The Bitcoin Backbone Protocol with Chains of Variable Difficulty

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

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

Nikos Leonardos†
National and Kapodistrian
University of Athens
nikos.leonardos@gmail.com

November 7, 2016

Abstract

Bitcoin's innovative and distributedly maintained *blockchain* data structure hinges on the adequate degree of difficulty of so-called "proofs of work," which miners have to produce in order for transactions to be inserted. Importantly, these proofs of work have to be hard enough so that miners have an opportunity to unify their views in the presence of an adversary who interferes but has bounded computational power, but easy enough to be solvable regularly and enable the miners to make progress. As such, as the miners' population evolves over time, so should the difficulty of these proofs. Bitcoin provides this adjustment mechanism, with empirical evidence of a constant block generation rate against such population changes.

In this paper we provide the first (to our knowledge) formal analysis of Bitcoin's target (re)calculation function in the cryptographic setting, i.e., against all possible adversaries aiming to subvert the protocol's properties. We extend the q-bounded synchronous model of the *Bitcoin backbone protocol* [Eurocrypt 2015], which posed the basic properties of Bitcoin's underlying blockchain data structure and shows how a robust public transaction ledger can be built on top of them, to environments that may introduce or suspend parties in each round.

We provide a set of necessary conditions with respect to the way the population evolves under which the "Bitcoin backbone with chains of variable difficulty" provides a robust transaction ledger in the presence of an actively malicious adversary controlling a fraction of the miners strictly below 50% at each instant of the execution. Our work introduces new analysis techniques and tools to the area of blockchain systems that may prove useful in analyzing other blockchain protocols.

*Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

†Research partly supported by ERC project CODAMODA, No. 259152, and Horizon 2020 project PANORAMIX, No. 653497.

1 Introduction

The Bitcoin backbone protocol [GKL15] extracts and analyzes the basic properties of Bitcoin's underlying *blockchain* data structure, such as "common prefix" and "chain quality," which parties ("miners") maintain and try to extend by generating "proofs of work" (POW, aka "cryptographic puzzle" [DN92, RSW96, Bac97, JB99])[1]. It is then formally shown in [GKL15] how fundamental applications including consensus [PSL80, LSP82] and a robust public transaction ledger realizing a decentralized cryptocurrency (i.e., Bitcoin [Nak09]) can be built on top of them, assuming that the hashing power of an adversary controlling a fraction of the parties is strictly less than 1/2.

The results in [GKL15], however, hold for the *static* setting, where the protocol is executed by a *fixed* number of parties (albeit not necessarily known to the participants), and therefore with POWs (and hence blockchains) of fixed difficulty. This is in contrast to the actual deployment of the Bitcoin protocol where a "target (re)calculation" mechanism adjusts the hardness level of POWs as the number of parties varies during the protocol execution. In more detail, in [GKL15] the target T that the hash function output must not exceed, is set and hardcoded at the beginning of the protocol, and in such a way that a specific relation to the number of parties running the protocol is satisfied, namely, that the ratio $f = qnT/2^\kappa$ is small, where q is the number of queries to the hash function that each party is allowed per round, n is the number of parties, and κ is the length of the hash function output. Security was only proven when the number of parties is n and the choice of target T is never recalculated, thus leaving as open question the full analysis of the protocol in a setting where, as in the real world, parties change dynamically over time.

In this paper, we abstract the target recalculation algorithm from the Bitcoin system, and present a generalization and analysis of the Bitcoin backbone protocol with chains of variable difficulty, as produced by an evolving population of parties, thus answering the aforementioned open question.

In this setting, there is a parameter m which determines the length of an "epoch" in number of blocks.[2] When a party prepares to compute the j-th block of a chain with $j \bmod m = 1$, it uses a target calculation algorithm that determines the proper target value to use, based on the party's local view about the total number of parties that are present in the system, as reflected by the rate of blocks that have been created so far and are part of the party's chain. (Each block contains a timestamp of when it was created; in our synchronous setting, timestamps will correspond to the round numbers when blocks are created—see Section 2.) To accomodate the evolving population of parties, we extend the model of [GKL15] to environments that are free to introduce and suspend parties in each round. In other respects, we follow the model of [GKL15], where all parties have the same "hashing power," with each one allowed to pose q queries to the hash function that is modeled as a "random oracle" [BR93]. We refer to our setting as the *dynamic q-bounded synchronous setting*.

