extend with message m'' in this round. Let $B_j = \langle s_j, m_j, x_j, \pi_j \rangle$ be the adversarial block that descends B_1 and maximizes the number of adversarial blocks between itself and B_1. Let $B' = \langle s', m', x', \pi' \rangle$ be the parent of B_j. If no such adversarial block exists, assume that $B_j = B_1$ and $B' = \langle \emptyset, x, w, \emptyset \rangle$. $\mathcal{A}_2'$ first runs the NIZK extractor $\mathsf{E}^{H(B_j)}(\Omega, ((\Lambda, x', x_j), \pi_j), tk)$ to obtain a witness w' for x'. Then, it runs $\Phi(x', w')$ and obtains a new witness w'' for x'; let $x'' = S(x', w'')$. Finally, it is going to make $\mathcal{A}_2$ believe that the block it has computed extends B^*, instead of B', by simulating a NIZK proof as follows: $\pi'' = \mathsf{S}_2^{H(H(B^*), m'', x'', |\mathcal{C}^*|+1)}(\Omega, (\Lambda, x^*, S(x', w'')), tk)$. The new block that $\mathcal{A}_2'$ is going to diffuse is $\langle H(B^*), m'', x'', \pi'' \rangle$. We point to Figure 5.1 for an example of the procedure described above. If $\mathcal{A}_2'$ was not successful when it run $M(x)$ to extend B_0, it is going to simulate honest parties work as follows: to extend block

$\hat{B} = \langle \hat{s}, \hat{m}, \hat{x}, \hat{\pi} \rangle$, it will first use Ψ to see if it succeeds and if yes generate the next problem statement $\hat{x}'$, and then use S as above to generate a NIZK proof $\hat{\pi}'$ for block $\langle H(\hat{B}), \hat{m}', \hat{x}', \hat{\pi}' \rangle$.

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 $\textsc{view}^{t,n}_{\Pi^{\mathsf{new}}_{\mathsf{PL}},\mathcal{A},\mathcal{Z}}$.

Claim 7. $H_{sim} \overset{c}{\approx} H_0$.

Proof. We start by describing a sequence of hybrids:

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

By the zero knowledge property of the NIZK proof system (Definition 114) it easily follows that H_0 is indistinguishable from H_1; H_0 corresponds to the real execution, while H_1 to the simulated one.

Next, we will argue that $H_{u-1,v}$ is indistinguishable from $H_{u,v}$, for some $u \in [n], v \in R$ (let $H_{0,i} = H_1$), by contradiction . Assume $H_{u-1,v} \overset{c}{\not\approx} H_{u,v}$. There are two cases.

In the first case, $u = P$, $v = i$. The difference between the two executions, is that in $H_{P,i}$, instead of running $M(x_0)$ and computing $S(x_0, M(x_0))$, $M(\mathcal{X})$ is run and the next problem computed is $S(\mathcal{X}, M(\mathcal{X}))$, where by $\mathcal{X}$ we denote the uniform distribution over X. Assuming that the two hybrids are distinguishable, by an averaging argument there exists a PPT distinguisher D, some auxiliary register z, and $\Lambda[X, W, \hat{R}] \in I_\lambda, x_0, x_1 \in X, t_0 \leq t_{\mathcal{H}}$ such that D distinguishes $(\Lambda, x_0, (S(x_0, M(x_0)), \mathsf{Steps}_M(x_0) < t_0))$ from $(\Lambda, x_0, (S(x_1, M(x_1)), \mathsf{Steps}_M(x_1) < t_0))$; z will be equal to the state of an execution where P has t_0 steps to extend problem x_0. This is a contradiction, since by the next-problem simulatability property it follows that:[8]

$$(\Lambda, x_0, (S(x_0, M(x_0)), \mathsf{Steps}_M(x_0))) \overset{c}{\approx} (\Lambda, x_0, \Psi(1^\lambda)) \overset{c}{\approx} (\Lambda, x_0, (S(x_1, M(x_1)), \mathsf{Steps}_M(x_1)))$$

where the last part follows from the fact that $\Psi_0(1^\lambda)$ and x_1 do not depend x_0.

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

For the second case, assume that either $u \neq P$ or $v \neq i$. W.l.o.g., assume that u is successful. Similarly, by an averaging argument we can show that there exists a PPT distinguisher D, some auxiliary register z, and $\Lambda[X,W,\hat{R}] \in I_\lambda, x_0, x_1 \in X, w_1 \in W, t_0 \leq t_{\mathcal{H}}$ such that D distinguishes $(\Lambda, x_0, (S(x_0, M(x_0)), \mathsf{Steps}_M(x_0) < t_0))$ from $(\Lambda, x_0, (S(x_1, \Phi(x_1, w_1)), \Psi_1(1^\lambda) < t_0))$ and $(x_1, w_1) \in \hat{R}$, where $\Psi_1(1^\lambda)$ is the steps component of $\Psi(1^\lambda)$. We arrive to a contradiction due to the witness malleability property:

$$\begin{aligned}(\Lambda, x_0, (S(x_0, M(x_0)), \mathsf{Steps}_M(x_0))) &\overset{c}{\approx} (\Lambda, x_0, \Psi(1^\lambda)) \\ &\overset{c}{\approx} (\Lambda, x_0, (S(x_1, M(x_1)), \mathsf{Steps}_M(x_1))) \overset{c}{\approx} (\Lambda, x_0, (S(x_1, \Phi(x_1, w_1)), \Psi_1(1^\lambda)))\end{aligned}$$

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

The claim follows by the fact that $H_{n,s}$ is the same as H_{sim}. $\dashv$

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

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

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

Note that we can do exactly the same reduction without simulating honest parties' work. Then, the total running time of the second stage of $\mathcal{A}'$ is $s \cdot (t'_{\mathcal{A}} + nt'_{\mathcal{H}})$-bounded.

Hence, we can derive the following bound on the longest chain that can be produced by both honest and malicious parties during a certain number of rounds.

Corollary 85. *For any set of consecutive rounds R, where $|R| \geq \frac{k_{\mathsf{hard}}}{\beta(t'_{\mathcal{A}}+nt'_{\mathcal{H}})}$, for any party P, and any round $i \in R$, the probability that $Z_i^P(R) + X_i^P(R) \geq \beta(t'_A + nt'_{\mathcal{H}}) \cdot |R|$ is* $\mathsf{negl}(\lambda)$.

Next, we prove lower bounds on the rate of successful and uniquely successful rounds. Our proof crucially depends on the next-problem simulatable property of I. Specifically, the fact that the runtime simulator $\Psi(1^\lambda)$ does not depend on the problem statement, implies that the steps honest parties take running M at each round will be indistinguishable from a set of i.i.d random variables following the distribution defined by $\Psi(1^\lambda)$. Hence, for some set of rounds the sum of the Bernoulli random variables of the event that a round is successful or uniquely successful, which only depends on the running time of M, will necessarily have good concentration properties. In turn, this implies that we can lower-bound the rates of successful and uniquely successful rounds with good probability.

Lemma 86. *For any set of consecutive rounds R, with $|R| \geq \frac{\lambda}{\gamma\delta^2}$, the following two events occur with negligible probability in λ:*

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

Proof. We follow a similar strategy as in the proof of Lemma 49. The ideal experiment this time, will be one where the honest parties behavior is simulated using the simulators Ψ and S of the next-problem simulatable property and the NIZK proof system similarly to Lemma 84; Ψ is used to determine whether a party is successful and the next problem statement, and S is used to generate the required NIZK. Using similar arguments we can show that the view of the adversary in the real execution and its view on the ideal one are computationally indistinguishable.

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

Claim 8. $\Pr[\Psi(1^\lambda) \leq t_{\mathcal{H}}] \leq t_{\mathcal{H}}\beta + \mathsf{negl}(\lambda)$.

Proof. For the sake of contradiction, assume that the difference $\Pr[\Psi(1^\lambda) \le t_{\mathcal{H}}] - t_{\mathcal{H}}\beta$ is non-negligible. First, we will argue that there exists an $x \in X$, such that $\Pr[\mathsf{Steps}_M(x) \le t_{\mathcal{H}}] - t_{\mathcal{H}}\beta$ is negligible. For the sake of contradiction, assume that for all $x \in X$, $\Pr[\mathsf{Steps}_M(x) \le t_{\mathcal{H}}] - t_{\mathcal{H}}\beta$ is non-negligible. By the iterated hardness property, we have that for $k \ge k_{\mathsf{hard}}$, any k/β-bounded adversary will compute k or more witnesses with negligible probability in λ (assume we pick a k that is a multiple of β). This implies that the expected number of blocks any such adversary computes is at most $k + \mathsf{negl}(\lambda)$. Let an adversary that is based on M work as follows: on some initial input x, it runs M for at most $t_{\mathcal{H}}$ steps. If, it succeeds on producing a witness, it computes the next problem, and runs M again with the new input. If not, it runs M on the initial input. By our assumption and the linearity of expectation, the expected number of blocks our adversary will mine on k/β steps, is greater than $(\beta t_{\mathcal{H}} + \varepsilon)\frac{k}{\beta t_{\mathcal{H}}} \ge k(1 + \frac{\varepsilon}{\beta t_{\mathcal{H}}})$, where ε is a non-negligible function. This is a contradiction. Hence, there exists an $x_0 \in X$, such that $\Pr[\mathsf{Steps}_M(x_0)] - t_{\mathcal{H}}\beta$ is negligible. This in turn implies that we can construct a distinguisher for $\Psi(1^\lambda)$ and $\mathsf{Steps}_M(x_0)$, by checking whether the input of the distinguisher is less or equal to $t_{\mathcal{H}}$. This is a contradiction to the next-problem simulatable property. Therefore, the claim follows. $\dashv$

Similarly to Lemma 49, we can now define random variables $\hat{X}(\cdot), \hat{Y}(\cdot), X'(\cdot), Y'(\cdot)$ for the ideal execution. Due to the fact that different invocations of $\Psi(1^\lambda)$ are independent, and from the lower and upper bounds we have already established, it follows that for any $i \in R : \mathbb{E}[Y'(\{i\})] \ge \gamma$ and $\mathbb{E}[X'(\{i\})] \ge f$. Moreover, by an application of the Chernoff Bound it holds that: $\Pr[Y'(R) \le (1-\frac{\delta}{4})\gamma|R|] \le \Pr[Y'(R) \le (1-\frac{\delta}{4})\mathbb{E}[Y'(R)]] \le e^{-\Omega(\delta^2\gamma|R|)}$ and similarly $\Pr[X'(R) \le (1-\frac{\delta}{4})f|R|] \le e^{-\Omega(\delta^2 f|R|)}$. Since the conditions of the above two events can be checked in polynomial time, it follows that they should also hold for the real execution with negligible difference in probability. Otherwise, a distinguisher would be able to use them to efficiently distinguish between the two executions. The lemma follows.

$\square$

Following the strategy of [GKL15], we are now ready to define the set of *typical executions* for this setting.

Definition 87 (Typical execution). An execution is *typical* if and only if $\lambda \ge 9/\delta$ and for any set R of consecutive rounds with $|R| \ge \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta^2}$, the following hold:

1. $Y(R) > (1-\frac{\delta}{4})\gamma|R|$ and $X(R) > (1-\frac{\delta}{4})f|R|$;

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

Theorem 88. *An execution is typical with probability* $1 - \mathsf{negl}(\lambda)$.

Proof. The proof of the theorem proceeds as in Theorem 51, with the only difference that for Lemma 84 and Corollary 85, we assume an adversary with power $t'_{\mathcal{A}} = \frac{\gamma}{\beta \cdot 2(1+\delta)}$, so that $\frac{k_{\mathsf{hard}}}{\beta t'_{\mathcal{A}}} \leq \frac{4k_{\mathsf{hard}}}{\gamma\delta^2} \leq |R|$ and $\frac{k_{\mathsf{hard}}}{\beta(t'_{\mathcal{A}}+nt'_{\mathcal{H}})} \leq \frac{4k_{\mathsf{hard}}}{\gamma} \leq |R|$.

