Bootstrapping the Blockchain — Directly

Juan A. Garay
Yahoo Research
garay@yahoo-inc.com

Aggelos Kiayias*
University of Edinburgh
akiayias@inf.ed.ac.uk

Nikos Leonardos
University of Athens
nikos.leonardos@gmail.com

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

October 18, 2016

Abstract

The Bitcoin backbone protocol [Eurocrypt 2015] extracts basic properties of Bitcoin's underlying *blockchain* data structure, such as "common prefix" and "chain quality," and shows how fundamental applications including consensus and a robust public transaction ledger can be built on top of them. The underlying assumptions are "proofs of work" (POWs), adversarial hashing power strictly less than 1/2 *and* no adversarial pre-computation—or, alternatively, the existence of an unpredictable "genesis" block.

In this paper we show how to remove the latter assumption, presenting a "bootstrapped" Bitcoin-like blockchain protocol relying on POWs that builds genesis blocks "from scratch" in the presence of adversarial pre-computation. The only known previous result in the same setting (unauthenticated parties, no trusted setup) [Crypto 2015] is indirect in the sense of creating a PKI first and then employing conventional PKI-based authenticated communication.

With our construction we establish that consensus can be solved directly by a blockchain protocol *without* trusted setup assuming an honest majority (in terms of computational power). We also formalize *miner unlinkability*, a privacy property for blockchain protocols, and demonstrate that our protocol retains the same level of miner unlinkability as Bitcoin itself.

1 Introduction

As the first decentralized cryptocurrency, Bitcoin [27] 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 [18] 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" [15, 30, 4, 20])[1], called *common prefix* and *chain quality*. It is then shown in [18] how applications such as consensus (aka Byzantine agreement) [29, 25] 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.

*Research supported by ERC project CODAMODA.

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

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 [6], arguably unpredictable.

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[2], 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 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.

This question was addressed by Andrychowicz and Dziembowski [1], who showed how to perform secure multiparty computation (MPC) [32, 19] without trusted setup by creating a PKI first using POWs. Given such PKI and assuming honest majority, MPC can then be performed using standard techniques and realize any cryptographic functionality. While this in principle addresses the foundational concerns, it leaves open the question of designing a blockchain protocol that is provably secure without trusted setup.

Our results. In this paper we answer the above question in the affirmative, by presenting a Bitcoin-like protocol that neither assumes a simultaneous start nor the existence of an unpredictable genesis block. 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 [18] and we assume a $1/2$ bound on adversarial hashing power. We call this protocol the *bootstrapped* (Bitcoin) backbone protocol.

The unique features of our blockchain protocol are as follows.

- *No trusted setup 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 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.
- *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 (check-

[2]The Ethereum Project, `https://www.ethereum.org/`.

ing 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 player running the protocol. The ranking process effectively yields a graded list of genesis blocks and is inspired by the "key ranking" protocol in [1], 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 players use different chain selection procedures since each predicate is "keyed" with the random coins that were contributed by each player in the challenge-exchange phase (and thus guaranteed to be fresh from the player'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, [18], 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 [18], 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 stabilise the genesis block and they might 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 challenges-exchange phase).

A high-level depiction of the protocol's phases, preceded by a period of potential precomputation by the corrupt players, is given in Figure 1.

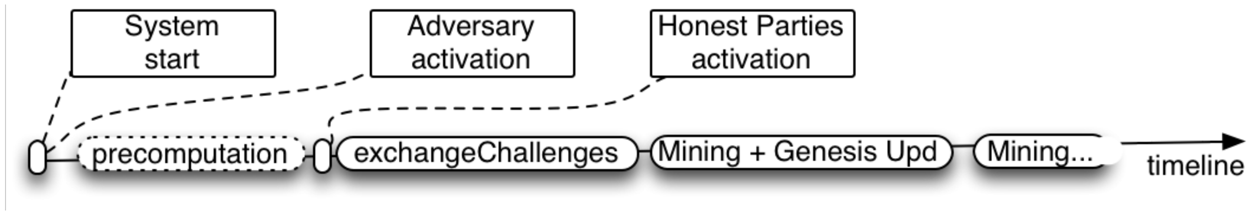

Figure 1: *Timeline and phases of the bootstrapped Bitcoin backbone protocol.*

We also formalize "miner unlinkability," which captures the property of a blockchain protocol to hide the identity of players (called "miners") assuming the network layer itself does not reveal this information. Unlinkability is formalized as follows: the adversary is incapable of distinguishing between a normal execution of the protocol and an execution of the protocol where all player messages are delivered via a "mix-net" [11]. It follows that if one wants to enjoy unlinkability in practice, a mix-net should be deployed among the players running the unlinkable blockchain protocol. This is feasible as it has been recently demonstrated by implementing a "peer-to-peer" anonymous broadcast; see [31] for a construction based on DC-nets [12].

We prove that our bootstrapped backbone protocol enjoys the same unlinkability level as the Bitcoin protocol; specifically, it is passively unlinkable and not actively unlinkable (see Section 2.3), as we construct an active attack against both protocols. It is worth stressing that passive unlinkability is another benefit of our direct design; protocols that build a PKI and subsequently use it to run a primitive such as authenticated broadcast cannot be unlinkable, even in the passive sense.

Finally, as mentioned above, besides the robust transaction ledger application, it is shown in [18] how a (randomized) consensus protocol can be built on top of the backbone protocol, assuming an honest majority (in terms of computational power) and trusted setup (an unpredictable genesis block). By applying the same construction on top of our bootstrapped backbone protocol, we immediately obtain a randomized consensus protocol for honest majority *without* a trusted setup, thus marking a contrast with the $\frac{2}{3}$ lower bound in the traditional network setting with no setup [7].

Related work. Nakamoto [26] 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[3], Primecoin [24], and Zerocash [5], and further analysis improvements (e.g., [17, 16]), to mention a few.

As mentioned above, we work in a model that generalizes the model of [18], who abstracted out and formalized the core of the Bitcoin protocol—the Bitcoin *backbone*. As presented in [18], 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 consider 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 [18] follow, since the probability that the adversary can use blocks mined before honest players "woke up" is negligible in the security parameter. In this paper we show how to establish such genesis block "from scratch," and directly, as the next comparison illustrates.

The problem of generating a genesis block ("unpredictable beacon") from scratch was also considered in [2], as part of the more general goal of investigating MPC without trusted setup assumptions. No trusted assumptions means no PKI (public-key infrastructure), a "graded" version of which is what the authors show how to build, based on POWs, in such a way that the number of identities each party gets is proportional to his hashing power.

In fact, the idea of using POWs as an identity-assignment tool was put forth earlier by Aspnes *et al.* [3], as a way to combat *Sybil* attacks [14], in such a way that the number of identities assigned to the honest and adversarial parties can be made proportional to their aggregate computational power, respectively. For example, by assuming that the adversary's computational power is less than 50%, one of the algorithms in [3] results in a number of adversarial identities less than half of that obtained by the honest parties. By running this procedure in a pre-processing stage, it is then suggested in [3] that a standard authenticated reliable broadcast protocol [13] could be run. Such protocols, however, would require that the PKI be *consistent*, details of which are not laid out in [3]. They are in [2], achieving the "graded" version mentioned above, with which parties can run MPC protocols in the setting of honest majority computing power, and reliable broadcast and unpredictable-beacon generation in the dishonest-majority (and random oracle) setting. As in [2], Katz *et al.* [22] also consider achieving pseudonymous broadcast and MPC from POWs ("cryptographic puzzles") and the existence of digital signatures without prior setup, but under the assumption of an unpredictable beacon.

In terms of additional properties for blockchain protocols and enhanced models, Kiayias and Panagiotakos [23] define "chain growth," together with chain quality and common prefix, and provide black-box arguments for reducing the properties of a robust transaction ledger (persistence and liveness—see Section 5) to these three properties. Here we follow a similar approach in the layout of our security arguments. Pass *et al.* [28] 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

[3]`http://www.litecoin.com`.

specified delay Δ, and show that the backbone protocol can be proven secure in this setting. It is an interesting direction to extend our results about the bootstrapped backbone protocol to their setting.

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we describe the network model, introduce some basic blockchain notation, and enumerate the varios security properties—the Bitcoin backbone's mentioned above as well as the new miner unlinkability property. In Section 3 we present the bootstrapped Bitcoin protocol, while in Section 4 its analysis. We conclude in Section 5 with a robust public transaction ledger as an application of the protocol, and the evaluation of its miner unlinkability.