In order to give an idea of the issues involved, we note that without a target calculation mechanism, in the dynamic setting the backbone protocol is not secure *even if all parties are honest* and follow the protocol faithfully. Indeed, it is easy to see that a combination of an environment that increases the number of parties and adversarial network conditions can lead to substantial divergence (a.k.a. "forks") in the chains of the honest parties, leading to the violation of the agreement-type properties that are needed for the applications of the protocol, such as maintaining a robust transaction ledger. The attack is simple: the environment increases the number of parties constantly so that the block production rate per round increases (which is roughly the parameter f mentioned above); then, adversarial network conditions may divide the parties into two sets A, B and schedule message delivery so that parties in set A receive blocks produced by parties in A first, and similarly

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

[2] In Bitcoin, m is set to 2016 and roughly corresponds to 2 weeks in real time—assuming the number of parties does not change much.

for set B. According to the Bitcoin protocol, parties adopt the block they see first, and thus the two sets will maintain two separate blockchains. While this specific attack could in principle be thwarted by modifying the Bitcoin backbone (e.g., by randomizing which block a party adopts when they receive in the same round two blocks of the same index in the chain), it certainly would not cope with all possible attacks in the presence of a full-blown adversary and target recalculation mechanism. Indeed, such an attack was shown in [Bah13], where by mining "privately" with timestamps in rapid succession, corrupt miners are able to induce artificially high targets in their private chain; even though such chain may grow slower than the main chain, it will still make progress and, via an anti-concentration argument, a sudden adversarial advance that can break agreement amongst honest parties cannot be precluded.

Given the above, our main goal is to show that the backbone protocol with a Bitcoin-like target recalculation function satisfies the common prefix and chain quality properties, as an intermediate step towards proving that the protocol implements a robust transaction ledger. Expectedly, the class of protocols we will analyze will not preserve its properties for arbitrary ways in which the number of parties may change over time. In order to bound the error in the calibration of the block generation rate that the target recalculation function attempts, we will need some bounds on the way the number of parties may vary. For $\gamma \in \mathbb{R}^+, s \in \mathbb{N}$, we will call a sequence of parties $(n_r)_{r\in\mathbb{N}}$ (γ, s)-*respecting* if it holds that in a sequence of rounds S with $|S| \leq s$, $\max_{r\in S} n_r \leq \gamma \cdot \min_{r\in S} n_r$, and will determine for what values of these parameters the backbone protocol is secure.

After formally describing blockchains of variable difficulty and the Bitcoin backbone protocol in this setting, at a high level our analysis goes as follows. We first introduce the notion of *goodness* regarding the approximation that is performed on f in an epoch. In more detail, we call a round r (η, θ)-*good* for some parameters $\eta, \theta \in \mathbb{R}^+$, if the value f_r computed for the actual number of parties and target used in round r by some honest party, falls in the range $[\eta f, \theta f]$, where f is the initial block production rate (note that the first round is always assumed good). Together with "goodness" we introduce a notion of *typical* executions, in which, informally, for any set S of consecutive rounds the successes of the adversary and the honest parties do not deviate too much from their expectations as well as no "bad" event concerning the hash function occurs (such as a collision). Using a martingale bound we demonstrate that almost all polynomially bounded (in κ) executions are typical. We then proceed to show that in a typical execution any chain that an honest party adopts (1) contains timestamps that are approximately accurate (i.e., no adversarial block has a timestamp that differs too much from its real creation time) and (2) has a target such that the probability of block production remains near the fixed constant f, i.e., it is "good." Finally, these properties of a typical execution allow us to demonstrate that a typical execution enjoys the common prefix and chain quality properties which is an intermediate stepping stone towards the ultimate goal, that of establishing that the backbone protocol with variable difficulty implements a robust transaction ledger. Specifically, we show the following:

Main result (cf. Theorems 26 and 27). The Bitcoin backbone protocol with chains of variable difficulty, suitably parameterized, satisfies with overwhelming probability in m, κ the properties of (1) persistence—if a transaction tx is confirmed by an honest party, no honest party will ever disagree about the position of tx in the ledger, and (2) liveness—if a transaction tx is broadcast, it will eventually become confirmed by all honest parties.