$\square$

Having established that typical rounds happen with overwhelming probability, the rest of the proof follows closely that of Chapter 4, Section 4.4.2. The only difference is that to prove the corresponding common-prefix lemma (cf. Lemma 56), although we can match u.s. blocks to adversarial blocks in one of the two chains that constitute the fork, the typicality of the execution only provides a bound on the maximum number of blocks in a single chain. Hence, only half of the blocks matched must outnumber the uniquely successful rounds in this interval, which is also the reason that our proof only works with an adversary controlling up to $1/3$ of the parties. Next, we state our main theorem.

Theorem 89. *Assuming the existence of a collision-resistant hash function family, a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), and an enhanced ISP problem I that comply with Assumptions 4 and 5, protocol* $\Pi_{\mathsf{PL}}^{\mathsf{new}}$ *implements a robust public transaction ledger with parameters* $k = \frac{\max\{4k_{\mathsf{hard}},\lambda\}}{\gamma\delta}$ *and* $u = \frac{2k}{(1-\frac{\delta}{4})f}$, *except with negligible probability in* λ.

5.4.3 An Enhanced ISP Construction

We now present an ISP problem that is plausibly hard against precomputation (Definition 71), and satisfies *all* other properties of an enhanced ISP (Definition 80). Our construction is based on a fixed-length input hash function that satisfies the properties of a *weak* computational randomness extractor. As explained in [DSGKM12], this assumptions is weaker than assuming a fixed-length input pseudorandom function family (FI-PRF), a common assumption in the hash function literature [BCK96, DGH$^+$04, Kra10]. Next, we give a formal definition of the property our hash function should satisfy.[9]

Definition 90. Let $\mathcal{H} = \{\{H_k : \{0,1\}^{d\lambda} \to \{0,1\}^{\lambda}\}_{k \in K(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a hash-function family, for some $d \in \mathbb{N}^+$. $\mathcal{H}$ is a *computational randomness extracting* (CRE) hash family if for some $c \in \mathbb{N}^+$, where $c < d$, the function family $E = \{E_\lambda : \{0,1\}^{(c+1)\lambda} \times \{0,1\}^{(d-c-1)\lambda} \to \{0,1\}^{\lambda}\}_\lambda$, where $E_\lambda(x,i) \overset{def}{=} H_k(x||i)$, is a $(c\lambda)$-computational extractor (Definition 113), for any key $k \in K(\lambda)$.

We are now ready to describe our ISP construction.

Construction 2. Let $\mathcal{H}$ be a hash function family as in Definition 90. Let $T \in \{0,1\}^\lambda$ be a hardness parameter. An instance of an enhanced ISP is as follows:

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

Construction 2 is similar to Bitcoin's ISP construction (see Section 5.3.1, Construction 1), with the following differences:

1. In our construction $H_k(x||w_1)$ is required to be smaller than the hardness parameter T, while in Bitcoin $H_k(H_k(x||w_1)||w_2)$ is expected to be small, where w_1 is the hash of some message. This change allows a party who already knows a witness (w_1, w_2) for some statement, to produce a new one by changing w_2 arbitrarily.

2. Each time M tests a new possible witness, w_1 is sampled randomly, instead of just being increased by one, as in Bitcoin. This will help us later on to argue that each test succeeds with probability proportional to T.

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

[9]For simplicity we model a hash function family parameterized by security parameter $\lambda \in \mathbb{N}$. We note that our results can be easily casted in the fully concrete setting.

Assuming $\mathcal{H}$ is a computational randomness extractor is sufficient for the security properties that make up an enhanced ISP, besides hardness, to be satisfied. First, the fact that $H_k(x||w_1)$ is computationally indistinguishable from uniform, for any $x \in X$, implies that the runtime and the output of M are computationally indistinguishable from a process that sampled repeatedly a uniform value from $\{0,1\}^\lambda$ until it finds one that is smaller than T. This in turn implies that the runtime distribution of M is indistinguishable from the geometric distribution with parameter $T/2^\lambda$, and thus the successful ISP property is satisfied with parameters comparable to that proven in Chapter 4 about the RO-based construction. Further, since w_2 is also chosen uniformly at random, we can show that a simulator that samples a random value from $\mathcal{U}_\lambda$ and the geometric distribution, satisfies the next-problem simulatability property. Finally, by resampling a new w_2 uniformly at random, an admissible witness is produced, and the witness malleability property follows. We are thus able to state:

Lemma 91. *If $\mathcal{H}$ is a CRE hash family, then Construction 2 is $O(\lambda)$-next-problem simulatable, $O(\lambda)$-witness malleable, and $(t, C_{T/2^\lambda}(O(t)))$-successful for any $t \in poly(\lambda)$, where $C_{T/2^\lambda}$ is the cumulative geometric distribution with parameter $T/2^\lambda$.*

Proof. We start by showing that even if the adversary chooses the problem statement x maliciously, hashing it once together with a uniformly random string, will result in a string that is computationally indistinguishable from a uniformly sampled string. Let c,d be as in Definition 90. Fix some $\lambda \in \mathbb{N}$, $k \in K(\lambda)$ and $x \in X$. Let random variable Z be equal to $H_k(x||\mathcal{U}_{c\lambda}||\mathcal{U}_{(d-c-1)\lambda})$. By our assumption that $E(x,i) \overset{def}{=} H_k(x||i)$ is a $(c\lambda)$-computational extractor, and since $x||\mathcal{U}_{c\lambda}$ has $c\lambda$ bits of min-entropy, it follows that[10] $Z \overset{c}{\approx} \mathcal{U}_\lambda$, and since x is fixed that $(x,Z) \overset{c}{\approx} (x, \mathcal{U}_\lambda)$. Assume instead that x is sampled from some efficiently samplable distribution $\hat{X}$, as it will be the case in the actual execution, and let $\hat{Z} = H_k(\hat{X}||\mathcal{U}_{c\lambda}||\mathcal{U}_{(d-c-1)\lambda})$. For any PPT distinguisher D, sufficiently large $\lambda \in \mathbb{N}$, all $z \in \{0,1\}^{poly(\lambda)}$ it holds that:

$$
\begin{aligned}
&|\Pr[D(1^\lambda,z,\hat{X},\hat{Z}) = 1] - \Pr[D(1^\lambda,z,\hat{X},\mathcal{U}_\lambda) = 1]| \\
&= |\sum_{x' \in X} \Pr[x' = \hat{X}]\big(\Pr[D(1^\lambda,z,x',H_k(x'||\mathcal{U}_{(d-1)\lambda})) = 1] - \Pr[D(1^\lambda,z,x',\mathcal{U}_\lambda) = 1]\big)| \\
&\leq \sum_{x' \in X} \Pr[x' = \hat{X}] \cdot |\Pr[D(1^\lambda,z,x',H_k(x'||\mathcal{U}_{(d-1)\lambda})) = 1] - \Pr[D(1^\lambda,z,x',\mathcal{U}_\lambda) = 1]| \\
&\leq \sum_{x' \in X} \Pr[x' = \hat{X}] \cdot \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda)
\end{aligned}
$$

[10]We abuse the notation and use the $\overset{c}{\approx}$ relation with random variables, instead of random variable ensembles. The relevant ensembles can be easily deduced.

where the last inequality follows from the fact that $(x,Z) \overset{c}{\approx} (x,\mathcal{U}_\lambda)$ for any $x \in X$. Hence, $(\hat{X},\hat{Z}) \overset{c}{\approx} (\hat{X},\mathcal{U}_\lambda)$.

We next argue about the distribution of the running time of M. Algorithm M on input x iteratively samples a uniformly random $w_1||w'_1$ from $\mathcal{U}_{c\lambda} \times \mathcal{U}_{(d-c-1)\lambda}$, and tests whether $H_k(x||w_1||w'_1) < T$, until it finds a solution. For a moment, assume that M instead tested whether a value sampled from $\mathcal{U}_\lambda$ is smaller than T. Then, its running time would be distributed according to the geometric distribution $\mathcal{G}_p$ with parameter $p = T/2^\lambda$. Since $Z \overset{c}{\approx} \mathcal{U}_\lambda$, we can use a hybrid argument to show that the distribution of $\mathsf{Steps}_M(x)$ is computationally indistinguishable from $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$, where c_1 is a constant related to the cost of sampling a random value for each test and evaluating H, and c_2 to the cost of sampling w_2. The hybrid argument proceeds by replacing a computation of $H(x||\mathcal{U}_{(d-1)\lambda}) < T$ at some step of M, with $\mathcal{U}_\lambda < T$. If between any two hybrids the distributions of the runtime of the respective modified M is not indistinguishable, then we can easily construct a distinguisher for $H(x||\mathcal{U}_{(d-1)\lambda})$ and $\mathcal{U}_\lambda$. Hence, $\mathsf{Steps}_M(x)$ should be computationally indistinguishable from $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$. It follows that M must be $(t, \mathcal{C}_{T/2^\lambda}(O(t)))$-successful, for any $t \in poly(\lambda)$.

Next, note that M, after finding a small hash, hashes again the result with a fresh randomly sampled string w_2. Using the same hybrid argument as in the previous paragraph we can show that $(x, M(x), \mathsf{Steps}_M(x)) \overset{c}{\approx} (x, H_k(\mathcal{W}||\mathcal{U}_\lambda), c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2)$, where $\mathcal{W}$ is the uniform distribution over the hash images that are smaller than T. By our previous analysis it follows that $(x, H_k(\mathcal{W}||\mathcal{U}_\lambda), c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2) \overset{c}{\approx} (x, \mathcal{U}_\lambda, c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2)$. By the transitivity of computational indistinguishability it follows that the simulator Ψ that outputs a randomly sampled pair from $\mathcal{U}_\lambda$ and $c_1 \cdot \mathcal{G}_{T/2^\lambda} + c_2$ satisfies the next-problem simulatability property. Note, that using the inverse transform technique, we can sample from the geometric distribution (truncated to 2^λ) in $O(\lambda)$ steps.

Finally, the witness malleability property holds for $\Phi(x,(w_1,w_2))$ that outputs the witness (w_1,w'_2), where w'_2 is sampled uniformly at random. Again, $S(x,\Phi(x,(w_1,w_2)))$ will be indistinguishable from $\mathcal{U}_\lambda$. The lemma follows.

□

Regarding the hard-ISP property, to our knowledge no reductions exist (yet) of iterated hardness to weaker assumptions in the standard model, even for the more extensively studied iterated sequential functions mentioned earlier. We argue the plausibility of our construction being hard against precomputation (Definition 71), by the fact that Construction 2 is based on Bitcoin's ISP construction (see Section 5.3.1 for a simplified version), for which no attacks are known. The main idea is that if there exists an attacker against our Construction, then we can use it to break the hardness of Construction 1. In more detail, given as input a statement x, the attacker runs the attacker of our Construction with input $H(x||w)$, where w is sampled at random. It is easy to see that if $((w_1, w'_1), \ldots, (w_m, w'_m))$ are the witnesses it is going to produce, then $((w, w_1), (w'_1, w_2), \ldots, (w'_{m-1}, w_m))$ are valid witnesses for Construction 1. The following lemma highlights this relation:

Lemma 92. *If Construction 1 is based on a CRE hash family $\mathcal{H}$ and is (t, δ, k)-hard against precomputation, then Construction 2, also based on $\mathcal{H}$, is (t, δ, k)-hard against precomputation.*

Proof. As mentioned, Construction 2 is a mirror image of Construction 1, in the sense that the first hash, instead of the second, is required to be smaller than T, and the other one can have an arbitrary value. We make the following simplifying assumptions: $\mathcal{H}$ is the same in both constructions, M in Construction 1 chooses ctr at random[11] as in Construction 2, and the size of W in Construction 1 is adapted accordingly. For the sake of contradiction, assume that Construction 2 is not (t, δ, k)-hard against precomputation. Using similar arguments as in the analysis of Lemma 91, the distribution of the runtime of the solvers of both ISPs is identical. Hence, it has to be the case that there exists an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that for infinitely many λ and some $m \geq k$ breaks the hardness of Construction 2. Using $\mathcal{A}$, we are going to construct an attacker $\mathcal{A}'$ that breaks the hardness of Construction 1.