2 Model and Definitions

We describe our protocols in a model that extends the synchronous communication network model presented in [18] 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 [8, 9]). As in [18], the protocol execution proceeds in rounds with inputs provided by an environment program denoted by $\mathcal{Z}$ to parties that execute the protocol.

Next we provide a high level overview of the model, focusing on the differences that are intrinsic to our setting where the adversary has a precomputation advantage. The adversarial model in the network is actively malicious following the standard cryptographic approach. The adversary is *rushing*, meaning that in any given round it gets to see all honest players's messages before deciding its strategy. 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. Importantly, the honest parties are not guaranteed to have the same view of the messages delivered in each round, except for the fact that all honest messages from the previous round are delivered. Furthermore, the adversary is allowed to change the source information on every message (i.e., communication is not authenticated). In the protocol description, we will use DIFFUSE as the message transmission command to capture the "send-to-all" functionality that is available in our setting.[4] Note that, as in [18], an adversarial sender may abuse DIFFUSE and attempt to confuse honest parties by sending and delivering inconsistent messages to them.

In contrast to [18], where all parties (the honest ones and the ones controlled by the adversary), are activated in the same round, in our model the environment will choose the round at which all the honest parties will become active; the corrupted parties, on the other hand, are activated in the first round. Once honest parties become active they will remain active until the end of the execution. In each round, after the honest parties become active, the environment activates each one by providing input to the party and receives the party's output when it terminates. When activated, parties are able to read their input tape INPUT() and communication tape RECEIVE(), perform some computation that will be suitably restricted (see below) and issue a DIFFUSE message that is guaranteed to be delivered to all parties at the beginning of the next round.

In more detail, we model the execution in the following manner. We employ the parameterized system of ITM's from [9] (2013 version) that is comprised of an initial ITM $\mathcal{Z}$, called the environment, and C, a control function that is specified below. We remark that our control function C is suitably restricted compared to that of [9, 10] to take into account restrictions in the order of execution that are relevant to our setting.

The execution is defined with respect to a protocol Π, a set of parties $P_1, \ldots, P_n$ and an adversary $\mathcal{A}$. The adversary is allowed to corrupt parties adaptively up to a number of $t < n$ parties. The

[4]In [18] the command name BROADCAST is used for this functionality, which we sometimes also will use informally.

protocol Π has access to two resources or "ideal functionalities," the random oracle, and the diffusion channel. Initially, the environment may pass input to either the adversary $\mathcal{A}$ or spawn an instance running the protocol Π which will be restricted to be assigned to the lexicographically smallest honest party (such restrictions are imposed by the control function [9]). After a party P_i is activated, the environment is restricted to activate the lexicographically next honest party, except in the case when no such party is left, in which case the next program to be activated is the adversary $\mathcal{A}$; subsequently, the round-robin execution order between the honest parties will be repeated.

Whenever a party is activated the control function allows for q queries to be made to the random oracle while in the case of an activation of $\mathcal{A}$ a number of $t \cdot q$ queries are allowed where t is the number of corrupted parties. Honest parties are also allowed to annotate their queries to the random oracle for verification purposes, in which case an unlimited amount of queries is permitted. Note that the adversary is not permitted to take advantage of this feature of the execution. With foresight, this asymmetry will be necessary, since otherwise it would be trivial for the adversary to break the properties of our protocols by simply "jamming" the incoming communication tape of the honest parties with messages whose verification would deplete their access quota to the random oracle per activation. Furthermore, for each party a single invocation to the diffusion channel is permitted. The diffusion channel maintains the list of messages diffused by each party, and permits the adversary $\mathcal{A}$ to perform a "fetch" operation so that it obtains the messages that were sent. When the adversary $\mathcal{A}$ is activated, the adversary will interact with the diffusion channel, preparing the messages to be delivered to the parties and performing a fetch operation. This write and fetch mode of operation with the communication channel enables the channel to enforce synchrony among the parties running the protocol (cf. [21]).

The term $\{\text{VIEW}^{P}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ denotes the random variable ensemble describing the view of party P after the completion of an execution with environment $\mathcal{Z}$, running protocol Π, and adversary $\mathcal{A}$, on auxiliary input $z \in \{0,1\}^*$. We often drop the parameters κ and z and simply refer to the ensemble by $\text{VIEW}^{P}_{\Pi,\mathcal{A},\mathcal{Z}}$ if the meaning is clear from the context. Following the resource-bounded computation model of [9], it holds that the total length of the execution is bounded by a polynomial in the security parameter κ and the length of the auxiliary string $|z|$, provided that the environment is *locally bounded* by a polynomial (cf. Proposition 3 in [9]). Note that the above execution model captures 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 round-robin execution of the honest parties commences.

We note that the above modeling obviates the need for a strict upper bound on the number of messages that may be transmitted by the adversary in each activation (as imposed by [2]). In our setting, honest parties, at the discretion of the environment, will be given sufficient time to process all the messages delivered via the diffusion channel including all messages that are injected by the adversary.

The concatenation of the view of all parties ever activated in the execution, say, $P_1, \ldots, P_n$, is denoted by $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$. As in [18], we are interested in protocols Π that do not make explicit use of the number of parties n or their identities. Further, note that because of the unauthenticated nature of the communication model the parties may never be certain about the number of participants in a protocol execution.

In our correctness and security statements we will be concerned with *properties* of protocols Π running in the above setting (as opposed to simulation-based notions of security). Such properties will be defined as predicates over the random variable $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, q, z)$ by quantifying over all locally polynomial-bounded adversaries $\mathcal{A}$ and environments $\mathcal{Z}$ (in the sense of [9]). Note that all our protocols will only satisfy properties with a small probability of error in κ as well as in a parameter k that is selected from $\{1, \ldots, \kappa\}$. (Note that, in practice, one may choose k to be much smaller than κ, e.g., $k = 6$.)

2.1 Blockchain notation

Next, we introduce some basic blockchain notation, following [18]. 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 $\mathsf{validblock}_q^D(B)$ 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 *blockchain*, or simply a *chain* is a sequence of *blocks*. The rightmost block is the *head* of the chain, denoted $\mathrm{head}(\mathcal{C})$. Note that the empty string ε is also a chain; by convention we set $\mathrm{head}(\varepsilon) = \varepsilon$. A chain $\mathcal{C}$ with $\mathrm{head}(\mathcal{C}) = \langle s', x', ctr' \rangle$ 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 $\mathrm{head}(\mathcal{C}_{\mathsf{new}}) = B$.

Consider a chain $\mathcal{C}$ of length m and any nonnegative integer k. We denote by $\mathcal{C}^{\lceil k}$ the chain resulting from the "pruning" the k rightmost blocks. Note that for $k \geq \mathrm{len}(\mathcal{C})$, $\mathcal{C}^{\lceil k} = \varepsilon$. If $\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$ we write $\mathcal{C}_1 \preceq \mathcal{C}_2$.

2.2 Security properties

We are going to show that the blockchain data structure built by our protocol satisfies a number of basic properties, as formulated in [18, 23]. At a high level, the first property, called *common prefix*, has to do with the existence, as well as persistence in time, of a common prefix of blocks among the chains of honest players [18]. Here we will consider a stronger variant of the property, presented in [23, 28], 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 5).

Definition 1 ((Strong) Common Prefix Property)**.** The *strong common prefix property* Q_{cp} with parameter $k \in \mathbb{N}$ states that the chains $\mathcal{C}_1, \mathcal{C}_2$ reported by two, not necessarily distinct honest parties P_1, P_2, at rounds r_1, r_2, with $r_1 \leq r_2$, satisfy $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.

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

Definition 2 (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 $\mathrm{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, q, z)$, 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 [18] an important lemma was established relating to the rate at which the chains of honest players were increasing as the Bitcoin backbone protocol was run. This was explicitly considered in [23] as a property under the name *chain growth*. Similarly to the variant of the common prefix property above, this property along with chain quality were shown sufficient for the black-box proof of application-level properties (in this case, transaction ledger *liveness*; see Section 5).

Definition 3 (Chain Growth Property)**.** The chain growth property Q_{cg} with parameters $\tau \in \mathcal{R}$ (the "chain speed" coefficient) and $s, r_0 \in \mathbb{N}$ states that for any round $r > r_0$, where honest party P has chain $\mathcal{C}_1$ at round r and chain $\mathcal{C}_2$ at round $r+s$ in $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, q, z)$, it holds that $|\mathcal{C}_2|-|\mathcal{C}_1| \geq \tau{\cdot}s$.