Remark. The most important lesson from our analysis is that the length of an epoch, m, is a security parameter that should be selected to be high enough in order to bound the probability of an attack by an arbitrary cryptographic adversary. Regarding the actual parameterization of the Bitcoin system (that uses epochs of $m = 2016$ blocks), even though it is consistent with all the constraints of our theorems (cf. Remark 3 in Section 6.1), our martingale analysis is not tight

enough to provide a tangible security guarantee. In fact, our probabilistic analysis would require extremely long epochs to provide a sufficiently small probability of attack; as such, it is primarily of theoretical interest. Nevertheless, we are confident that the analysis and tools that we introduce will motivate further research on how the parameters of blockchain systems may be chosen in practice w.r.t. target recalculation.

Finally, we note that various extensions to our model are relevant to the Bitcoin system and constitute interesting directions for further research. Importantly, a security analysis in the "rational" setting (see, e.g., [ES14, SSZ15, KKKT16]), and in the "semi-synchronous" network model [PSS16].

2 Model and Definitions

We describe our protocols in a model that extends the synchronous communication network model presented in [GKL14, GKL15] for the analysis of the Bitcoin backbone protocol in the static setting with a fixed number of parties (which in turn is based on Canetti's formulation of "real world" notion of protocol execution [Can00a, Can00b, Can01] for multi-party protocols) to the dynamic setting with a varying number of parties. In this section we provide a high-level overview of the model, highlighting the differences that are intrinsic to our dynamic setting; a more detailed specification can be found in Appendix A.

As in [GKL14], the protocol execution proceeds in rounds with inputs provided by an environment program denoted by $\mathcal{Z}$ to parties that execute the protocol Π; the underlying communication graph is not fully connected and messages are delivered through a "diffusion" mechanism that reflects Bitcoin's peer-to-peer structure; and our adversarial model in the network is *adaptive*, meaning that the adversary ($\mathcal{A}$) is allowed to take control of parties on the fly, and *rushing*, meaning that in any given round the adversary gets to see all honest parties' messages before deciding his strategy.

The parties that *may* become active in a protocol execution are encoded as part of a control program C and come from a universe $\mathcal{U}$ of parties. The protocol execution is driven by an environment program $\mathcal{Z}$ that interacts with other instances of programs that it spawns at the discretion of the control program C. The pair $(\mathcal{Z}, C)$ forms of a *system of interactive Turing machines* (ITM's) in the sense of [Can00b]. The execution is with respect to a program Π, an adversary $\mathcal{A}$ (which is another ITM) and the universe of parties $\mathcal{U}$. Additionally, C maintains a flag for each instance of an ITM (abbreviated ITI in the terminology of [Can00b]), that is called the `ready` flag and is initially set to false for all parties. Observe that parties are unaware of the set of activated parties.

The parties' access to the hash function and their communication mechanism is captured by a joint random oracle/diffusion functionality. The "diffusion" aspect of the functionality, see [GKL14], allows the order of messages to be controlled by $\mathcal{A}$, i.e., there is no atomicity guarantees in message broadcast [HT94], and, furthermore, the adversary is allowed to spoof the source information on every message (i.e., communication is not authenticated). Still, the adversary cannot change the contents of the messages nor prevent them from being delivered. We will use DIFFUSE as the message transmission command that captures this "send-to-all" functionality.