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

We proceed to analyze the winning probability of $\mathcal{A}'$. We have already argued that whenever $\mathcal{A}$ wins, $\mathcal{A}'$ also wins. Moreover, we have assumed that $\mathcal{A}$ succeeds in

[11]We can avoid this simplification by further assuming that $\mathcal{H}$ is a FI-PRF, similarly to [Theorem 1, [Kra10]].

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

□

Following similar techniques as in Theorem 65 of Chapter 4, and due to Theorem 89 and the previous two lemmas, we can implement a ledger assuming the existence of a robust NIZK, a hash family that is collision-resistant, the ISP hardness of Bitcoin's ISP scheme (Construction 1) based on a CRE hash for appropriate parameters, and that the adversary controls less than a third of the total computational power. The following theorem holds:[12]

Theorem 93. *Assuming the existence of a one-way trapdoor permutation and a dense cryptosystem (for the NIZK), collision-resistant hash functions, that Construction 1 is a* $(t_{\mathsf{hard}}, \delta_{\mathsf{hard}}, k_{\mathsf{hard}})$*-hard-ISP against precomputation based on a CRE hash family* $\mathcal{H}$*, and that for some* $\delta_{\mathsf{Steps}} \in (0,1)$*, sufficiently large* $\lambda \in \mathbb{N}$*, and* T *equal to* $\lfloor 2^{\lambda} \cdot \min\{\frac{\ln((1-\delta^2_{\mathsf{Steps}}/4)^{-1})}{t'_{\mathcal{H}}}, \frac{\delta_{\mathsf{Steps}}/4}{(t'_{\mathcal{A}}+nt'_{\mathcal{H}})(1+\delta_{\mathsf{Steps}}/2)}\}\rfloor$*, it holds that:*

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

there exists a $T \in \mathbb{N}$ *such that protocol* $\Pi^{\mathsf{new}}_{\mathsf{PL}}$ *based on Construction 2 implements a robust public transaction ledger, except with negligible probability in* λ*.*

[12]For simplicity, we assume that the cost in computational steps of evaluating H, and the hidden constant in the successful property of Lemma 91 are both 1. The theorem can be easily generalized for arbitrary costs.

Chapter 6

Security Analysis of the `GHOST` Protocol

6.1 Introduction

The popularity of Bitcoin [Nak08] has lead to a surge in the interest about its core protocol that maintains a distributed data structure called the “blockchain.” In [GKL15], the core of the Bitcoin protocol was abstracted under the moniker “Bitcoin Backbone” and it was demonstrated to be a powerful tool for solving consensus, [PSL80, LSP82], in a synchronous, permissionless and Byzantine setting where (unreliable) broadcast is the communication operation available to the participants, (a problem first considered in [AJK05, Oku05a]). In [GKL15], it was shown that the core protocol provably guarantees two properties: (i) *persistence:* if a transaction is reported as stable by one node, then it will be also reported as such by any other honest node of the system, (ii) *liveness:* all honestly generated transactions that are broadcast, are eventually reported as stable by all honest nodes. This provides a formal framework for proving the security of systems like Bitcoin, since their security can be reduced to the persistence and liveness of the underlying transaction ledger. Furthermore, it provides a way to argue formally about transaction confirmation time since the liveness property is equipped with a delay parameter that specifies the maximum transaction delay that can be caused by an adversary.

Naturally, implementing a robust transaction ledger may be achieved in various other ways, and it is a fundamental open question of the area whether the Bitcoin protocol itself is an “optimal” implementation of a robust transaction ledger, i.e. whether the parameters of the persistence and liveness properties are optimal. Indeed, many researchers have challenged various aspects of the Bitcoin system and they have proposed modifications in its core operation. Some of the modified systems maintain the protocol structure but modify the hard-coded parameters (like the block generation rate) or the basic primitives, e.g., the way proof of work is performed (a number of alternative proof of work implementations have been proposed using functions like scrypt [Per09], lyra2 [SJAA$^+$14] and others). However, more radical modifications are possible that alter the actual operation of the protocol.

One of the most notable such variants is the GHOST protocol, which was suggested by Sompolinsky and Zohar in [SZ15]. After the initial suggestion many cryptocurrencies using variants of the GHOST rule were proposed. The most popular among them, Ethereum has received substantial research attention [KMS$^+$16, KZZ16, Bon16, PK15, JKS16, Omo14].[1] Ethereum is essentially a Bitcoin-like system where transaction processing is Turing-complete and thus it can be used to implement any public functionality in a distributed way. Bitcoin-NG [EGSvR15] is another popular Bitcoin-like system relying on GHOST that separates blocks in two categories, namely key blocks and microblocks, reflecting the fact that transaction serialization and leader election may be separated.

Unfortunately, the security analysis of [SZ15] is not as general as [GKL15] (they analyze security against a specific double-spending attack), while the analysis of [GKL15] does not carry to the setting of GHOST. This is because the GHOST rule is a natural, albeit radical, reformulation of how each miner determines the main chain. In GHOST, miners adopt blocks in the structure of a *tree*. Note that in both Bitcoin and GHOST one can consider parties collecting all mined blocks in a tree data structure. However, while in Bitcoin the miners would choose the most difficult chain as the main chain, in GHOST, they will determine the chain by greedily following the "heaviest observed subtree." This means that for the same subtree, a Bitcoin miner and a GHOST miner may choose a completely different main chain. Furthermore, it means that the length of the main chain of honest parties does not necessarily increase monotonically (it may decrease at times) and thus a fundamental argument (namely that blockchains monotonically increase) that made the analysis of [GKL15] possible, does not hold anymore.

Our Results. We propose a new analysis framework for blockchain protocols focusing on trees of blocks as opposed to chains as in [GKL15]. Our framework enables us to argue about random variables on the trees of blocks that are formed by the participants. In our framework, we can express concepts like a node being **d-dominant**, which

[1]The GHOST proposal is currently abandoned by Ethereum.

means that the block corresponding to that node would be preferred by a margin of d compared to other sibling nodes according to a specified weight measure. This actually enables us to unify the description of Bitcoin and `GHOST` by showing they obey the same rule, but simply for a different weight measure.

Using our framework we then provide the first formal security analysis of `GHOST` against byzantine adversaries. Unlike Bitcoin, a transaction in `GHOST` is k-stable whenever the block that the transaction belongs to is the root of a subtree of at least k blocks. We prove that `GHOST` for this stability predicate implements a robust transaction ledger with a probability of error which drops exponentially in the security parameter over all executions of the protocol. We achieve this result, by a new methodology, that reduces the properties of the robust transaction ledger to a single lemma, that we call the *fresh block lemma* and is informally stated as follows.

> *Fresh Block Lemma*. (Informally) At any point of the execution and for any past sequence of s consecutive rounds, there exists an honest block mined in these rounds, that is contained in the chain of any honest player from this point on.

As we will demonstrate, the fresh block lemma is a powerful tool in the presence of an adversary: we show easily that the properties of the robust transaction ledger reduce to it in a black-box fashion.

In more detail, our proof strategy for persistence and liveness utilizes the fresh block lemma in the following (black-box) manner. In the case of persistence, it is sufficient to ensure that reporting the transaction as stable by any honest player implies that a sufficient amount of time has passed so that the fresh block lemma is applicable and has produced a block that is a descendant of the block that contains the transaction. Using the moderate hardness of proofs of work (specifically that they are *hard enough*) it is easy to translate from number of blocks in a subtree to actual running time in terms of rounds. It follows that the fresh block lemma applies and all honest parties will be mining on a subtree rooted at this fresh block for the remaining of the execution. As a result, the transaction will always be reported at the same position, as it belongs to the heaviest observed path for all of the honest parties.

In the case of liveness, we proceed in two steps. First, for our choice of u, in a time window lasting $\Theta(k)$ rounds, it will be ensured that the fresh block lemma can be applied once, implying that all honest parties will mine blocks in a subtree rooted by a common block that includes the transaction. Then, after another $\Theta(k)$ rounds, the honest parties will have accumulated enough honest blocks in this subtree to pronounce this transaction as stable. This latter statement requires again the moderate hardness of proofs of work but from the opposite perspective, i.e., that they are *easy enough*.

The above strategy provides an alternative proof methodology for establishing the properties of a robust transaction ledger compared to previous works that analyzed blockchain protocols, [GKL15], [KP15] and [PSS17] who reduced the properties of the robust transaction ledger to three other properties called common prefix, chain quality and chain growth. As such, the proof strategy itself may be of independent interest as it could be applicable to other blockchain variants, especially those that are using trees of blocks instead of chains of blocks as in Bitcoin in their chain selection rule.

Our results align with the original expectation that `GHOST` performs better than Bitcoin in terms of liveness, since our proven liveness parameter is superior to the liveness parameter for Bitcoin proven in [GKL15], under the same assumptions. However, we were not able to verify the claims that the protocol remains secure, even when the block generation rate increases considerably.

Limitations and directions for future research. Our analysis is in the standard Byzantine model where parties fall into two categories, those that are honest (and follow the protocol) and those that are dishonest and may deviate in an arbitrary (and coordinated) fashion as dictated by the adversary. It is an interesting direction for future work to consider the rational setting where all parties wish to optimize a certain utility function. Designing suitable incentive mechanisms, for instance see [Ler14] for a suggestion related to the `GHOST` protocol, is a related important consideration. In our analysis we do not take into account the fact that the number of parties changes dynamically and that the protocol calibrates the difficulty of the PoW instances to account for that; we note that this may open the possibility for additional attacks, say [Bah13], and we point to [GKL17] for a first analysis of Bitcoin in this setting. In principle, our results can be extended in the semi-synchronous model of [PSS17].

Organization. In section 6.2 we overview our security model, as well as the notation that we use for expressing the protocol. In section 6.3 we provide a description of the `GHOST` backbone protocol, introduce our new tree-based framework, and show that it satisfies the fresh block lemma, mentioned earlier. Finally, in section 6.4 we demonstrate that `GHOST` can be used to implement a robust transaction.

6.2 Preliminaries

We assume that we are in the synchronous q-bounded ROM setting with a fresh CRS and a static adversary, as defined in Section 2.2. We also use the same definitions about the format of blocks and blockchains as in Section 3.2.

6.3 The GHOST Backbone protocol

6.3.1 Protocol Description

In order to study the properties of the core Bitcoin protocol, the term *Backbone Protocol* was introduced in [GKL15]. At this level of abstraction we are only interested in properties of the blockchain, independently from the data stored inside the blocks. The main idea of the Bitcoin Backbone is that honest parties, at every round, receive new chains from the network and pick the longest valid one to mine. Then, if they obtain a new block (by finding a small hash), they diffuse their chain at the end of the round.

The same level of abstraction can also be used to express the GHOST protocol (Algorithm 16). The GHOST Backbone protocol, as presented in [SZ15], is based on the principle that blocks that do not end up in the main chain, should also matter in the chain selection process. In order to achieve this, parties store a tree of all mined blocks they have received, and then using the greedy heaviest observed subtree (GHOST) rule (Algorithm 17), they pick which chain to mine on.

At every round, parties update their tree by adding valid blocks sent by other parties. The same principle as Bitcoin applies; for a block to be added to the tree, it suffices to be a valid child of some other tree block. The adversary can add blocks anywhere he wants in the tree, as long as they are valid. Again, as in Bitcoin, parties try to extend the chains they choose by one or more blocks. Finally, in the main function, a tree of blocks is stored and updated at every round. If a party updates his tree, he diffuses it to all other parties.

The protocol is parameterized by three external functions $\mathrm{V}(\cdot), \mathrm{I}(\cdot), \mathrm{R}(\cdot)$ which are called: the input validation predicate, the input contribution function, and the chain reading function, respectively. $\mathrm{V}(\cdot)$ dictates the structure of the information stored in a valid chain, $\mathrm{I}(\cdot)$ determines the data that parties put in the blocks they mine, $\mathrm{R}(\cdot)$ specifies how the data in the blocks should be interpreted depending on the application.