2.3 Miner unlinkability

Next, we formalize a blockchain protocol property that concerns the privacy of the parties ("miners"). At an intuitive level, *miner unlinkability* requires that it is hard for an adversary observing the network to distinguish between them, even at the level of linking the messages of a single miner across different rounds in the protocol execution.

Unlinkability is a relevant property for the Bitcoin backbone and other blockchain protocols that build "permissionless" blockchains since they do not require sender authentication in order to operate. Given this, it is easy to observe that the protocols can operate even if all message passing is performed via a "mix-net." A mix-net, introduced by Chaum in [11], is a network overlay that enables the delivery of a sequence of messages coming from a set of parties in some arbitrary random order so that it is not feasible to link the messages to their sources. We can incorporate an abstract mix-net in our execution model by augmenting the diffusion channel so that when it collects all messages transmitted by honest parties (or the empty message if no message is transmitted by an honest party), concatenate them and permute them according to some permutation over n elements. The diffusion channel will otherwise operate in the same fashion: after the adversary fetches the mixed sequence of messages, it will provide the list of messages for each party as before. We denote the view of the adversary in the execution of the protocol Π when the diffusion is passed via a mix-net (i.e., a random permutation in each round is applied each time the adversary performs a fetch operation.) by $\text{VIEW}^{\mathcal{A},\text{mix}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$. We are now ready to define the property.

Definition 4 (Miner Unlinkability)**.** A protocol Π satisfies *statistical* (resp., *computational*) *miner unlinkability* if the random variables $\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ and $\text{VIEW}^{\mathcal{A},\text{mix}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ are statistical (resp., computationally) indistinguishable for all z.

As a special case of the above, a protocol will be said to satisfy *passive miner unlinkability* if it satisfies the definition against an adversary $\mathcal{A}$ that acts "semi-honestly," i.e., it does not change the program of the corrupted parties. In contrast, in the general case of Definition 4, we will use the term miner unlinkability against an active attacker.

3 The Bootstrapped Backbone Protocol

In this section we present the "bootstrapped" Bitcoin backbone protocol; its security analysis is presented in the next section. In a nutshell, the protocol is a generalization of the protocol in [18], 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 players, 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 [1], where it is used to produce a "graded" PKI, as mentioned in Section 1.

The bootstrapped Bitcoin backbone protocol is executed by an arbitrary number of parties over an unauthenticated network (cf. Section 2). For concreteness, we assume that the number of parties running the protocol is n; however, parties need not be aware of this number when they execute the protocol. Communication over the network is achieved by utilizing a send-to-all DIFFUSE functionality that is available to all parties (and may be abused by the adversary in the sense of delivering different messages to different parties). After an initial ("challenge") phase, each party is to maintain a data structure called a "blockchain," as defined above. 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.

As in [18], the 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 $V(\cdot), I(\cdot), R(\cdot)$ which are specified by the application that runs "on top" of the backbone protocol.

Algorithm 1 The *bootstrapped backbone* protocol, parameterized by the *input contribution function* $I(\cdot)$, the *chain reading function* $R(\cdot)$, 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: $Rank \leftarrow \langle \epsilon \rangle$
6: $(\mathbf{c}, \mathbf{A}, c) \leftarrow \mathsf{exchangeChallenges}(1^\kappa)$
7: while TRUE do
8:     $k \leftarrow round - l - 2$
9:     $M_{Gen} \leftarrow \{(\langle s', x', ctr' \rangle, \langle A'_{l+1}, \ldots, A'_{l+1-k} \rangle)\}$ from RECEIVE()
10:    $M_{Chain} \leftarrow$ chains $\mathcal{C}'$ found in RECEIVE()
11:    $(Gen, Rank) \leftarrow \mathsf{updateValidate}(c, A, M_{Gen}, Gen, Rank)$
12:    $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, M_{Chain}, Gen, Rank)$
13:    $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \text{INPUT}(), \text{RECEIVE}())$
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 $\mathcal{C} = \epsilon$ then                                 ▷ New genesis block has been produced
17:            DIFFUSE( $(\mathcal{C}_{\mathsf{new}}, \langle A_{l+1}, \ldots, A_{l+1-(k+1)} \rangle)$ )
18:        end if
19:        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
20:        DIFFUSE($\mathcal{C}$)
21:    end if
22:    $round \leftarrow round + 1$
23:    if INPUT() contains READ then
24:        write $R(\mathbf{x}_{\mathcal{C}})$ to OUTPUT()
25:    end if
26: end while
```

The bootstrapped 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.

The challenge-exchange phase. In order to generate an unpredictable genesis block, players 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 player's k-round challenge, $1 \leq k \leq l$, depends on the $(k-1)$-round challenges of all honest players. This dependence is made explicit through a one-way hash function. 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.

```
function exchangeChallenges($1^\kappa$)
    $c_1 \overset{R}{\leftarrow} \{0,1\}^\kappa$
    DIFFUSE($c_0$)
    $round \leftarrow round + 1$
    while $round \leq l+1$ do
        $A_{round} \leftarrow \kappa$-bit messages found in RECEIVE()
        $r_{round} \overset{R}{\leftarrow} \{0,1\}^\kappa$
        $A_{round} \leftarrow A_{round} || r_{round}$
        $c_{round} \leftarrow H(A_{round})$                ▷ Compute challenge
        DIFFUSE($c_{round}$)
        $round \leftarrow round + 1$
    end while
    return $(\langle c_1, \dots c_l \rangle, \langle A_2, \dots A_{l+1} \rangle, c_{l+1})$
end function
```

Validation predicate update. In the original backbone protocol [18], 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 players 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, players 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 players to also identify the same genesis blocks, players broadcast the valid genesis blocks they have accepted together with the additional information needed by the other players for verification. The validation predicate update function is shown in Algorithm 3. Recall that *Gen* is the set of candidate genesis blocks.

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

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 $k \geq l$ then
4:         return $Gen, Rank$                                   ▷ No updates after round $2l + 2$
5:     end if
6:     for each $(\langle s', x', ctr' \rangle, \langle A'_{l+1}, \ldots, A'_{l+1-k} \rangle)$ in $M_{Gen}$ do
7:         if validblock$^D_q(\langle s, x, ctr \rangle) \wedge \langle s, x, ctr \rangle \notin Gen$ then
8:             $flag \leftarrow (H(A'_{l+1}) = s) \wedge (c_{l-k} \in A'_{l+1-k})$
9:             for $i = l + 1 - k$ to $l$ do
10:                if $H(A'_i) \notin A'_{i+1}$ then
11:                    $flag \leftarrow$ FALSE
12:                end if
13:            end for
14:            if $flag =$ TRUE then
15:                $Gen \leftarrow Gen \cup \langle s, x, ctr \rangle$
16:                $Rank[\langle s, x, ctr \rangle] \leftarrow l - k$
17:                DIFFUSE($\langle s, x, ctr \rangle, \langle A'_{l+1}, \ldots, A'_{l+1-k}, A_{l-k} \rangle$)   ▷ Augment $A'$ sequence with own $A$ value.
18:            end if
19:        end if
20:    end for
21:    return $Gen, Rank$
22: end function
```

Chain selection. The objective of the next algorithm in Algorithm 1, called maxvalid, is to find the "best possible" chain when given a set of chains. The accepted genesis blocks have different *weights* depending on when a player received them. In addition, it is possible that the same genesis block is received by honest players 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 players 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 reason for this will become apparent in the analysis—cf. Definition 5.) 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 $b =$ 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:                  $s' \leftarrow s$                                ▷ Retain hash value
10:                 $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$                    ▷ Remove the head from $\mathcal{C}$
11:             else
12:                 $b \leftarrow$ False
13:             end if
14:         until $(\mathcal{C} = \varepsilon) \vee (b =$ False$)$
15:     end if
16:     return $b$
17: end function
```

The proof-of-work function. Finally, we need to modify the proof-of-work function in [18], 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, in addition to the proof of genesis information sent in the backbone protocol, is required so that other honest players accept this block as valid and rank it accordingly. The code is presented in Algorithm 6.

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($\mathcal{C}_1, \ldots, \mathcal{C}_k$, $Gen$)
2:      $temp \leftarrow \varepsilon$
3:      $maxweight \leftarrow 0$
4:      for $i = 1$ to $k$ do
5:          if validate($\mathcal{C}_i, Gen$) then
6:              $w_{Gen_i} \leftarrow Rank(\text{tail}(\mathcal{C}_i))$
7:              $weight \leftarrow w_{Gen_i} + 3(\mathcal{C}_i| - 1)$
8:              if $maxweight < weight$ then
9:                  $maxweight \leftarrow weight$
10:                 $temp \leftarrow \mathcal{C}_i$
11:             end if
12:         end if
13:     end for
14:     return $temp$
15: end function
```

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

```
function pow($x, \mathcal{C}, c$)
    if $\mathcal{C} = \varepsilon$ then
        $s \leftarrow c$                                        ▷ c is required to prove freshness
    else
        $\langle s', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$
        $s \leftarrow H(ctr', G(s', x'))$
    end if
    $ctr \leftarrow 1$
    $B \leftarrow \varepsilon$
    $h \leftarrow G(s, x)$
    while $(ctr \leq q)$ do
        if $(H(ctr, h) < D)$ then                               ▷ Proof of work found
            $B \leftarrow \langle s, x, ctr \rangle$
            break
        end if
        $ctr \leftarrow ctr + 1$
    end while
    $\mathcal{C} \leftarrow \mathcal{C}B$                       ▷ Extend chain
    return $\mathcal{C}$
end function
```

Figure 2 presents the overall structure (phases and corresponding rounds) of the bootstrapped backbone protocol. Next, we turn to its analysis.