Regarding the random oracle aspect of the functionality, the Bitcoin backbone protocol requires from parties (miners) to solve a "proof of work" (POW, aka "cryptographic puzzle" [DN92]) in order to generate new blocks for the blockchain[3]. This is modeled in [GKL15] as parties having access to the oracle $H(\cdot)$. In more detail, given a query with a value x marked for "calculation" for the function $H(\cdot)$ from a honest party P_i and assuming x has not been queried before, the functionality returns a value y which is selected at random from $\{0,1\}^\kappa$, where κ is the security parameter; furthermore, it stores the pair (x, y) in the table of $H(\cdot)$, in case the same value x is queried in the

[3] As mentioned earlier, in Bitcoin a POW essentially amounts to brute-forcing a hash inequality based on SHA-256.

future. Each honest party P_i is allowed to ask q queries in each round. Rounds are determined by the diffusion functionality which keeps track of all activations. On the other hand, each honest party is given unlimited queries for "verification" for the function $H(\cdot)$. The adversary $\mathcal{A}$, on the other hand, is given at each round r a number of queries that cannot exceed $t_r \cdot q$ the upper bound on the number of corrupted parties that may be activated in round r. No verification queries are provided to $\mathcal{A}$. The value q is a polynomial function of κ.

We will refer to the setting defined by the two series $\mathbf{n} = \{n_r\}_{r\in\mathbb{N}}$ and $\mathbf{t} = \{t_r\}_{r\in\mathbb{N}}$, representing the number of "`ready`" honest parties and the bound on corrupted parties that may be activated at each round r, with the above bounded access to the random oracle functionality as allowed by the control program C, as the *dynamic q-bounded synchronous setting*.

We will use $\{\text{VIEW}^{P,\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ to denote the random variable ensemble describing the view of party P after the completion of an execution running protocol Π with environment $\mathcal{Z}$ and adversary $\mathcal{A}$, on input $z \in \{0,1\}^*$. Since we will only consider a "standalone" execution without any auxiliary information, we will restrict ourselves to executions with $z = 1^\kappa$, and thus we will simply refer to the ensemble by $\text{VIEW}^{P,\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$. The concatenation of the view of all parties ever activated in the execution will be denoted by $\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$. In our theorems we will be concerned with *properties* of protocols Π running in the above setting. Such properties will be defined as predicates over the random variable $\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$ by quantifying over all possible adversaries $\mathcal{A}$ and environments $\mathcal{Z}$. 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., $k = 6$).

The protocol class that we will analyze will not be able to preserve its properties for arbitrary sequences of parties. To restrict the way the sequence $\mathbf{n}$ is fluctuating we will introduce the following class of sequences.

Definition 1. For $\gamma \in \mathbb{R}^+$, we call a sequence $(n_r)_{r\in\mathbb{N}}$ (γ, s)-*respecting* if it holds that in a sequence of rounds S with $|S| \leq s$ rounds, $\max_{r\in S} n_r \leq \gamma \cdot \min_{r\in S} n_r$.

Observe that the above sequence is fairly general and can also capture exponential growth; for example, by setting $\gamma = 2$ and $s = 10$, it follows that every 10 rounds the number of ready parties may double.[4] More formally, a protocol Π would satisfy a property Q for a certain class of sequences $\mathbf{n}, \mathbf{t}$, provided that for all PPT $\mathcal{A}$ and locally polynomial bounded $\mathcal{Z}$, it holds that $Q(\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}})$ is true with overwhelming probability on the coins of $\mathcal{A}, \mathcal{Z}$ and the random oracle functionality.

In this paper we will be interested in (γ, s)-respecting sequences $\mathbf{n}$, sequences $\mathbf{t}$ suitably restricted by $\mathbf{n}$, environments restricted according to (γ, s)-respecting sequences (we will call such environments also (γ, s)-respecting) and protocols Π that are suitably parameterized given $\mathbf{n}, \mathbf{t}$. Refer to Appendix A for further details on the model of protocol execution. For convenience, Table 1 in Section 6.1 summarizes all the parameters relevant to the analysis.