Next, for completeness we present the remaining procedures of the GHOST back-

Algorithm 16 The GHOST backbone protocol, parameterized by the *input contribution function* I(·) and the *reading function* R(·). $\mathbf{x}_\mathcal{C}$ is the vector of inputs of all block in chain $\mathcal{C}$.

```
1:  $\mathcal{T} \leftarrow v_{root}$                          ▷ $\mathcal{T}$ is a tree. $v_{root}$ is the genesis block.
2:  $state \leftarrow \varepsilon$
3:  $round \leftarrow 0$
4:  while TRUE do
5:      $\mathcal{T}_{new} \leftarrow \mathsf{update}(\mathcal{T}$, blocks found in RECEIVE())
6:      $\tilde{\mathcal{C}} \leftarrow \mathtt{GHOST}(\mathcal{T}_{new})$
7:      $\langle state, x\rangle \leftarrow I(state, \tilde{\mathcal{C}}, round, \textsc{Input}(), \textsc{Receive}())$
8:      $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
9:      if $\tilde{\mathcal{C}} \neq \mathcal{C}_{\mathsf{new}}$ or $\mathcal{T} \neq \mathcal{T}_{\mathsf{new}}$ then
10:         $\mathcal{T} \leftarrow \mathsf{update}(\mathcal{T}_{\mathsf{new}}, \mathrm{head}(\mathcal{C}_{\mathsf{new}}))$
11:         DIFFUSE(head($\mathcal{C}_{\mathsf{new}}$))
12:     $round \leftarrow round + 1$
13:     if INPUT() contains READ then
14:         write $R(\mathbf{x}_\mathcal{C})$ to OUTPUT()
```

bone protocol. Function pow (see Algorithm 18), which has to do with block mining and is the same as the one defined in the Bitcoin Backbone, and function update (see Algorithm 19) which refers to the way the block tree is updated.

Finally, we assume that the CRS, denoted as v_{root} in Algorithm 16, is sampled from a high-min entropy distribution. It follows that the adversary will not be able to use any blocks that he computed before the round the honest parties were activated with overwhelming probability in κ. Moreover, as in Chapter 3, we assume that no two invocations of I(·) performed by honest parties return the same result, as formalized by the Input Entropy condition in [GKL15].

6.3.2 A Unified Description of Bitcoin and GHOST

Next, we introduce our new analysis framework for backbone protocols that is focusing on trees of blocks, and we show how the description of Bitcoin and GHOST can be unified. We will use the term node and block interchangeably from now on.

In GHOST, every player stores all blocks he receives on a tree, starting from a pre-shared block called the *genesis* block (or v_{root}). If we assume that the same information

Algorithm 17 The chain selection algorithm. The input is a block tree $\mathcal{T}$. The $|\cdot|$ operator corresponds to the number of nodes of a tree. By $C_1||C_2$ we denote the concatenation of chains C_1, C_2.

```
1: function GHOST($\mathcal{T}$)
2:     $B \leftarrow root(\mathcal{T})$
3:     if $children_{\mathcal{T}}(B) = \emptyset$ then
4:         return $B$
5:     else
6:         $B \leftarrow argmax_{B' \in children_{\mathcal{T}}(B)} |subtree_{\mathcal{T}}(B')|$
7:         return $B||$GHOST$(subtree_{\mathcal{T}}(B))$
```

is stored by Bitcoin, both protocols can be described in a unified language. We first define block trees (or just trees), that capture honest players' knowledge about valid blocks on different moments of some round.

Definition 94. We denote by $\mathcal{T}_r^P$ (resp. $\mathcal{T}_r$) the tree that is formed from the blocks that player P (resp. at least one honest player) has received up to round r. Similarly, $\hat{\mathcal{T}}_r$ is the tree that contains all blocks of $\mathcal{T}_r$ and all blocks mined by honest players at round r. For any tree T and block $B \in T$, we denote by $T(B)$ the subtree of T rooted on B.

Notice that, due to the fact that diffuses of honest players always succeed, blocks in $\hat{\mathcal{T}}_r$ are always in $\mathcal{T}_{r+1}^P$. Thus, for every honest player P it holds that:

$$\mathcal{T}_r^P \subseteq \mathcal{T}_r \subseteq \hat{\mathcal{T}}_r \subseteq \mathcal{T}_{r+1}^P$$

Intuitively, heavier trees represent more proof of work. However, there is more than one way to define the weight of a tree. For example, in Bitcoin the heaviest tree is the longest one. On the other hand, for GHOST a heavy tree is one with many nodes. We abstract this notion, by conditioning our definitions on a function w that assigns weights on trees. Deciding which tree has more proof of work, and thus which tree is favored by the chain selection rule, depends on w. Building on these ideas, we next present the notion of a node being d-dominant in some block tree, i.e., being d units heavier than its siblings on the tree.

Definition 95. Let w be a real-valued function defined on trees. For any tree $\mathcal{T}$ let $siblings(v)$ denote the set of nodes in $\mathcal{T}$ that share the same parent with node v. Node v is **d-dominant** in $\mathcal{T}$ (denoted by $\mathrm{Dom}_{\mathcal{T}}(v,d)$) iff

Algorithm 18 The *proof of work* function, parameterized by q, D and hash functions $H(\cdot), G(\cdot)$. The input is $(x, \mathcal{C})$.

```
 1: function pow($x, \mathcal{C}$)
 2:     if $\mathcal{C} = \varepsilon$ then                    ▷ Determine proof of work instance
 3:         $s \leftarrow 0$
 4:     else
 5:         $\langle s', x', ctr' \rangle \leftarrow \mathrm{head}(\mathcal{C})$
 6:         $s \leftarrow H(ctr', G(s', x'))$
 7:     $ctr \leftarrow 1$
 8:     $B \leftarrow \varepsilon$
 9:     $h \leftarrow G(s, x)$
10:     while $(ctr \leq q)$ do
11:         if $(H(ctr, h) < D)$ then
12:             $B \leftarrow \langle s, x, ctr \rangle$
13:             break
14:         $ctr \leftarrow ctr + 1$
15:     $\mathcal{C} \leftarrow \mathcal{C}B$                            ▷ Extend chain
16:     return $\mathcal{C}$
```

$$w(\mathcal{T}(v)) \geq d \wedge \forall v' \in siblings(v) : w(\mathcal{T}(v)) \geq w(\mathcal{T}(v')) + d$$

Next, note that the chain selection rule in the Bitcoin protocol can be described using the notion of the d-dominant node. Let $w(\mathcal{T})$ be the height of some tree $\mathcal{T}$. At round r, each player P starting from the root of his $\mathcal{T}_r^P$ tree, greedily decides on which block to add on his chain, by choosing one of its 0-dominant children and continuing recursively[2] (ties are broken based on time-stamp, or based on which block was received first). Moreovoer, the GHOST selection rule can be described in exactly the same way by setting w to be the number of nodes of the tree. Thus, we have a unified way for describing the chain selection rule in both protocols. Looking forward, in both protocols, a transaction will be k-stable if the weight of the subtree of the block that contains it is at least k, for the respective definitions of norm w.

Building upon this formalism we can describe the paths that fully informed honest

[2]This is exactly algorithm 17 with a minor modification. At line 6 the subtree $\mathcal{T}$ that is chosen maximizes $w(\mathcal{T})$.

Algorithm 19 The tree update function, parameterized by q, D and hash functions $H(\cdot), G(\cdot)$. The inputs are a block tree $\mathcal{T}$ and an array of blocks.

```
1: function update($\mathcal{T}$,$B$)
2:     foreach $\langle s,x,ctr \rangle$ in $\mathcal{T}$
3:     foreach $\langle s',x',ctr' \rangle$ in $B$
4:     if $((s' = H(ctr, G(s,x))) \wedge (H(ctr', G(x',ctr')) < D))$ then
5:         $children_{\mathcal{T}}(\langle s,x,ctr \rangle) = children_{\mathcal{T}}(\langle s,x,ctr \rangle) \cup \langle s',x',ctr' \rangle$    ▷ Add to the tree.
6:     return $\mathcal{T}$
```

players choose to mine at round r (denoted by HonestPaths(r)) in a quite robust way, thus showcasing the power of our notation.

$$\text{HonestPaths}(r) = \{p = v_{\text{root}} v_1 \ldots v_k | p \text{ is a root-leaf path in } r \text{ and } \forall i \in [k] : \text{Dom}_{\mathcal{T}_r}(v_i, 0)\}$$

Next, we present two key properties that both the Bitcoin and GHOST backbones satisfy. The first property states that by diffusing k blocks the adversary can decrease the dominance of some block at most by k. Intuitively, if the adversary's ability to mine new blocks is limited, then his influence over the block tree is also limited. On the other hand, the second property states that uniquely successful rounds increase the dominance of the nodes on the path from the root to the new block.

Proposition 96. For the Bitcoin and GHOST backbone protocols it holds that:

- if the adversary diffuses $k \le d$ new blocks at round $r-1$, then for every block $B \in \hat{\mathcal{T}}_{r-1}$ it holds that $\text{Dom}_{\hat{\mathcal{T}}_{r-1}}(B, d)$ implies $\text{Dom}_{\mathcal{T}_r}(B, d-k)$.
- if exactly one honest party computes a block at round r, and the newly mined block B extends a path in HonestPaths(r), then $\text{Dom}_{\hat{\mathcal{T}}_r}(B, 1)$ and for any block B' in the path from v_{root} to B it holds that $\text{Dom}_{\mathcal{T}_r}(B', d)$ implies $\text{Dom}_{\hat{\mathcal{T}}_r}(B', d+1)$.

We delay the proof of the proposition until the next subsection, where we present our security assumptions.

6.3.3 Analysis of the GHOST Backbone Protocol

In this subsection, we prove that the GHOST backbone protocol satisfies the fresh block lemma mentioned in Section 6.1. While for the rest of this section we assume that norm $w(\mathcal{T})$ is defined as the total number of nodes of tree $\mathcal{T}$, all results hold also for the Bitcoin norm.

We follow the notation and definitions of Section 3.3.2, for: successful and uniquely successful rounds, honest, adversarial and u.s. blocks, quantities $p, \alpha, \beta, \gamma, f$, random variables $X(\cdot), Y(\cdot), Z(\cdot)$. Additionally, We define $Z'(S)$ to be the number of *new* valid adversarial blocks, diffused by the adversary during round interval S, i.e., adversarial blocks that have not been diffused to any honest party before the first round of S.

Similarly to how Lemmas 10 and 15 follow from Assumption 1, the following lemmas and theorem follow from Assumption 6. For a summary of our notation we refer to Table 6.1.

Assumption 6. For $\delta \in (0,1)$, we assume that:

- $t < (1-\delta)(n-t)$; *(Honest majority)*
- $3f \leq \delta < 1$; *(Bounded block generation rate)*

Lemma 97. *Assume Assumption 6. It holds that* $\gamma \geq (1+\frac{\delta}{2})\beta$.

Definition 98. An execution is typical, if, $\kappa > \frac{7}{\delta}$ and for any set S of at least κ consecutive rounds, it holds that:

- $(1-\delta/8)\alpha|S| < X(S)$ and $(1-\delta/8)\gamma|S| < Y(S)$;
- $Z(S) < (1+\delta/9)\beta|S|$, and $Z(S)+X(S) \leq (1+\delta)f|S|$;
- no bad events happen for the RO.

Theorem 99. *Assume Assumption 6. An execution is typical with probability* $1-\mathsf{negl}(\kappa)$.

Lemma 100. *Assume a typical execution and Assumption 6. For any set of consecutive rounds* $S = i, \ldots, j$*, where* $|S| \geq \kappa$*, it holds that* $Y(S \setminus \{i\}) > Z(S)$.

Next, we show that proposition 96 holds in a typical execution.[3]

Proof of Proposition 96. The lemma follows from the fact that adding only one block in the tree reduces or increases the dominance of some block by at most 1, for both the Bitcoin and GHOST norms. For the first bullet, adding k blocks one by one on some tree, implies that the dominance of any node will reduce or increase by at most k. $\mathcal{T}_r$ includes $\hat{\mathcal{T}}_{r-1}$ and the k nodes broadcast by the adversary at round $r-1$, hence the dominance

[3]We require the execution to be typical to exclude pathological cases where the adversary for example creates cycles among blocks.