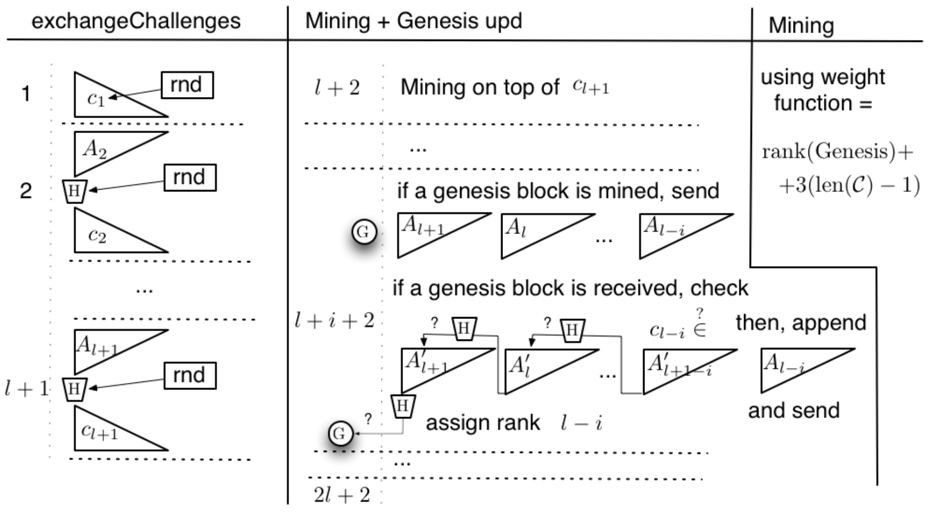

Figure 2: *The different phases and corresponding rounds of the bootstrapped backbone protocol.*

4 Analysis of the Bootstrapped Backbone Protocol

First, 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 5. Let $w_P(B)$ be the weight that P assigned to genesis block B. We define the weight of a chain $\mathcal{C}$ with genesis block B (with respect to party P)to be:

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

If block B was not received by P until round $2l+1$, or if $\mathcal{C} = \epsilon$, then $w_P(\mathcal{C}) = -1$.

In [18], 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 6. Regarding chains and their weight:
- Define $h_{\mathcal{C}} = \max_P\{w_P(\mathcal{C})\}$ and $\ell_{\mathcal{C}} = \min_P\{w_P(\mathcal{C})\}$.
- 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 $\ell_{\mathcal{C}}(B)$. As long as no collision happens $\ell(B)$ is well defined. The same holds for $h(B)$.)
- 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 [18], which we are also going to use in our analysis:

Definition 7. A round is called:
- *successful* if at least one honest party computes a solution in this round;
- *uniquely successful* if exactly one honest party computes a solution in this round.

Whether a block was mined by an honest party or by the adversary will be critical for our analysis.

Definition 8. In an execution blocks are called:
- *honest*, if mined by an honest party,
- *adversarial*, if mined by the adversary, and
- *u.s. blocks*, if mined in a uniquely successful round by an honest player.

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. The total hashing power of the honest players is $\alpha = pq(n-t)$, the hashing power of the adversary is $\beta = pqt$, and the total hashing power is $f = \alpha + \beta$. Moreover, in [18], a lower bound on the probability that a round is uniquely successful was established; denoted by γ and equal to $\alpha - \alpha^2$. Notice that γ is also a bound for the probability of a round being just successful.

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*, i.e., at least one honest party computed a POW at round j, and let $Y_j = 1$ iff j was a *uniquely successful round*, i.e., exactly one honest party computed a POW at round j. With respect to a set of rounds S, let $Z(S)$ denote the number of 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. Note that $\gamma|S| \leq \mathbb{E}[Y(S)] \leq \mathbb{E}[X(S)] \leq \alpha|S|$ and $\mathbb{E}[Z(S)] = \beta|S|$.

Lemma 9. *If $|S| = k$ and $\gamma \geq (1+\delta)\beta$ for some $\delta \in (0,1)$, then*

$$\Pr[Y(S) > (1 + \frac{5\delta}{9})Z(S)] > 1 - e^{-\Omega(\delta^2 k)}.$$

Proof. By the Chernoff bound we have that:

$$\Pr[Y(S) \leq (1 - \frac{\delta}{8})\mathbb{E}[Y(S)]] \leq e^{-\frac{\delta^2 \gamma k}{128}} \text{ and } Pr[Z(S) \geq (1 - \frac{\delta}{9})\mathbb{E}[Z(S)] \leq e^{-\frac{\delta^2 \beta k}{243}}.$$

Suppose none of the above events happens. Then, from the union bound, we get that with probability $1 - e^{-(2\min(\frac{\beta}{243}, \frac{\gamma}{128})\delta^2 k - \ln(2))}$ it holds that

$$Y(S) > (1 - \frac{\delta}{8})\gamma k \geq (1 - \frac{\delta}{8})(1+\delta)\beta k \geq (1 + \frac{5\delta}{9})(1 + \frac{\delta}{9})\beta k > (1 + \frac{5\delta}{9})Z(S).$$

□

Remark 1. For ease of exposition, in our analysis we will assume that there are no collisions; that is, for any two different queries to the random oracle, always a different response is returned. This would generally be a problem since for example it would break independence of X_i, X_j, for $i \neq j$, and we would not be able to apply the Chernoff bound in the previous lemma. However, since the probability of a collision happening, as well as all other events we consider, is at most $e^{-\Omega(\kappa)}$, we can always use the union bound to include the event of no collision occurring to our other assumptions. In addition, we assume that no two queries to the oracle are the same, as formalized by the Input Entropy condition in [18].

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

Lemma 10 (Graded Consistency). *If any honest party P_i accepts genesis block B with rank $w_{P_i}(B) > 1$, then all honest parties accept B with rank at least $w_{P_i}(B) - 1$.*

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 is going to broadcast $(B, E_{l+1}, .., E_{k+1}, A_k)$, 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 11 (Validity). *Genesis blocks computed by honest parties before round $2l+2$, 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 in the last round of the challenge-exchange phase. In addition, when the party broadcasts 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 12 (Freshness). *Let $r \leq l + 2$. Every block computed before round r cannot be part of some chain with genesis block B, where $w_P(B) \geq r - 1$ for some honest party P, with overwhelming probability in the security parameter κ.*

Proof. We first look into the case where the block in the statement is a genesis block. So 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 $A_{l+1}, \ldots, A_r$ 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-1} is the $r - 1$ round challenge of P, c_{r-1} is in A_r.

Due to the random nonce honest parties add to their challenges at every round, the probability that the adversary can guess c_{r-1} before round r is negligible in κ. In addition, since H is modeled as a random oracle, the probability that the adversary can create these sets, conditioned on the event that he cannot calculate c_{r-1} before round r, is also negligible. Hence, the adversary cannot create such a genesis block with overwhelming probability.