3 Blockhains of Variable Difficulty

We start by introducing blockchain notation; we use similar notation to [GKL14], and expand the notion of blockchain to explicitly include *timestamps* (in the form of a round indicator). Let $G(\cdot)$ and $H(\cdot)$ be cryptographic hash functions with output in $\{0,1\}^\kappa$. A *block with target* $T \in \mathbb{N}$ is a quadruple of the form $B = \langle r, st, x, ctr\rangle$ where $st \in \{0,1\}^\kappa, x \in \{0,1\}^*$, and $r, ctr \in \mathbb{N}$ are such that

[4] Note that this will not lead to an exponential running time overall since the total run time is bounded by a polynomial in κ.

they satisfy the predicate $\mathsf{validblock}_q^T(B)$ defined as

$$(H(ctr, G(r, st, x)) < T) \wedge (ctr \leq q).$$

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 (cf. Section 2). 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 $\mathrm{head}(\mathcal{C}) = \langle r, st, x, ctr \rangle$ can be extended to a longer chain by appending a valid block $B = \langle r', st', x', ctr' \rangle$ that satisfies $st' = H(ctr, G(r, st, x))$ and $r' > r$, where r' is called the *timestamp* of block B. In case $\mathcal{C} = \varepsilon$, by convention any valid block of the form $\langle r', st', x', ctr' \rangle$ may extend it. In either case we have an extended chain $\mathcal{C}_{\mathsf{new}} = \mathcal{C}B$ that satisfies $\mathrm{head}(\mathcal{C}_{\mathsf{new}}) = B$.

The *length* of a chain $\mathrm{len}(\mathcal{C})$ is its number of blocks. Consider a chain $\mathcal{C}$ of length ℓ and any nonnegative integer k. We denote by $\mathcal{C}^{\lceil k}$ the chain resulting from "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 $\mathcal{C}_1 \preceq \mathcal{C}_2$.

Given a chain $\mathcal{C}$ of length $\mathrm{len}(\mathcal{C}) = \ell$, we let $\mathbf{x}_{\mathcal{C}}$ denote the vector of ℓ values that is stored in $\mathcal{C}$ and starts with the value of the first block. Similarly, $\mathbf{r}_{\mathcal{C}}$ is the vector that contains the timestamps of the blockchain $\mathcal{C}$.

For a chain of variable difficulty, the target T is recalculated for each block based on the round timestamps of the previous blocks. Specifically, there is a function $D : \mathbb{Z}^* \to \mathbb{R}$ which receives an arbitrary vector of round timestamps and produces the next target. The value $D(\varepsilon)$ is the initial target of the system. The *difficulty* of each block is measured in terms of how many times the block is harder to obtain than a block of target T_0. Specifically, the difficulty of a block with target T will be equal to T_0/T. We will use $\mathrm{diff}(\mathcal{C})$ to denote the difficulty of a chain. This is equal to the sum of the difficulties of all the blocks that comprise the chain.

The target calculation function. Intuitively, the target calculation function $D(\cdot)$ aims at maintaining the block production rate constant. It is parameterized by $m \in \mathbb{N}$ and $f \in (0, 1)$; Its goal is that m blocks will be produced every m/f rounds. We will see in Section 6 that the probability $f(T, n)$ with which n parties produce a new block with target T is approximated by

$$f(T, n) \approx \frac{qTn}{2^\kappa}.$$

(Note that $T/2^\kappa$ is the probability that a single player produces a block in a single query.)

To achieve the above goal Bitcoin tries to keep $qTn/2^\kappa$ close to f. To that end, Bitcoin waits for m blocks to be produced and based on their difficulty and how fast these blocks were computed it computes the next target. More specifically, say the last m blocks of a chain $\mathcal{C}$ are for target T and were produced in Δ rounds. Consider the case where a number of players

$$n(T, \Delta) = \frac{2^\kappa m}{qT\Delta}$$

attempts to produce m blocks of target T; note that it will take them approximately Δ rounds in expectation. Intuitively, the number of players at the point when m blocks were produced is estimated by $n(T, \Delta)$; then the next target T' is set so that $n(T, \Delta)$ players would need m/f rounds in expectation to produce m blocks of target T'. Therefore, it makes sense to set