κ :	security parameter
D :	block difficulty parameter
n :	number of parties
t :	number of corrupted parties
q :	number of queries to the RO per round per party
β :	expected number of blocks computed by the adversary per round
α :	lower bound on the rate of successful rounds
γ :	lower bound on the rate of uniquely successful rounds
f :	upper bound on the rate of computed blocks
δ :	honest parties' advantage from Assumption 1

Table 6.1: *The parameters in our analysis. Parameters* $\kappa,n,t,D,q \in \mathbb{N}$, *while* $\alpha,\beta,\gamma,f,\delta \in \mathbb{R}$

of any node on $\hat{\mathcal{T}}_{r-1}$ will decrease by at most k. Note, that nodes in $Z'(\{1,\dots,r-2\})$ are included in $\hat{\mathcal{T}}_{r-1}$, since they have been received by honest parties, up to round $r-1$.

For the second bullet, notice that dominance increases only for blocks that get heavier. The only blocks that get heavier in this case are the ones in the path from the root to the newly mined block. Since these blocks are in HonestPaths(r), they are at least 0-dominant and so their dominance will further increase. Furthermore, the newly mined block is 1-dominant since it does not have any siblings.

□

In [GKL15], it was shown that the Bitcoin Backbone satisfies two main properties: common prefix and chain quality. However, another fundamental property needed for their proof, is that the chains of honest parties grow at least at the rate of successful rounds. This does not hold for GHOST. The reason is that, if an honest party receives a chain that is heavier than the one he currently has, he will select it, *even if it is shorter*. To reflect these facts, we develop an argument that is a lot more involved and leads to a power lemma that we call the "fresh block lemma".

First, we introduce a new notion, that of a path that all of its nodes are dominant up to a certain value. Intuitively, the more dominant a path is, the harder it gets for the adversary to stop honest parties from choosing it.

Definition 101. For $d > 0$, $\mathrm{p_{dom}}(r,d)$ is the longest path $p = v_{\mathrm{root}}v_1 \dots v_k$ in $\hat{\mathcal{T}}_r$ s.t.

$$p \neq v_{\mathrm{root}} \wedge \forall i \in [k] : \ \mathrm{Dom}_{\hat{\mathcal{T}}_r}(v_i, d)$$

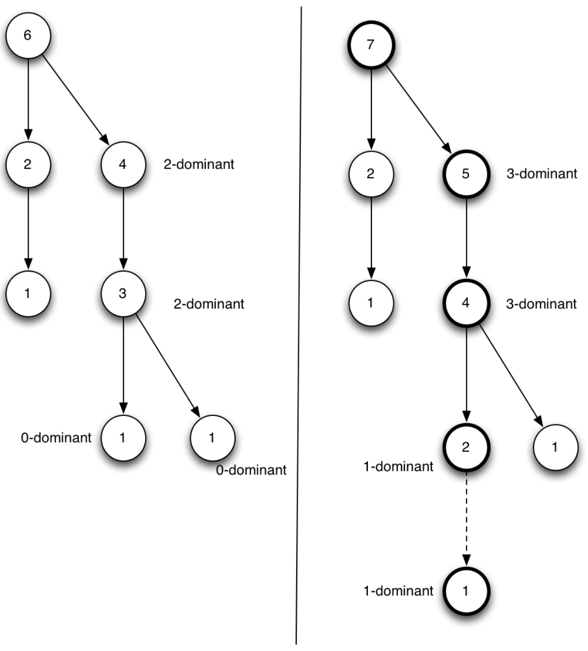


Figure 6.1: An example of the change in dominance after a uniquely successful round. The only nodes which increase their dominance are the ones in the path from the root to the newly mined block as stated in Proposition 96.

If no such path exists $\mathrm{p_{dom}}(r,d) = \perp$.

From the requirement that $d > 0$, it follows that the dominant path $\mathrm{p_{dom}}(r,d)$, if it is not $\perp$, will be unique.

In the next lemma, we show that unless the number of new blocks the adversary diffuses in a round interval is at least as big as the number of uniquely successful rounds that have occurred, an honest block mined in one of these rounds will be deep enough in the chains of honest parties. More specifically, for any sequence of m (not necessarily consecutive) uniquely successful rounds starting at some round r', no matter the strategy of the adversary, at round r there will be at least one honest block in $\mathrm{p_{dom}}(r,m-k)$ where k is the number of adversarial blocks that have been diffused during rounds $[r'-1,r-1]$.[4]

Lemma 102. *Assume a typical execution. Let $r_1 < .. < r_m$ be uniquely successful rounds from round r' until round r. If $Z'([r'-1,r-1]) < m$, then there exists an honest block B, mined in one of the u.s. rounds, such that B is in $\mathrm{p_{dom}}(r,m-Z'([r'-1,r-1]))$. Moreover, if the honest block mined at round r_1 is in $\mathrm{p_{dom}}(r_1,1)$ and $Z'([r_1,r-1]) < m$, then B is in $\mathrm{p_{dom}}(r,m-Z'([r_1,r-1]))$*

[4]We slightly abuse the notation, and use $[a,b]$, for integers $a \le b$, to denote the set $\{a,\dots,b\}$.

Proof. We are first going to prove two preliminary claims that show the effect of a uniquely successful round to $\mathrm{p_{dom}}$. The first claim shows that if a uniquely successful round s is not compensated accordingly by the adversary, a newly mined block will be forced into $\mathrm{p_{dom}}(s,1)$.

Claim 9. *Let round s be a uniquely successful round and B be the honest block mined at round s. If $Z'(\{s-1\})=0$, then $B \in \mathrm{p_{dom}}(s,1)$.*

Proof of Claim. First, notice that since the adversary does not broadcast any new block at round $s-1$, it holds that $\mathcal{T}_s$ is equal to $\mathcal{T}_s^P$, for any honest player P. Therefore, all nodes in the path from v_{root} to the parent of B are at least 0-dominant in $\mathcal{T}_s$ and thus this path is in HonestPaths(s). Since s is uniquely successful, all conditions of the second bullet of Proposition 96 are met, and thus it is implied that all nodes up to the newly mined block in $\hat{\mathcal{T}}_s$ are 1-dominant. It follows that $B \in \mathrm{p_{dom}}(s,1)$. ⊣

The second claim shows the effect of a uniquely successful round s to an existing $\mathrm{p_{dom}}(s-1,d)$ path. Notice that if the adversary broadcasts less than d blocks, the same nodes continue to be at least 1-dominant in the following round.

Claim 10. *Let round s be a uniquely successful round, B be the corresponding u.s. block, and $\mathrm{p_{dom}}(s-1,d) \neq \bot$, for some $d>0$. If $Z'(\{s-1\})<d$, then $\mathrm{p_{dom}}(s-1,d) \subseteq \mathrm{p_{dom}}(s,d+1-Z'(\{s-1\}))$. If $Z'(\{s-1\})=d$, then either $B \in \mathrm{p_{dom}}(s,1)$ or $\mathrm{p_{dom}}(s-1,d) \subseteq \mathrm{p_{dom}}(s,1)$ and B in $\hat{\mathcal{T}}_s$ is a descendant of the last node in $\mathrm{p_{dom}}(s-1,d)$.*

Proof of Claim. Let $Z'(\{s-1\})=k$. For the first item, suppose the adversary broadcasts $k<d$ blocks. Then, according to the first bullet of Proposition 96, the adversary can lower the dominance in $\mathcal{T}_s$ of nodes in $\mathrm{p_{dom}}(s-1,d)$ by at most k. Thus $\mathrm{p_{dom}}(s-1,d)$ will be a prefix of all the chains in HonestPaths(s). Now, since s is a uniquely successful round, the dominance in $\hat{\mathcal{T}}_s$ of all nodes in $\mathrm{p_{dom}}(s-1,d)$ will increase by one. Therefore $\mathrm{p_{dom}}(s-1,d) \subseteq \mathrm{p_{dom}}(s,d+1-k)$ and B will be a descendant of the last node in $\mathrm{p_{dom}}(s-1,d)$.

For the second item, suppose the adversary broadcasts $k=d$ blocks. If he does not broadcast all of these blocks to reduce the dominance in $\mathcal{T}_s$ of some node in $\mathrm{p_{dom}}(s-1,d)$, then $\mathrm{p_{dom}}(s-1,d)$ will be a prefix of all the chains in HonestPaths(s) and as in the previous case, $\mathrm{p_{dom}}(s-1,d) \subseteq \mathrm{p_{dom}}(s,d+1-k) \subseteq \mathrm{p_{dom}}(s,1)$ and B will be a descendant of the last node in $\mathrm{p_{dom}}(s-1,d)$.

Otherwise the adversary will reduce the dominance in $\mathcal{T}_s$ of at least one node in $\mathrm{p_{dom}}(s-1,d)$ to zero. If B is a descendant of the last node in $\mathrm{p_{dom}}(s-1,d)$, then again by Proposition 96 all nodes in $p_{\mathrm{dom}}(s-1,d)$ will be 1-dominant in $\hat{\mathcal{T}}_s$ and $p_{\mathrm{dom}}(s-1,d) \subseteq p_{\mathrm{dom}}(s,1) = p_{\mathrm{dom}}(s,d+1-d)$. If B is not a descendant of the last node in $p_{\mathrm{dom}}(s-1,d)$, then for the player P that mined this block it holds that $\mathcal{T}_s^P = \mathcal{T}_s$, he would have not mined a chain that does not contain $p_{\mathrm{dom}}(s-1,d)$ at round s otherwise. Therefore, P at round s was mining a chain that belonged to HonestPaths(s, v_{root}) and thus all nodes in the chain are at least 0-dominant in $\mathcal{T}_s$. But because s is a uniquely successful round, the dominance of all nodes in the chain that B belongs to, will increase by one and thus $B \in p_{\mathrm{dom}}(s,1)$. $\dashv$

Let B_i denote the honest block mined at round r_i. We are going to prove the lemma using induction on the number of uniquely successful rounds m.

For the base case, suppose $m=1$. Then, $Z'([r'-1,r-1]) = 0$, thus from the first claim $B_1 \in p_{\mathrm{dom}}(r_1,1)$. Moreover, all honest parties will mine new blocks on paths containing B_1 in the next rounds, hence $B_1 \in p_{\mathrm{dom}}(r,1)$. Thus, the base case is proved.

Suppose the lemma holds for a sequence of $m-1$ uniquely successful rounds. We take two cases. In the first case, $Z'([r'-1,r_m-2]) = m-1$. Hence, the adversary broadcasts no new blocks in the rest of the rounds. From the first claim for round r_m, it follows that $B_m \in p_{\mathrm{dom}}(r_m,1)$, and as before $B_m \in p_{\mathrm{dom}}(r,1)$.

In the second case, $Z'([r'-1,r_m-2]) < m-1$, which from the induction hypothesis implies that there exists some block B' mined by an honest party at one of the uniquely successful rounds in $\{r_i\}_{i\in[m-1]}$, where $B' \in p_{\mathrm{dom}}(r_m-1, m-1-Z'([r'-1,r_m-2]))$. By the second claim, if $Z'([r'-1,r_m-1]) = m-1$, then either $B_m \in p_{\mathrm{dom}}(r_m,1)$ or $B' \in p_{\mathrm{dom}}(r_m,1)$, and as before this condition extends to round r. Otherwise, $Z'([r'-1,r_m-1]) < m-1$, and thus $B' \in p_{\mathrm{dom}}(r_m, m-Z'([r'-1,r_m-1]))$. By a simple application of the second claim, it follows that $B' \in p_{\mathrm{dom}}(r, m-Z'([r'-1,r-1]))$.

For the second part of the lemma, note that the base hypothesis holds independently of the number of new blocks the adversary diffused before round r_1. Hence, exactly by the same arguments the lemma follows.

□

The fresh block lemma is stated next. Informally, it states that in a typical execution, at any point in time, in any past sequence of s consecutive rounds, at least one honest block was mined and is permanently inserted in the chain that every honest party adopts.