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, $w_p(B) \geq r - 1$. If no collision has occurred, 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. □

Weak chain growth. We now turn our attention to the weight of chains and prove a *weak* chain-growth property. In the original Backbone protocol [18], 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 13. *Let round r such that $l + 2 \leq r < 2l + 2$, and 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, where $r \leq s < 2l + 2$, every honest party P will have received a chain $\mathcal{C}'$ of weight at least $w_P(\mathcal{C}') = d - 2 + 3\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i$.*

Proof. Since $r < 2l + 2$, the genesis blocks of the chains that honest players have at this or any previous round, must have weight at least 1. Hence, by Lemma 10, for any chain $\mathcal{C}'$ of these chains, it should hold that $h_{C'} \leq \ell_{C'} + 1$. Let $\ell(r) = d$ iff d is the minimum value of the set $\{\ell_{\mathcal{C}} | P$ is honest and at round r has chain $\mathcal{C}\}$. Then we can show the following:

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

Suppose that by round s, 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. For the sake of contradiction, suppose that there exists some round $s > r$ such that $\ell(s) < d + 3$. 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. ⊣

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. 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' has received a chain $\mathcal{C}$ where $w_P(\mathcal{C}) \geq k$, then for every round $s' \geq r'$ it holds that $\ell(s') \geq k - 1$. Thus for every round $s' \geq r$ it holds that $\ell(s') \geq d - 2$.

We now have two cases. In the first case, $\sum_{i=r}^{s-1} Y_i \leq \sum_{i=r}^{s-1} Z_i$. 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:

$$\begin{aligned}\ell(s) &\geq d - 2 + 2\sum_{i=r}^{s-1} Y_i \geq d - 2 + 3\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Y_i \\ &\geq d - 2 + 3\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i,\end{aligned}$$

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

Otherwise, $\sum_{i=r}^{s-1} Y_i > \sum_{i=r}^{s-1} Z_i$. 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 $\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i$ 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,

$$\begin{aligned}\ell(s) &\geq d-2+3(\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i) + 2\sum_{i=r}^{s-1} Z_i \\ &\geq d-2+3\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i.\end{aligned}$$

□

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 wiil 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.

Lemma 14. *Suppose that for some $\delta \in (0,1)$, $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, and $\gamma \geq (1+\delta)\beta$, 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 with probability $1-e^{-\Omega(\delta^2 k)}$.*

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 the $\mathcal{C}$'s genesis block B. By Lemma 11 all honest parties know the genesis blocks mined by other honest parties, thus B must have been computed by the adversary. We have two cases. In the first case, round r is before round $2l+2$. Recall that $2l+1$ is the last round when the validation predicate is updated. Then, since P has adopted $\mathcal{C}$ in the previous round, he must have also broadcast B in the previous round. Thus, all honest parties will accept B as a valid genesis block at round r and will also accept $\mathcal{C}$ as valid, which is a contradiction.

Otherwise, suppose $r = 2l+2$. Again, if B was received before round $2l+1$ by some honest party, $\mathcal{C}$ will look valid to all parties. So B must have been received for the first time at round $2l+1$ by P; no honest party accepts new genesis blocks after this round. We will show that with overwhelming probability in k, no honest party will ever accept a chain based on B.

Without loss of generality, suppose that P is the first honest party that accepts a chain based on B at round r. Let E_1 be the event where the honest parties mine a genesis block after round $l+2+((1-\delta)k-1)$ for the first time. It holds that the probability of E_1 is at most $(1-\gamma)^{(1-\delta)k} < e^{-(1-\delta)k\gamma}$. So suppose that E_1 does not happen and at round $l+2+((1-\delta)k-1)$ $(< r)$ the honest parties have computed at least one genesis block that has weight at least $l-(1-\delta)k$. Hence, in this case, it follows from Lemma 13 that every honest party at round r will have a chain of weight at least $l-(1-\delta)k-2+3(Y(S')-1)-Z_1(S')$, where $S' = \{l+2,\ldots,r\}$ 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 12 the adversary can start working building $\mathcal{C}$ 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 $S = \{2,\ldots,r\}$ that are not in $Z_1(S')$. In order for $\mathcal{C}$ to be accepted by some honest party, it must hold that

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

Since $Z_1(S)$ and $Z_2(S')$ are disjoint and $S' \subseteq S$, the above implies:

$$3Z(S) \geq l - (1-\delta)k + 3Y(S') - 3. \tag{1}$$

Let E_2 be the event that

$$Y(S') \leq Z(S'). \tag{2}$$

From Lemma 9, if $|S'| > k$, which is the case here, $\Pr(E_2) \leq e^{-\Omega(\delta^2 k)}$. Also, let E_3 be the event that $Z(S \setminus S') \geq (1+\delta)l\beta$. Again, by an application of the Chernoff bound we have that $\Pr(E_3) \leq e^{-\Omega(\delta^2 k\beta)}$, since $|S \setminus S'| = l \geq k$. Suppose now that none of E_2 or E_3 holds. Then, since by our assumptions we have that

$$3(1+\delta)l\beta < l - (1-\delta)k - 3, \tag{3}$$

it follows that inequality 1 cannot hold and thus no honest party will ever accept a chain based on B. By an application of the union bound the event $E_1 \vee E_2 \vee E_3$ has probability at most $e^{-\Omega(\delta^2 k)}$ and the lemma follows.

A subtle point here is that since the lemma holds for $r = 2l+2$, it follows that Lemma 13 should hold for $r = 2l+2$. The same proof then can be applied for $r = 2l+3$ in this lemma, and inductively it follows that both lemmas hold for any round of the execution. However, since we have to apply repeatedly the universal validity proof for different r, we must argue about the probability that the statement holds for any r. Let $E_2(r)$ be the parameterized version of E_2, where $E_2(r)$ is the event where $Y(S') \leq Z(S')$ for $S' = \{l+2, \ldots, r\}$. Then, for some $\epsilon > 0$ it holds that

$$\bigvee_{i \geq 2l+2} E_2(i) \leq \sum_{i \geq 2l+2} e^{-(\epsilon\delta^2 k - \ln(2))} \leq e^{-(\epsilon\delta^2 k - \ln(2) + \ln(1 - e^{-\epsilon\delta^2}))} \leq e^{-\Omega(\delta^2 k)}$$

Thus, again by the union bound, the event $E_1 \vee \bigvee_{i \geq 2l+2} E_2(i) \vee E_3$ has probability at most $e^{-\Omega(\delta^2 k)}$. If this event does not occur, as we have argued universal validity holds for any r. Hence, the lemma follows. $\square$

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

Corollary 15. *Suppose that for some $\delta \in (0,1)$, $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, and $\gamma \geq (1+\delta)\beta$. Let round r such that $r \geq l+2$, and 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, where $r \leq s < 2l+2$, every honest party P will have received a chain $\mathcal{C}'$ of weight at least $w_P(\mathcal{C}') = d - 2 + 3\sum_{i=r}^{s-1} Y_i - \sum_{i=r}^{s-1} Z_i$ with probability at most $1 - e^{-\Omega(\delta^2 k)}$.*

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 16 (Precomputed blocks). *Assume* $3(1+\delta)f < 1$ *and* $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, *for some* $\delta \in (0,1)$. *Let* R *be the set that contains any adversarial block* B *mined before round* $l+2$, *where* $h(B) > l-1-(1-\delta)\delta^2 k$. *Then* $\Pr[|R| > \frac{5\delta}{9}k\beta] \leq e^{-\Omega(\delta^4 k)}$.

We are now ready to prove the security properties listed in Section 2.2.

4.1 Common Prefix

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

Lemma 17. *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* $\ell_{\mathcal{C}'} \geq \ell_{\mathcal{C}}(B) - 1$ *then there exists a unique adversarial block* B' *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.

For the sake of contradiction, suppose B is not a genesis block while B' is a genesis block and let B'' be the parent of B. Then $h_{\mathcal{C}}(B'') < \ell_{\mathcal{C}'}(B')$ since $h_{\mathcal{C}}(B'') \leq \ell_{\mathcal{C}}(B) - 2$. This implies than every honest party received B' before block B''. But then, no honest party would mine on the parent of B, because he would have lower weight than B', which leads to a contradiction. Hence, the lemma follows. □

Next, we use Lemma 17 in order to show that the existence of a fork implies that the adversary must have mined blocks proportional in number to the time the fork started. The proof is in the Appendix.

Theorem 18. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0,1)$. *Let* S *be the set of the chains of the honest parties from round* $2l+2$ *and onwards of the bootstrapped backbone protocol. Then the probability that* S *does not satisfy the strong common-prefix property with parameter* k *is at most* $e^{-\Omega(\delta^4 k)}$.

4.2 Chain Growth

We proved that after round $2l+1$ the strong common-prefix property is satisfied. This implies that all players 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.

Lemma 19. *Suppose 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* $\mathcal{C}'$ *of weight at least* $w_P(\mathcal{C}') = d - 1 + 3\sum_{i=r}^{s-1} X_i$.

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 according to their view 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 the 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 $w_P(\mathcal{C}') = d' = d - 1 + 3\sum_{i=r}^{s-2} X_i$ 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 19 implies the chain growth property after round $2l+1$.

Theorem 20. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0,1)$. *The bootstrapped Bitcoin protocol satisfies the chain growth property for* $r_0 = 2l+2$ *with speed coefficient* $(1-\delta)\gamma$ *and probability at least* $1 - e^{-\Omega(\delta^4 s)}$.

4.3 Chain Quality

We first observe a consequence of Theorem 18.

Lemma 21. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0,1)$. *From round* $2l+2$ *and onwards of the bootstrapped backbone protocol, the probability that the adversary has a chain which is more than* k *blocks longer than the chain of some honest party is at most* $e^{-\Omega(\delta^4 k)}$.

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 22. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0, 1/2)$. *Suppose* $\mathcal{C}$ *belongs to an honest party and consider any* k *consecutive blocks of* $\mathcal{C}$ *computed after round* $2l+2$ *of the bootstrapped backbone protocol. The probability that the adversary has contributed more than* $(1+\frac{\delta}{2})\frac{\beta}{\gamma} \cdot k \leq (1-\frac{\delta}{3})k$ *of these blocks is less than* $e^{-\Omega(\delta^5 k)}$.

Proof. 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 $S = \{r : r_1 \leq r \leq r_2\}$.

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

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

Let Z be the random variable that corresponds to the POWs obtained by the adversary during the rounds in S and X the successful rounds of the honest players in the same sequence of rounds.

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

$$Z \geq K - x \geq \Big(1+\frac{\delta}{2}\Big)\frac{\beta}{\gamma}K \geq \Big(1+\frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big(X - \frac{\gamma\delta k}{8f}\Big)$$

The first inequality comes from the fact that the adversary computed $K - x$ of the K blocks. The second one comes from the postulated relation between x and K. To see the last inequality, assume $X - \frac{\gamma\delta k}{8f} > K$. But then, by Lemma 21 for $k = \frac{\gamma\delta k}{8f}$ and Lemma 19, the assumption than an honest party is on B_w at r_2 is contradicted as all honest parties should be at chains of greater length.

To obtain the stated bound, note that if $|S| < (1-\delta)K/f$, then, since f is bounded away from 1 by a constant, the Chernoff bound implies that in $|S|$ rounds the total number of solutions is at least K with probability at most $e^{-\Omega(\delta^2 k)}$. Otherwise, $|S| \geq (1-\delta)K/f \geq (1-\delta)k/f$ and the bound follows from an application of the Chernoff bound, since $\mathbb{E}[Z] = \beta|S|$, while (using $E[X] \geq \gamma|S|$, $\mathbb{E}[X] \geq (1-\delta)\gamma k/f > \gamma k/2f$, and $(1+\frac{\delta}{2})(1-\frac{\delta}{4}) \geq (1+\frac{\delta}{8})$)

$$\mathbb{E}\Big[\Big(1+\frac{\delta}{2}\Big)\frac{\beta}{\gamma}\Big(X - \frac{\gamma\delta k}{8f}\Big)\Big] \geq \Big(1+\frac{\delta}{2}\Big)\frac{\beta}{\gamma}\cdot\mathbb{E}\Big[X - \frac{\delta}{4}\cdot\mathbb{E}[X]\Big] \geq \Big(1+\frac{\delta}{8}\Big)\beta|S|.$$

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. To manage this for a block he computed before the rounds in S implies he predicted the hash of a block in $\{B_j : u' \leq j \leq v'\}$; this occurs with probability negligible in $\log D$. If the block was computed after the rounds in S, then it was inserted between two existing blocks; this implies a collision. ☐

Corollary 23. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0, 1/2)$. *The bootstrapped Bitcoin protocol satisfies the chain-quality property with parameters* $\mu = (1+\frac{\delta}{2})\frac{\beta}{\gamma}$, $k_0 = 2f(1+\epsilon)(l+1)$, *and* k, *with probability at least* $1 - e^{\Omega(\delta^5 k)}$.

Proof. Note that the next two events occur with probability at least $1 - e^{\Omega(\epsilon^2 l)}$, for any $\epsilon \in (0,1)$. The honest parties in the first $l+1$ rounds have computed at most $\alpha(1+\epsilon)(l+1)$ blocks. The adversary, who might have been mining also during the challenges phase, has computed at most $2\beta(1+\epsilon)(l+1)$. The statement then follows from Theorem 22, since $\alpha(1+\epsilon)(l+1)+2\beta(1+\epsilon)(l+1) < 2f(1+\epsilon)(l+1)$. ☐

5 Robust Public Transaction Ledger

In [18] an instantiation of the functions $\mathrm{V}(\cdot), \mathrm{R}(\cdot), \mathrm{I}(\cdot)$ was given, in order for the Bitcoin backbone protocol to implement a robust public transaction 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. We denote the instantiation of our protocol with the same functions by $\Pi_{\mathsf{PL}}^{\mathsf{Boot}}$.

Definition 24. A protocol Π implements a *robust public transaction ledger* in the q-bounded synchronous setting without trusted setup if there is a round r_0 so that the following two properties are satisfied:

- *Persistence:* Parameterized by $k \in \mathbb{N}$ (the "depth" parameter), if in a certain round after r_0 an honest player reports a ledger that contains a transaction tx in a block more than k blocks away from the end of the ledger, then tx will always be reported in the same position in the ledger by any honest player from this round on.

– *Liveness:* Parameterized by $u, k \in \mathbb{N}$ (the "wait time" and "depth" parameters, resp.), provided that a transaction either (i) issued by Txgen, or (ii) is neutral, is given as input to all honest players continuously for u consecutive rounds after round r_0, then there exists an honest party who will report this transaction at a block more than k blocks from the end of the ledger.

Chain quality, chain growth and the strong common prefix property were shown in [23] to be sufficient to implement such a ledger[5] in a black-box manner. Our protocol satisfies all these properties after a specific condition is met; chain quality holds after the $2f(1+\epsilon)(l+1)$ block in the chain of any player, as Corollary 23 dictates, and common-prefix and chain growth hold after round $2l+2$ according to Theorem 18. Finally, 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.

Lemma 25 (Persistence). *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0, 1/2)$. *Then for all* $k \in \mathbb{N}$ *protocol* $\Pi_{\mathsf{PL}}^{\mathsf{Boot}}$ *satisfies Persistence after round* $2l+2$ *with probability* $1 - e^{-\Omega(\delta^5 k)}$, *where* k *is the depth parameter.*

Lemma 26 (Liveness). *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0, 1/2)$. *Further, assume oracle* Txgen *is unambiguous. Then for all* $k \in \mathbb{N}$ *protocol* $\Pi_{\mathsf{PL}}^{\mathsf{Boot}}$ *satisfies Liveness after round* $14(l+1)$ *with wait time* $u = \frac{3}{(1-\delta)\gamma} \cdot \max(k, \frac{1}{1-(1+\frac{\delta}{2})\frac{\beta}{\gamma}})$ *rounds and depth parameter* k *with probability at least* $1 - e^{-\Omega(\delta^5 k)}$.

Corollary 27. *Assume* $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$, *for some real* $\delta \in (0, 1/2)$. *Then, the protocol* $\Pi_{\mathsf{PL}}^{\mathsf{Boot}}$ *implements a robust transaction ledger with parameter* r_0 *equal to* $14(l+1)$.

Miner unlinkability. We now turn our attention to the miner unlinkability of the bootstrapped backbone protocol. We examine this property in the specific instantiation of the ledger application because unlinkability is impossible to show without taking into account the definition of the input selection function $I(\cdot)$; indeed, in the general case, it is always possible to define $I(\cdot)$ so that the miners leak their identity in the way they prepare the current block and thus enable the adversary to learn their identity. Following [18], the definition of $I(\cdot)$ can be performed so that a fresh bitcoin account is selected at each round as the recipient of the coinbase block reward. Using this we can prove the following proposition, which we prove both for the original bitcoin backbone as well as our bootstrapped version.

Proposition 28. Both the bootstrapped Bitcoin backbone protocol and the original Bitcoin backbone protocol *[18]* satisfy passive miner unlinkability for the choice of $I(\cdot)$ stated above. Neither protocol satisfies active miner unlinkability.