Lemma 103. (Fresh Block Lemma) *Assume a typical execution and Assumption 6. For any rounds r,s of the execution where $r \geq s \geq \kappa$, it holds that there exists a block* *mined by an honest party after round $r-s$, that is contained in the chain which any honest party adopts after round r.*

Proof. The difficulty of proving this lemma comes from the fact that in GHOST, the length of the chains of honest parties are not always strictly increasing. That is, honest parties may switch from a longer to a shorter chain. Monotonicity, allows us to prove many useful things; for example that the adversary cannot use very old blocks in order to maintain a fork as in [GKL15].

To overcome this difficulty, we will show that in order for honest parties to adopt a chain that is on a different branch than the one they currently are, the adversary must broadcast as many blocks as the ones that where mined on uniquely successful rounds on this branch. Hence, forcing honest parties to older branches will be costly, as it would imply that the adversary must compute a number of blocks comparable to the number of u.s. rounds. Moreover, after κ rounds, this will be impossible due to Lemma 100. Finally, if all honest parties stay on one branch, using Lemma 102, we can argue that at least one honest block will enter their chains.

We are first going to prove two preliminary claims. First, we show that as long as from some round r and afterwards the adversary diffuses less blocks than the total number of uniquely successful rounds, the chain that any honest party adopts after round r extends $\mathrm{p_{dom}}(r, Y([r]) - Z'([r-1]))$. Formally:

Claim 11. *For some round s_1, assume that for all rounds $s_2 \geq s_1$ it holds that $Z'([s_2]) < Y([s_2])$. Then, the chain that any honest party adopts after round s_1 extends* $\mathrm{p_{dom}}(s_1, Y([s_1]) - Z'([s_1-1]))$.

Proof of Claim. Since $Y([s_1]) > Z'([s_1-1])$, from Lemma 102 it follows that

$$p = \mathrm{p_{dom}}(s_1, Y([s_1]) - Z'([s_1-1])) \neq \perp$$

As long as the number of blocks that the adversary diffuses at some round $i \geq s_1$ are less than the dominance of the nodes in p in $\hat{\mathcal{T}}_i$, all honest parties at round $i+1$ will adopt chains containing p. Thus, uniquely successful rounds will further increase the dominance of these nodes. By our assumption, $Z'([i]) < Y([i])$, for all $i \geq s_1$, hence the nodes in p are at least 1-dominant in $\mathcal{T}_i^P$, where P is an honest party. The claim follows. $\dashv$

Next, we show that if successive u.s. rounds occur such that the blocks mined are on different branches, then the adversary must broadcast an adequate number of blocks, as specified below.

Claim 12. *Let $s_1 < s_2 < ... < s_m$ be u.s. rounds, such that for any $k \in [m-1]$, s_{k+1} is the first u.s. round after s_k, where the honest block mined at this round is not a descendant of the honest block mined in round s_{k-1}. It holds that either $Z'([s_1-1, s_m-1]) > Y([s_1, s_m-1])$ or $Z'([s_1-1, s_m-1]) = Y([s_1, s_m-1])$ and the honest block mined at round s_m will be in $\mathrm{p_{dom}}(s_m, 1)$.*

Proof of Claim. Let $B_1, .., B_m$ denote the honest blocks mined at rounds $s_1, .., s_m$, respectively. We are going to prove the claim using induction. For the base case, suppose that $m = 2$. For the sake of contradiction, assume that the claim does not hold, and thus $Z'([s_1-1, s_2-1]) \leq Y([s_1, s_2-1])$, and $Z'([s_1-1, s_2-1]) = Y([s_1, s_2-1])$ implies that $B_2 \notin \mathrm{p_{dom}}(s_2, 1)$. We take two cases. In the first one, assume that $Z'([s_1-1, s_2-1]) < Y([s_1, s_2-1])$. By the definition of s_2, the honest blocks mined on all u.s. rounds up to round $s_2 - 1$ are descendants of B_1. From Lemma 102 at least one honest block B computed in one of the u.s. rounds in $[s_1, s_2-1]$ will be in $\mathrm{p_{dom}}(s_2-1, Y([s_1, s_2-1]) - Z'([s_1-1, s_2-2]))$. From our hypothesis, the adversary will broadcast less than $Y([s_1, s_2-1]) - Z'([s_1-1, s_2-2])$ blocks at round $s_2 - 1$. Hence, from Claim 10, it is impossible for B_2 not to be a descendant of B, and thus of B_1, which is a contradiction.

For the second case, assume that $Z'([s_1-1, s_2-1]) = Y([s_1, s_2-1])$ and $B_2 \notin \mathrm{p_{dom}}(s_2, 1)$. If at round $s_2 - 1$ the adversary does not broadcast any new block, from Claim 9 of Lemma 102 B_2 will be in $\mathrm{p_{dom}}(s_2, 1)$, which is a contradiction. Otherwise, the adversary broadcasts exactly $Y([s_1, s_2-1]) - Z'([s_1-1, s_2-2]) > 0$ blocks at round $s_2 - 1$, and thus $Y([s_1, s_2-1]) > Z'([s_1-1, s_2-2])$. From Lemma 102, there exists some block B mined at a u.s. round such that $B \in \mathrm{p_{dom}}(s_2-1, Y([s_1, s_2-1]) - Z'([s_1-1, s_2-2]))$. It follows that B_2 will be a descendant of B, which is a contradiction. Hence, the base case proven.

Next, suppose that the lemma holds until round s_k. We are going to show that it also holds for s_{k+1}. We take three cases. In the first case, assume that $Z'([s_1-1, s_{k+1}-1]) > Y([s_1, s_{k+1}-1])$. The claim trivially holds. For the second one, assume that $Z'([s_1-1, s_{k+1}-1]) = Y([s_1, s_{k+1}-1])$. If the adversary does not broadcast any block at round $s_{k+1} - 1$, then as before it follows that $B_{k+1} \in \mathrm{p_{dom}}(s_{k+1}, 1)$, as is required by the claim. Otherwise, based on the inductive hypothesis we have two cases. In the first one, $Z'([s_1-1, s_k-1]) > Y([s_1, s_k-1])$, which implies that $Z'([s_k, s_{k+1}-2]) < Y([s_k, s_{k+1}-1]) - 1$. Hence, $Z'([s_k, s_{k+1}-2]) < Y([s_k+1, s_{k+1}-1])$, and by Lemma 102 it follows that B_{k+1} should be a descendant of B_k, which is a contradiction. For the second one, assume that $Z'([s_1-1,s_k-1]) = Y([s_1,s_k-1])$ and $B_k \in \mathrm{p_{dom}}(s_k,1)$. It is implied that $Z'([s_k,s_{k+1}-2]) < Y([s_k,s_{k+1}-1])$, and again by Lemma 102 we reach the same contradiction. Hence, it should hold that $B_{k+1} \in \mathrm{p_{dom}}(s_{k+1},1)$, as the claim states.

For the last case, assume that $Z'([s_1-1,s_{k+1}-1]) < Y([s_1,s_{k+1}-1])$. Again, by the inductive hypothesis it follows that either $Z'([s_k,s_{k+1}-1]) < Y([s_k+1,s_{k+1}-1])$ or $Z'([s_k,s_{k+1}-1]) < Y([s_k,s_{k+1}-1])$ and $B_k \in \mathrm{p_{dom}}(s_k,1)$, which as before contradicts the definition of s_{k+1}. Hence, this case is impossible, and by induction the claim follows.

$\dashv$

Next, we observe that Lemma 102 as well as Claim 11 and 12 can be applied on a subtree of the block tree, as long as all u.s. blocks mined after the round the root of the subtree was mined are part of the subtree.

Observation 104. Let B be an honest block computed at round s_1, that is in the chains adopted by all honest parties after round s_2. Also, assume that all u.s. blocks mined after round s_1 are descendants of B. Then the following hold:

1. Regarding applications of Lemma 102 and Claim 12 on the *subtree* of the block tree rooted on B, we can ignore all blocks that the adversary has mined up to round s_1-1.

2. Regarding applications of Claim 11 after round s_2, we can ignore all blocks that the adversary has mined up to round s_1-1.

To see why the observation holds consider the following. Firstly, the adversary sees block B for the first time at round s_1, hence all blocks that the adversary mines before round s_1 cannot be descendants of B. Moreover, blocks that are not descendants of B do not affect the validity of Lemma 102 and Claim 12 on the subtree of the block tree rooted on B, since they do not affect the dominance of the nodes of the subtree rooted at B. Finally, regarding the second point, consider the dominant path at any round $s_3 > s_2$ in the subtree that is rooted on B. This path can be extended up to the root node, since, by our assumption, B is in the chains adopted by all honest parties after round s_2.

We are now ready to prove the lemma. We will use the convention that block B_i is mined at round r_i for the rest of this proof. For the sake of contradiction, assume that the lemma does not hold, and no honest block mined after round $r-s$ is contained in the chains of all honest parties on all rounds after r. Let B_0 be the most recent honest (or genesis) block that (i) is in the chains that all honest parties have adopted after round r, and (ii) all u.s blocks mined after round r_0 are descendants of B_0. This block is well defined, since in the worst case it is the genesis block. By our assumption, it follows that $r_0 <= r - s$.

For the rest of the proof, we are going to apply Lemma 102, and Claims 11 and 12 on the subtree rooted on B_0, for which due to Observation 104, we can assume that $Z'([r_0, r']) \leq Z([r_0, r'])$, for any $r' \geq r$. Moreover, since $r - r_0 \geq s \geq \kappa$, by the typicality of the execution it holds that $Z'([r_0, r']) < Y([r_0 + 1, r'])$, for any $r' \geq r$. Hence, by Lemma 102 and Claim 11 there exists at least one honest block B_1, mined in a u.s. round and contained in the chains of all honest parties after round r. W.l.o.g. let B_1 be the block corresponding to the earlier such round. By the definition of B_0, B_1 is a descendant of B_0. It should hold that $r_1 \leq r - s$, otherwise B_1 satisfies the conditions of the lemma. Moreover, there should exist some u.s. block B_2, mined after B_1, that is not a descendant of B_1. Otherwise, B_1 would satisfy the definition of B_0, which is a contradiction. Let B_2 be the earliest mined such block.

Now, let block B'_0, mined at round r'_0, be the first u.s. block mined after round r_0. If $B'_0 \neq B_1$, it necessarily holds that it is also not a descendant of B_1. Hence, in this case, there exists a sequence of u.s. rounds $r'_0 < \ldots < r_1 < r_2$, such that we can apply Claim 12; note that no u.s. block exists between B_0 and B_1, by the definition of B_1. Hence, it follows that either $Z'([r'_0 - 1, r_2 - 1]) > Y([r'_0, r_2 - 1])$ or $Z'([r'_0 - 1, r_2 - 1]) = Y([r'_0, r_2 - 1])$ and $B_2 \in \mathrm{p_{dom}}(r_2, 1)$. Note also, that by the definition of r'_0, it holds that $Y([r'_0, r_2 - 1]) = Y([r_1 + 1, r_2 - 1])$. In the other case, where $B'_0 = B_1$, we get exactly the same result by applying Claim 12 for the sequence r_1, r_2.

Next, we will show that we can apply the same argument repeatedly, until we reach a contradiction. By what we have said previously, it follows that either $Z'([r_2, r']) < Y([r_2, r']) + 1$ or $Z'([r_2, r']) < Y([r_2, r'])$ and $B_2 \in \mathrm{p_{dom}}(r_2, 1)$, for any $r' \geq r$. Again, by Lemma 102 and Claim 11, there exists at least one honest block B, mined in a u.s. round after round r_2, and contained in the chains of all honest parties on and after round r. Obviously B is a descendant of B_1, and is computed on or before round $r - s$, otherwise if would refute our initial assumption. Let B_3 be the first u.s. block in the subchain from B_1 to B, computed after round r_2. This block is well defined by our previous observation.