Proof. (sketch) Regarding passive unlinkability, it is easy to see that under the conditions suggested the protocol satisfies perfect miner unlinkability. Indeed, the way that miners produce fresh bitcoin accounts in $I(\cdot)$ is indistinguishable. On the other hand, it is easy to see that there is an attack in the active setting for both protocols. We describe it in the case of but the attack operates as follows: the adversary corrupts t players and performs the proof of work operation honestly until a block is computed. Subsequently, it delivers the new block to exactly half the honest parties in order to create two sets of honest players working on two different chains, say A and B (we assume without loss of generality that $n-t$ is even). The adversary will remain passive until the honest parties

[5]A similar definitional approach was pursued in [28].

compute a new block. When this happens, the adversary will associate a position in the message sequence with the corresponding set, A or B. Subsequently, once the honest parties converge in a single chain, the adversary will repeat the same division in the sets A, B for a total number of k times for the same sets A, B. It is easy to see that in the case of execution $\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$, the assignment of A and B will always be consistent, i.e., a position marked with A will never be marked as B and vice versa. On the other hand, in the case of execution $\text{VIEW}^{\mathcal{A},\text{mix}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$, a position marked with A initially, may be marked as B in the next iteration with probability 1/2. It follows that the event that the execution will be consistent is an event that happens with probability 2^{-k} and hence this provides an effective distinguisher between the two distributions. $\square$

Remark 2. It is worth pointing out that in case one bootstraps a blockchain protocol on top of the PKI forming stage following [2], the resulting protocol will not even be passively unlinkable. Indeed, posting a proof of work for a certain public-key and subsequently submitting a digital signature under this key, immediately breaks unlinkability, even passively, as it is only with negligible probability in the number of players that the randomly mixed execution will appear indistinguishable to a regular execution.

References

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

[2] M. Andrychowicz and S. Dziembowski. Pow-based distributed cryptography with no trusted setup. In R. Gennaro and M. 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.

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

[4] A. Back. Hashcash. http://www.cypherspace.org/hashcash, 1997.

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. *IACR Cryptology ePrint Archive*, 2014:349, 2014.

[6] Bitcoinwiki. Genesis block. https://en.bitcoin.it/wiki/Genesis_block.

[7] M. Borderding. Levels of authentication in distributed agreement. In Ö. Babaoglu and K. 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.

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

[9] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. *IACR Cryptology ePrint Archive*, 2000:67, 2000.

[10] R. 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.

[11] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[12] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[13] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[14] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

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

[16] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-ng: A scalable blockchain protocol. *CoRR*, abs/1510.02037, 2015.

[17] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

[18] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 281–310, 2015.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

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

[21] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. *IACR Cryptology ePrint Archive*, 2011:310, 2011.

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

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

[24] S. King. Primecoin: Cryptocurrency with prime number proof-of-work. http://primecoin.io/bin/primecoin-paper.pdf, July 2013.

[25] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

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

[27] S. Nakamoto. Bitcoin open source implementation of p2p currency. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source, February 2009.

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

[29] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

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

[31] T. Ruffing, P. Moreno-Sanchez, and A. Kate. P2p mixing and unlinkable bitcoin transactions. Cryptology ePrint Archive, Report 2016/824, 2016. `http://eprint.iacr.org/2016/824`.

[32] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.

A Proofs

A.1 Proof of Theorem 18.

We first prove a weaker lemma.

Lemma 29. *Let $3(1+\delta)f < 1$, $l > \frac{(1-\delta)k+3}{1-3(1+\delta)f}$, $\gamma \geq (1+\delta)\beta$ for some $\delta \in (0,1)$. Suppose that at round $r(\geq 2l+2)$ an honest party P_1 adopts chain $\mathcal{C}_1$ and and some honest party P_2 (possibly the same) has or adopts $\mathcal{C}_2$. The probability that $\mathcal{C}_1$ and $\mathcal{C}_2$ diverge at round $r' < r$ is at most $e^{-\Omega(\delta^4\beta(r-r'))}$.*

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 chains at some round r 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 $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}'_k$: $\ell_{\mathcal{C}} \geq \ell(B'_k) - 1$, 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 13, if $(B, \mathcal{C})$ and $(B', \mathcal{C}')$ are two consecutive elements of U, then

$$\ell_{\mathcal{C}'}(B') - \ell_{\mathcal{C}}(B) \geq 2 \tag{4}$$

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 covering most of the cases). In each case B' is determined by $\mathcal{C}'$ and Lemma 17.

- 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 $\bar{B} \notin \mathcal{C}$ and $\bar{B}' \notin \mathcal{C}$, then let $\mathcal{C}' = \{\mathcal{C}_1, \mathcal{C}_2\} \setminus \bar{\mathcal{C}}'$ and let B^* in chain $\mathcal{C}^* = \mathcal{C}$ be the block guaranteed from Lemma 17 for block $\bar{B}$ in chain $\mathcal{C}$. This block will be used in a subsequent step. Otherwise, $\bar{B} \in \mathcal{C}$ or $\bar{B}' \in \mathcal{C}$. 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 17 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 $\{\mathcal{C}_1, \mathcal{C}_2\} \setminus \bar{\mathcal{C}}'$. The adversarial block guaranteed by Lemma 17 for $\mathcal{C}'$ is no 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 $\{\mathcal{C}_1, \mathcal{C}_2\}$.

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 (4), this is true if they are not consecutive in U. To see this, observe that by Lemma 17,

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

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. These covers all different cases, except two. The first one, is when the first rule is applied and $\bar{B}'$ is in $\mathcal{C}$. Then, B' will be different than $\bar{B}'$ because, either is on a different chain ($\{\mathcal{C}_1, \mathcal{C}_2\} \setminus \mathcal{C}$) or they are on the same chain and $\ell_{\mathcal{C}}(B) - \ell_{\mathcal{C}}(\bar{B}') \geq 3$. The other case is when $\mathcal{C}^*$ has been defined by the first rule and the second rule is applied. In this case however, the honest block is matched to B^*, so we can safely ignore the matched block and still have a complete matching. Note that, B^* is unique since no block consecutive to the one matched on B^* is matched in chain $\mathcal{C}^*$. Hence, again it is impossible that these two blocks are the same.

$\dashv$

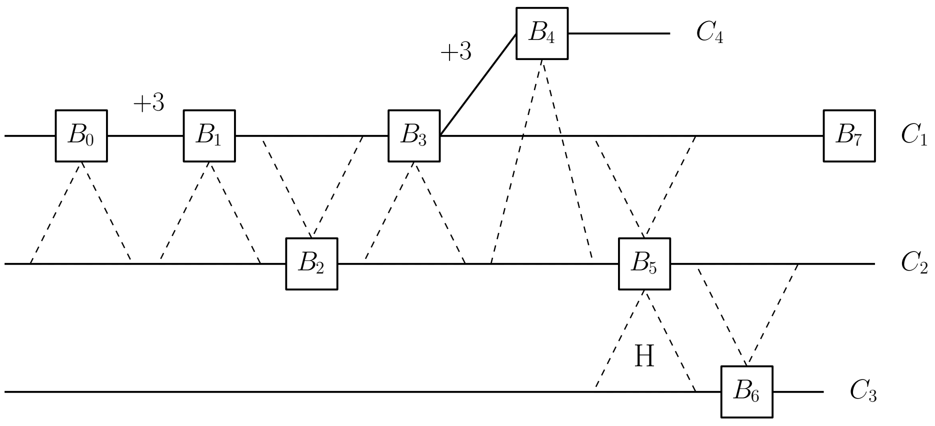

Figure 3: An example of the chains selected for the matching described in the claim in Lemma 29. $B_0, \ldots, B_7$ are u.s. blocks.

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 only with small probability that the adversary has mined this number of blocks.

We start by defining a "bad" event that will only happen with negligible probability. Afterwards, we will show that if this event does not occur, then there cannot be a fork as described on the statement of the Lemma. Let BAD be the event where at least one of the following events happens:

- a successful rounds happens for the first time after round $l + 1 + (1-\delta)\delta^2(r - r')$
- the event of Lemma 16 does happen
- for any $s \leq r - r'$, for the set of rounds $S = \{s, \dots, r\}$:

$$Y(S) \leq (1 + \frac{5\delta}{9})Z(S)$$

The probability that any of these events happens is at most $e^{-\Omega(\delta^4\beta(r-r'))}$. It follows from an application of the union bound to the disjunction of these events, that the probability that BAD happens is at most $e^{-\Omega(\delta^4\beta(r-r'))}$.

We will first study the case where C_2 has been adopted by P_2. Suppose $\mathcal{C}_1, \mathcal{C}_2$ diverge at round r' and assume BAD does not occur. Let block $B'_0 \in \mathcal{C}_1 \cap \mathcal{C}_2$ be the most recent u.s. block where all subsequent 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 4, 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, any u.s. block mined after B'_0 will be a descendant of B'_0 and $B'_0 \in C_1 \cap C_2$.

Now let B'_1 be the most recently mined u.s. block in some chain $\mathcal{C}'_1$ 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' in chain $\mathcal{C}'$ 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. Moreover, for the same reason, the last block of $\mathcal{C}_1 \cap \mathcal{C}'_1$ either is B'_0 or it is not u.s. block. Hence, if B' is the first u.s. block mined after B'_0 that is in some chain C', 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 $\ell_{\mathcal{C}_1} \geq \ell_{\mathcal{C}'}(B'_1) - 1$. Thus, we can apply Claim 2 for the chains $\mathcal{C}_1$ and $\mathcal{C}'_1$ from the first u.s. block mined after B'_0 up to block B'_1.

We apply this process as many times as possible. So B'_2 would be the most recently mined u.s. block in some chain $\mathcal{C}'_2$ 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 up to block 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. Notice that in the worst case B'_1 has been matched to a block B'' in $\mathcal{C}_1$. Also, B' will be a descendant of the (real) genesis block of $\mathcal{C}_1$. Hence, $l(B') - l(B'') = 3k$ 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 17 cannot be B''. This 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 in some chain $\mathcal{C}$, mined after B'_i, it holds that the last block of $\mathcal{C} \cap \mathcal{C}_1$ is not in $\mathcal{C}_1 \cap \mathcal{C}_2$. 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 we have two cases. If B'_0 actually exists, all adversarial blocks of the matching must have been mined 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 $S = \{r_0, \dots, r', \dots, r\}$ such that $Z(S) \geq Y(S)$. This implies that BAD does occur, which is a contradiction.

Otherwise, the adversary may use blocks that he has precomputed. Since BAD does not hold, the honest parties have computed at least one genesis block that weighs at least $l-(1-\delta)\delta^2(r-r')$. Then, all adversarial blocks in the matching must weigh at least $l-(1-\delta)\delta^2(r-r')-1$, and by Lemma 16 there are at most $5\delta/9$ such blocks. This implies that for $S=\{1,\dots,r\}$ it holds that $Y(S)\le(1+\frac{5\delta}{9})Z(S)$, which is a contradiction.

Thus, $\neg BAD$ implies that $\mathcal{C}_1$ and $\mathcal{C}_2$ diverge at some round greater or equal to r'. By contraposition we get that if $\mathcal{C}_1$ and $\mathcal{C}_2$ diverge at some round before r', BAD is implied. Thus, the probability of this event is at most the probability of BAD happening, and the lemma follows.

As a final note, if P_2 has chain $\mathcal{C}_2$ at the beginning of round r and $head(\mathcal{C}_1)$ is a u.s. block, it may not have a corresponding adversarial block in our matching. In this case, there exists players P_1', P_2' that adopted the chain ending in the parent of $head(\mathcal{C}_1)$ and $\mathcal{C}_2$ at round $r-1$, and these two chains diverge at round $r-r'$. Hence, again the proof holds with probability at most $e^{-\Omega(\delta^4\beta(r-r'-1))}=e^{-\Omega(\delta^4\beta(r-r'))}$. $\square$

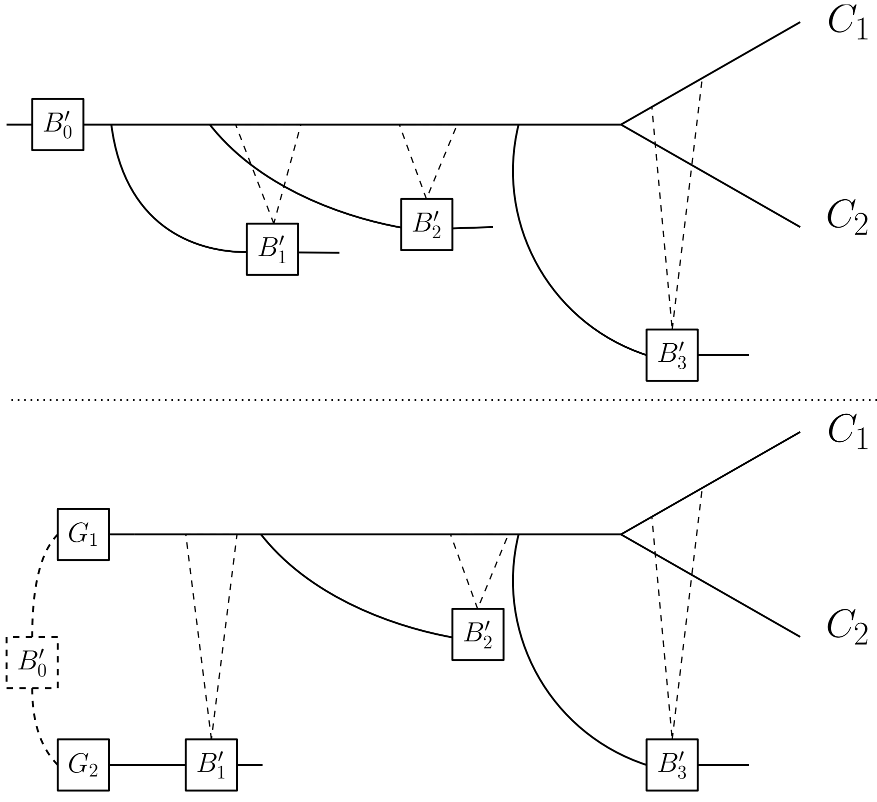

Figure 4: An example of two scenarios for the matching described by Lemma 29. Notice that in the second one an artificial block B_0' has been introduced.

The strong common-prefix property follows from the above lemma as in [23].

A.2 Proof of Lemma 16

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 $l+1-2(1-\delta)\delta^2k$, then B must have been computed after round $l+1-(1-\delta)\delta^2k$. Suppose that block B' was computed at round $l+1-r^*$

of the challenges phase, for $r^* > 2(1-\delta)\delta^2 k$; thus, by Lemma 12 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

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

blocks, starting from round $l + 1 - r^*$. Let E_1 be the event that the adversary after $r^* - (1-\delta)\delta^2 k$ rounds has computed more than $(1+\delta)(r^* - (1-\delta)\delta^2 k)\beta$ blocks. By an application of the Chernoff bound, $\Pr(E_1) \leq e^{-\Omega(\delta^4 k)}$. By our assumptions, this number of blocks is not sufficient to get a chain of the required weight, that is

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

Therefore, if E_1 does not hold, then the adversary will start mining B after round $l + 1 - (1-\delta)\delta^2 k$, for any B in R.

Next, we are going to bound the number of blocks the adversary can compute in $2(1-\delta)\delta^2 k$ rounds; recall that we are interested in blocks that were mined after round $l - 2(1-\delta)\delta^2 k$ and before round $l + 2$. Let E_2 be the event that the adversary mines at least $2(1+\delta/8)(1-\delta)\delta^2 k\beta (< \frac{5\delta}{9} k\beta)$ blocks in $2(1-\delta)\delta^2 k$ rounds. By an application of the Chernoff bound, $\Pr(E_2) \leq e^{-\Omega(\delta^4 k)}$. Since by an application of the union bound $E_1 \vee E_2$ happens with probability at most $e^{-\Omega(\delta^4 k)}$, the lemma follows. $\square$

A.3 Proof of Proposition 28

Proof. (sketch) Regarding passive unlinkability, it is easy to see that under the conditions suggested the protocol satisfies perfect miner unlinkability. Indeed, the way that miners produce fresh bitcoin accounts in $I(\cdot)$ is indistinguishable. On the other hand, it is easy to see that there is an attack in the active setting for both protocols. We describe it in the case of but the attack operates as follows: the adversary corrupts t players and performs the proof of work operation honestly until a block is computed. Subsequently, it delivers the new block to exactly half the honest parties in order to create two sets of honest players working on two different chains, say A and B (we assume without loss of generality that $n - t$ is even). The adversary will remain passive until the honest parties compute a new block. When this happens, the adversary will associate a position in the message sequence with the corresponding set, A or B. Subsequently, once the honest parties converge in a single chain, the adversary will repeat the same division in the sets A, B for a total number of k times for the same sets A, B. It is easy to see that in the case of execution $\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$, the assignment of A and B will always be consistent, i.e., a position marked with A will never be marked as B and vice versa. On the other hand, in the case of execution $\text{VIEW}^{\mathcal{A},\text{mix}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$, a position marked with A initially, may be marked as B in the next iteration with probability $1/2$. It follows that the event that the execution will be consistent is an event that happens with probability 2^{-k} and hence this provides an effective distinguisher between the two distributions. $\square$