Again, in order for B_3 to not satisfy the definition of B_0, there must exist some u.s. block B_4 that is was mined after B_3, and is not its descendant. We can apply Claim 12 for the sequence of u.s. rounds starting from r'_0, and ending at r_4. Each time we repeat this argument the round sequence for which we can apply Claim 12 increases. Hence, after κ repetitions, it must be the case that the adversary computed more blocks that the number of uniquely successful rounds, in a round interval of size at least κ. This contradicts the typicality of the execution. Thus, there exists some honest block mined after round $r-s$, that is in the chains of all honest parties after round r.

□

6.4 Implementing a Transaction Ledger

In this section, we show how we implement a robust public transaction ledger, using the functions $\mathrm{V}, \mathrm{R}, \mathrm{I}$ defined in Table 3.2 of Section 3.4, on top of the GHOST backbone. For our protocol, a transaction is considered k-stable if the subtree rooted at the block containing the transaction is of size at least k. We call the resulting protocol, $\Pi_{\mathsf{PL}}^{\mathtt{GHOST}}$.

Having established, in the previous section, that every s rounds a fresh and honest block is inserted permanently in the chains of all parties, we are in a position to prove the main properties of a robust transaction ledger. Regarding liveness, we make the following two observations: After κ rounds, from the time a transaction was issued, an honest block that contains this transaction will permanently be part of the chain of any party, and in about $k/pq(n-t)$ rounds, honest parties mine k blocks. Hence, by waiting for a total of $\kappa + k/pq(n-t)$ rounds, honest parties will mine enough blocks so that this transaction becomes stable. For the rest of this section, we extend the notion of a typical execution, to include the event where in $s \geq \kappa$ rounds, honest parties mine at least $(1-\delta)\cdot s \cdot pq(n-t)$ blocks. Typical executions continue to happen with overwhelming probability, due to the Chernoff bound.

Lemma 105 (Liveness)**.** *Assume a typical execution and Assumption 6. Further, assume oracle* Txgen *is unambiguous. Protocol* $\Pi_{\mathsf{PL}}^{\mathtt{GHOST}}$ *satisfies Liveness with wait time* $u = \kappa + \frac{k}{(1-\delta)pq(n-t)}$ *rounds and depth parameter k, for* $k \geq \kappa$*.*

Proof. We prove that, assuming all honest parties receive as input the transaction tx for at least u rounds, any honest party at round r with chain $\mathcal{C}$ will report tx as stable. By Lemma 103, it follows that there exists an honest block mined during rounds $[r-u, r-u+k]$, such that all honest parties have this block in the chains they mine from round $r-u+k$ and onward. W.l.o.g., this block contains tx. All blocks that honest parties mine during the remaining $\frac{k}{(1-\delta)pq(n-t)}$ rounds will be descendants of this block. Due to the typicality of the execution, the honest parties will mine at least k blocks in this round interval, and thus the lemma follows with the desired probability.

□

The liveness property of the Bitcoin protocol in [GKL15] is proven in a similar way, and under the same assumptions. First, it is shown that some honest block containing the target transaction becomes part of the chains of all honest parties. Then, after k successful rounds occur, we are sure that the transaction is stabilized. This line of thought, implies that the liveness parameter there, depends on k/α, where α is the expected rate of successful rounds, instead of $k/pq(n-t)$. Remember now, that it holds that $\alpha < pq(n-t)$, due to the distributed way at which honest parties compute blocks. The difference between the two quantities grows as $pq(n-t)$ increases, i.e., as the block generation rate grows, it is more probable that two honest parties compute a block at the same round. Hence, we conclude, that the liveness parameter proved for GHOST here, is better than the one proved for Bitcoin in [GKL15].

Regarding the second property, Persistence, it is implied by the fact that when a party reports a transaction as stable for the first time, due to the upper bound f on the block production rate, more than κ rounds will have passed from the round the block containing the transaction was mined. Thus, there exists another honest block descending it, that has been permanently added to the chain of all honest parties, which implies the desired property.

Lemma 106 (Persistence)**.** *Assume a typical execution and Assumption 6. Protocol $\Pi_{\mathsf{PL}}^{\mathtt{GHOST}}$ satisfies Persistence with parameter k, for $k \geq 2\kappa f$.*

Proof. Let B be the block that contains transaction tx, that the honest party P reported in its chain $\mathcal{C}$ as stable at round r. We will argue that B must have been computed before round $r-k$, and thus by Lemma 103 all honest parties will report it in the same position in their chains.

For the sake of contradiction, assume that B was computed after round $r - \frac{k}{(1+\delta)f}$. The total number of blocks that descend B at round r, is at most $X([r - \frac{k}{(1+\delta)f}, r-1]) + Z[r - \frac{k}{(1+\delta)f}, r-1]$, which is smaller than $\frac{k}{(1+\delta)f}(1+\delta)f = k$. Hence, it must be that B was computed on or before round $r - \frac{k}{(1+\delta)f}$. Note now, that $\frac{k}{(1+\delta)f} > \kappa$, hence by an application of Lemma 103, there exists an honest block B' mined on and after round $r - \frac{k}{(1+\delta)f}$, that will be in the chains of all honest parties from round r and onward. B must be an ancestor of B', since they are both part of $\mathcal{C}$, and it was mined at a later round. Hence, all honest parties will report B and tx at the same position as P from round r and onward. Persistence follows.

□

Theorem 107. *If Assumption 6 holds, then protocol* $\Pi_{\mathsf{PL}}^{\mathtt{GHOST}}$ *implements a robust transaction ledger with wait time* $u = k + \frac{k}{(1-\delta)\alpha}$ *and depth parameter* k*, for* $k \geq 2\kappa f$*, with overwhelming probability in* κ.

As a final note, Lemma 103 is sufficient to prove Persistence and Liveness in a black-box way. Compared to the approach of [GKL15], that was further expanded in [KP15] and [PSS17], only one property, instead of three, of the underlying "backbone" protocol suffices in order to get a robust public transaction ledger in a black-box manner. On the other hand, the three properties described in these works, common-prefix, chain quality and chain growth, also serve as metrics of the efficiency of the underlying mechanism and provide more information than the fresh block lemma.

Chapter 7

Concluding Remarks

In this thesis a number of foundational issues of PoW-based blockchain protocols were investigated, with the hope of a better understanding of this technology. Limitations and future directions of this work were presented in the individual chapters. We conclude by presenting how some of the most interesting ideas of the thesis were developed, showcasing among others that scientific research is a non-linear process.

First, we focus on the creation of the Bootstrapped Bitcoin backbone protocol of Chapter 3. Before even starting to think about the bootstrapping problem, we performed the security analysis of the `GHOST` protocol, presented in Chapter 6, which led to the formulation of Proposition 96; essentially, an abstracted view of a set of conditions that allow for agreement in a probabilistically changing weighted block tree. Later, when we tried to tackle the bootstrapping problem explored in Chapter 3, we applied the same thinking by adjusting the weights of genesis and non-genesis blocks to satisfy the conditions of Proposition 96. In the process of doing so, we understood that the longest chain selection rule remains robust in the presence of bounded party-dependent noise on chain weights.

Another important point of this thesis, is the use a computational randomness extractor in Construction 2 of Chapter 5. Initially, following a top-down approach, we formulated the information theoretic runtime independence property in Chapter 4, in order to ensure that the distribution of uniquely successful rounds is sufficiently concentrated. Later on, when we tried to reduce the security of our protocol to the hardness of the ISP in Lemma 84 of Chapter 5, we noticed that we need to be able to efficiently simulate the honest parties' runtime distribution, thus leading to the formulation of the (computational) next-problem simulatability property. After some time, we understood that this property was also sufficient to guarantee the strong concentration properties we needed. Putting everything together, led us to the idea of using a computational extractor in Construction 2, a fairly realistic hash function property.

Finally, we discuss the development of the idea of using a witness malleable construction in Chapter 5. At some point in our research journey, we tried to implement a transaction ledger assuming the existence of iterated sequential functions, such as the one based on repeated squaring in a composite order group. This attempt was not fruitful, and one of the problems we had was that our PoW construction was witness malleable, thus allowing the adversary to generate witnesses quite cheaply. Later, when we were developing Lemma 84 mentioned earlier, we faced the problem that the reduction should be able to generate some witnesses cheaply. Our previous experience, made us realize that the problem could be solved if the ISP was witness malleable. Finally, we though of an easy way to obtain this property in Construction 2, by reversing Bitcoin's ISP construction and thus retaining some of its hardness properties as described in Lemma 92.

Appendix A

Mathematical Background

A.1 Inequalities

Throughout this thesis we make use of the following well-known inequalities:

Proposition 108. (Bernoulli) $\forall x \in \mathbb{R}^*, n \in \mathbb{N} : (x > -1) \wedge (n > 1) \Rightarrow 1 + xn < (1+x)^n$

Proposition 109. $\forall x \in \mathbb{R} : 1 - x \leq e^{-x}$

A.2 Probabilistic Inequalities

We remind the reader of some basic inequalities from probability theory that are used in this thesis. We assume some level of familiarity with discrete probability theory, e.g., see [Sho09].

Let Ω be a sample space, and $P : \Omega \to [0,1]$ a probability distribution defined on Ω, that satisfies the property

$$\sum_{\omega \in \Omega} P(\omega) = 1.$$

An event is defined as a subset E of Ω. It holds that the probability of event E is

$$\Pr[E] := \sum_{\omega \in E} P(\omega).$$

Let E_1, E_2 be events on Ω. We denote by $E_1 \wedge E_2$ (resp. $E_1 \vee E_2$) the conjunction (resp. disjunction) of these two events. It holds that:

Proposition 110. (Union Bound) $\Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2]$

A random variable is defined as a function $X : \Omega \to S$, where S is some set. For any $s \in S$, the event "X is equal to s" is equivalent to the event $\{\omega \in \Omega | X(\omega) = s\}$. Hence:

$$\Pr[X = s] := \sum_{\omega \in X^{-1}(s)} P(\omega).$$

If $S \subseteq \mathbb{R}$, the expected value $\mathbb{E}[X]$ of X is defined as:

$$\mathbb{E}[X] := \sum_{\omega \in \Omega} X(\omega) \cdot P(\omega).$$

We say that a finite set of random variables $\{X_i : \Omega \to S_i\}_{i \in I}$ is mutually independent iff for every $(s_i)_{i \in I}$, where $s_i \in S_i$, and any $J \subseteq I$, it holds that:

$$\Pr[\bigwedge_{i \in J} X_i = s_i] = \prod_{i \in J} \Pr[X_i = s_i].$$

We make extensive use of the Chernoff bound.

Proposition 111. (Chernoff bound) Let $\{X_1, \ldots, X_k\}$ be a family of i.i.d. random variables, where $\Pr[X_i = 0] = p$ and $\Pr[X_i = 1] = 1 - p$, for $p \in (0,1)$. Let $X = \sum_{i=1}^{k} X_i$. Then, it holds that:

- $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{\delta^2 \mathbb{E}[X]}{2+\delta}}$, for all $\delta > 0$.
- $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{\delta^2 \mathbb{E}[X]}{2}}$, for $\delta \in (0,1)$

Appendix B

Cryptographic Primitives

In this section, we recite the definitions of cryptographic primitives that we are going to use in this thesis.

Collision resistance. We make use of the following notion of security for cryptographic hash functions [Dam87]:

Definition 112. 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 $\mathcal{H}$,

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

Randomness extractors. We make use of the notion of *weak* computational randomness extractors, as formalized in [DSGKM12].

Definition 113. An extractor is a family of functions $\mathsf{Ext} = \{\mathsf{Ext}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{d(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$, where $n(\cdot), d(\cdot)$ and $m(\cdot)$ are polynomials. The extractor is called weak $k(\cdot)$-computational if Ext_λ is PPT, and for all efficiently samplable probability ensembles $\{X_\lambda\}_\lambda$ with min-entropy $k(\lambda)$:

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

Robust non-interactive zero-knowledge. We make use of the following composable notion of non-interactive zero-knowledge, introduced in [SCO$^+$01].

Definition 114. Given an NP relation R, let $L = \{x : \exists w \text{ s.t. } R(x,w) = 1\}$. $\Pi = (q, P, V, S = (S_1, S_2), E))$ is a *robust NIZK argument* for L, if $P, V, S, E \in PPT$ and $q(\cdot)$ is a polynomial such that the following conditions hold:

1. Completeness. For all $x \in L$ of length λ, all w such that $R(x,w) = 1$, and all $\Omega \in \{0,1\}^{q(\lambda)}$, $\mathsf{V}(\Omega, x, \mathsf{P}(\Omega, w, x))] = 1$.

2. Multi-Theorem Zero-knowledge. For all PPT adversaries $\mathcal{A}$, we have that $\textsc{Real}(\lambda) \approx \textsc{Sim}(\lambda)$, where

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

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

and $\mathsf{S}_2'(\Omega, x, w, tk) \overset{def}{=} \mathsf{S}_2(\Omega, x, tk)$ if $(x,w) \in R$, and outputs `failure` if $(x,w) \notin R$.

3. Extractability. There exists a PPT algorithm E such that, for all PPT $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); (x,\pi) \leftarrow \mathcal{A}^{\mathsf{S}_2(\Omega,\cdot,tk)}(\Omega); w \leftarrow \mathsf{E}(\Omega,(x,\pi),tk): \\ R(x,w) \neq 1 \wedge (x,\pi) \notin \mathcal{Q} \wedge \mathsf{V}(\Omega, x, \pi) = 1\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{Q}$ contains the successful pairs (x_i, π_i) that $\mathcal{A}$ has queried to S_2.

As in [FMNV14], 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 115 ([SCO+01])**.** *Assuming trapdoor permutations and a dense cryptosystem exist, robust NIZK arguments exist for all languages in $\mathcal{NP}$.*

Iterated sequential functions. We recite the hardness definition introduced in [BBBF18]:

Definition 116. $f : X \to Y$ 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 $(1-\varepsilon)\cdot t$ 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 117. 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.

Bibliography

[ACC$^+$17] Elli Androulaki, Christian Cachin, Angelo De Caro, Alessandro Sorniotti, and Marko Vukolic. Permissioned blockchains and hyperledger fabric. *ERCIM News*, 2017(110), 2017.

[AD15] Marcin Andrychowicz and Stefan Dziembowski. Pow-based distributed cryptography with no trusted setup. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 379–399. Springer, 2015.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[AJK05] James Aspnes, Collin Jackson, and Arvind Krishnamurthy. Exposing computationally-challenged Byzantine impostors. Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer Science, July 2005.

[AT17] Joël Alwen and Björn Tackmann. Moderately hard functions: Definition, instantiations, and applications. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 493–526, 2017.

[Bac97] Adam Back. Hashcash. http://www.cypherspace.org/hashcash, 1997.

[Bac00] Adam Back. Hashcash-amortizable publicly auditable cost functions. *Early draft of paper*, 2000.

[Bac02] Adam Back. Hashcash–a denial of service counter-measure, 2002.

[Bah13] Lear Bahack. Theoretical Bitcoin attacks with less than half of the computational power (draft). Cryptology ePrint Archive, Report 2013/868, 2013. `http://eprint.iacr.org/`.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 757–788, 2018.

[BCD$^+$14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 514–523, 1996.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 394–403, 1997.

[Ben83] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 17-19, 1983*, pages 27–30. ACM, 1983.

[BGJ$^+$16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 345–356. ACM, 2016.

[BGK$^+$19] Christian Badertscher, Peter GazÌŇi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros chronos: Permissionless clock synchronization via proof-of-stake. Cryptology ePrint Archive, Report 2019/838, 2019. `https://eprint.iacr.org/2019/838`.

[BGM$^+$18] Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? A rational protocol design treatment of bitcoin. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 34–65, 2018.

[BHMN17] Iddo Bentov, Pavel Hub'avcek, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptology ePrint Archive*, 2017:300, 2017.

[Bit] Bitcoinwiki. Genesis block. https://en.bitcoin.it/wiki/Genesis_block.

[BJL17] Mihir Bellare, Joseph Jaeger, and Julia Len. Better than advertised: Improved collision-resistance guarantees for md-based hash functions. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 891–906, New York, NY, USA, 2017. ACM.

[BKT$^+$19] Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia C. Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 585–602, 2019.

[BL13] Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *Advances in Cryptology*

- ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II, pages 321–340, 2013.

[BMTZ17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 324–356, 2017.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 236–254, 2000.

[Bon16] Joseph Bonneau. Ethiks: Using ethereum to audit a CONIKS key transparency log. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, pages 95–105, 2016.

[Bor96] Malte Borderding. Levels of authentication in distributed agreement. In Özalp Babaoglu and Keith Marzullo, editors, *Distributed Algorithms, 10th International Workshop, WDAG '96, Bologna, Italy, October 9-11, 1996, Proceedings*, volume 1151 of *Lecture Notes in Computer Science*, pages 40–55. Springer, 1996.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Maurer [Mau96], pages 399–416.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Confer-*

ence, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I, pages 789–819, 2018.

[BSCG$^+$15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE, 2015.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[CBPS10] Bernadette Charron-Bost, Fernando Pedone, and André Schiper. Replication. *LNCS*, 5959:19–40, 2010.

[CCGZ16] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 240–269, 2016.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 91–122. Springer, 2018.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 19–40, 2001.

[CFF$^+$05] Jeffrey Considine, Matthias Fitzi, Matthew K. Franklin, Leonid A. Levin, Ueli M. Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *J. Cryptology*, 18(3):191–217, 2005.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CM19] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, Berlin, Heidelberg, 2002.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, pages 203–216, 1987.

[Dam89] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.

[DGH$^+$04] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the cbc, cascade and hmac modes. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 494–510, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[DLS88] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

[DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology*

Conference on Advances in Cryptology, CRYPTO ’92, pages 139–147, London, UK, UK, 1993. Springer-Verlag.

[Dod05] Yevgeniy Dodis. On extractors, error-correction and hiding all partial information. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security, 2005.*, pages 74–79. IEEE, 2005.

[Dou02a] John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony 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.

[Dou02b] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS ’01, London, UK, UK, 2002.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[DSGKM12] Dana Dachman-Soled, Rosario Gennaro, Hugo Krawczyk, and Tal Malkin. Computational extractors and pseudorandomness. In Ronald Cramer, editor, *Theory of Cryptography*, pages 383–403, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[DSU04] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4):372–421, 2004.

[EFL17] Lisa Eckey, Sebastian Faust, and Julian Loss. Efficient algorithms for broadcast and consensus based on proofs of work. *IACR Cryptology ePrint Archive*, 2017:915, 2017.

[EGSvR15] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. *CoRR*, abs/1510.02037, 2015.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th In-*

ternational Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers, pages 436–454, 2014.

[FG10] Pierre Fraigniaud and George Giakkoupis. On the bit communication complexity of randomized rumor spreading. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pages 134–143, 2010.

[FGKR18] Matthias Fitzi, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. Cryptology ePrint Archive, Report 2018/1119, 2018. `https://eprint.iacr.org/2018/1119`.

[Fit03] Matthias Fitzi. *Generalized communication and security models in Byzantine agreement.* PhD thesis, ETH Zurich, Zürich, Switzerland, 2003.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 465–488, 2014.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*. Springer, 1986.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *Foundations of Computer Science (FOCS 2003).* IEEE Computer Society, 2003.

[GK18] Juan A. Garay and Aggelos Kiayias. SoK: A consensus taxonomy in the blockchain era. *IACR Cryptology ePrint Archive*, 2018:754, 2018.

[GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 658–668, 2007.

[GKL14] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. *IACR Cryptology ePrint Archive*, 2014:765, 2014.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, 2015.

[GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 291–323, 2017.

[GKLP18] Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos 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.

[GKM$^+$18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 698–728, 2018.

[GKO$^+$20] Juan Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. Resource-restricted cryptography: Revisiting mpc bounds in the proof-of-work era. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 129–158, Cham, 2020. Springer International Publishing.

[GKP19] Juan A. Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Iterated search problems and blockchain security under falsifiable assumptions. Cryptology ePrint Archive, Report 2019/315, 2019. `https://eprint.iacr.org/2019/315`.

[GKP20] Juan A. Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Consensus from signatures of work. In *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, pages 319–344, 2020.

[GMPY11] Juan A Garay, Philip MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. *Journal of cryptology*, 24(4):615–658, 2011.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *Symposium on Theory of Computing, STOC 2011*, pages 99–108. ACM, 2011.

[JB99] Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999, San Diego, California, USA*. The Internet Society, 1999.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multime-*

dia Security, CMS ’99, pages 258–272, Deventer, The Netherlands, The Netherlands, 1999. Kluwer, B.V.

[JKS16] Ari Juels, Ahmed E. Kosba, and Elaine Shi. The ring of gyges: Investigating the future of criminal smart contracts. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 283–295, 2016.

[Kin13] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. http://primecoin.io/bin/primecoin-paper.pdf, July 2013.

[KKKZ18] Thomas Kerber, Markulf Kohlweiss, Aggelos Kiayias, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. *IACR Cryptology ePrint Archive*, 2018:1132, 2018.

[KMS14] Jonathan Katz, Andrew Miller, and Elaine Shi. Pseudonymous secure computation from time-lock puzzles. *IACR Cryptology ePrint Archive*, 2014:857, 2014.

[KMS$^+$16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 839–858, 2016.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO ’96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Technical report, IACR: Cryptology ePrint Archive, 2015.

[KP17] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. In *Progress in Cryptology - LATIN-*

CRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, 2017, Revised Selected Papers, 2017.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 631–648, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 357–388, 2017.

[KRs18] Lucianna Kiffer, Rajmohan Rajaraman, and abhi shelat. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 729–744, New York, NY, USA, 2018. ACM.

[KZZ16] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 705–734, 2016.

[LE10] David Leigh and Rob Evans. Wikileaks says funding has been blocked after government blacklisting. https://www.theguardian.com/media/2010/oct/14/wikileaks-says-funding-is-blocked, October 2010.

[Ler14] S. D. Lerner. Even faster block-chains with the decor protocol. Cryptology ePrint Archive, Report 2013/881, May 2014. `https://bitslog.wordpress.com/2014/05/02/decor/`.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security, 2015*, 2015.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015. `https://eprint.iacr.org/2015/366`.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[Mer89] Ralph C. Merkle. One way hash functions and DES. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 428–446, 1989.

[ML14] Andrew Miller and Joseph J. LaViola. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. University of Central Florida. Tech Report, CS-TR-14-01, April 2014.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[Nak09] Satoshi Nakamoto. Bitcoin open source implementation of p2p currency. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source, February 2009.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[Oku05a] Michael Okun. Agreement among unacquainted byzantine generals. In Pierre Fraigniaud, editor, *DISC*, volume 3724 of *Lecture Notes in Computer Science*, pages 499–500. Springer, 2005.

[Oku05b] Michael Okun. Distributed computing among unacquainted processors in the presence of byzantine distributed computing among unacquainted processors in the presence of byzantine failures. Ph.D. Thesis Hebrew University of Jerusalem, 2005.

[Omo14] Steve Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI matters*, 1(2):19–21, 2014.

[Per09] Colin Percival. Stronger key derivation via sequential memory-hard functions. *Self-published*, pages 1–16, 2009.

[PK15] Jack Peterson and Joseph Krug. Augur: a decentralized, open-source platform for prediction markets. *arXiv preprint arXiv:1501.01042*, 2015.

[Poe] A Poelstra. On stake and consensus (2015). *URL https://download. wpsoftware. net/bitcoin/pos. pdf.*

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pages 387–398.

[PS17] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 315–324, 2017.

[PS18] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 3–33, 2018.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017.

[PW92] Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 13-15, 1992, Proceedings*, pages 339–350, 1992.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990.

[SCO$^+$01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 566–598, 2001.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[SJAA$^+$14] Marcos A Simplicio Jr, Leonardo C Almeida, Ewerton R Andrade, Paulo CF dos Santos, and Paulo SLM Barreto. The lyra2 reference guide. Technical report, version 2.3. 2. Technical report, 2014.

[SKR$^+$11] Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Manuel González Nieto. Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 284–301, 2011.

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. SPECTRE: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 507–527, 2015.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE, 1982.