$$T' = \frac{\Delta}{m/f} \cdot T,$$

because if the number of players is indeed $n(T, \Delta)$ and remains unchanged, it will take them m/f rounds in expectation to produce m blocks. If the initial estimate of the number parties is n_0, we will assume T_0 is appropriately set so that $f \approx qT_0n_0/2^\kappa$ and then

$$T' = \frac{n_0}{n(T, \Delta)} \cdot T_0.$$

Remark 1. Recall that in the flat q-bounded setting all parties have the same hashing power (q-queries per round). It follows that n_0 represents the estimated initial hashing power while $n(T, \Delta)$ the estimated hashing power during the last m blocks of the chain $\mathcal{C}$. As a result the new target is equal to the initial target T_0 multiplied by the factor $n_0/n(T, \Delta)$, reflecting the change of hashing power in the last m blocks.

Based on the above we can give the formal definition of the target (re)calculation function as follows.

Definition 2. For fixed constants $\kappa, \tau, m, n_0, T_0$, the target calculation function $D : \mathbb{Z}^* \to \mathbb{R}$ is defined as

$$D(\varepsilon) = T_0 \quad \text{and} \quad D(r_1, \ldots, r_v) = \begin{cases} \frac{1}{\tau} \cdot T & \text{if } \frac{n_0}{n(T,\Delta)} \cdot T_0 < \frac{1}{\tau} \cdot T; \\ \tau \cdot T & \text{if } \frac{n_0}{n(T,\Delta)} \cdot T_0 > \tau \cdot T; \\ \frac{n_0}{n(T,\Delta)} \cdot T_0 & \text{otherwise,} \end{cases}$$

where $n(T, \Delta) = 2^\kappa m/qT\Delta$, with $\Delta = r_{m'} - r_{m'-m}$, $T = D(r_1, \ldots, r_{m'-1})$, and $m' = m \cdot \lfloor v/m \rfloor$.

In the definition, $(r_1, \ldots, r_v)$ corresponds to a chain of v blocks with r_i the timestamp of the ith block; m', Δ, and T correspond to the last block, duration, and target of the last completed epoch, respectively.

Remark 2. A remark is in order about the case $\frac{n_0}{n(T,\Delta)} \cdot T_0 \notin [\frac{1}{\tau}T, \tau T]$, since this aspect of the definition is not justified by the discussion preceeding Definition 2. At first there may seem to be no reason to introduce such a "dampening filter" in Bitcoin's target recalculation function and one should let the parties to try collectively to approximate the proper target. Interestingly, in the absence of such dampening, an efficient attack is known [Bah13] (against the common prefix property). As we will see, this dampening is sufficient for us to prove security against all attackers, including those considered in [Bah13] (with foresight, we can say that the attack still holds but it will take exponential time to mount).

4 The Bitcoin Backbone Protocol with Variable Difficulty

In this section we give a high-level description of the Bitcoin backbone protocol with chains of variable difficulty; a more detailed description, including the pseudocode of the algorithms, is given in Appendix B. The presentation is based on the description in [GKL15]. We then formulate two desired properties of the blockchain—*common prefix* and *chain quality*—for the dynamic setting.

4.1 The Protocol

As in [GKL15], in our description of the backbone protocol we intentionally avoid specifying the type of values/content 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. These checks and operations are handled by the external functions

$V(\cdot), I(\cdot)$ and $R(\cdot)$ (the *content validation function*, the *input contribution function* and the *chain reading function*, resp.) which are specified by the application that runs "on top" of the backbone protocol. The Bitcoin backbone protocol in the dynamic setting comprises three algorithms.

Chain validation. The validate algorithm performs a validation of the structural properties of a given chain $\mathcal{C}$. It is given as input the value q, as well as hash functions $H(\cdot), G(\cdot)$. It is parameterized by the content validation predicate predicate $V(\cdot)$ as well as by $D(\cdot)$, the *target calculation function* (Section 3). For each block of the chain, the algorithm checks that the proof of work is properly solved (with a target that is suitable as determined by the target calculation function), and that the counter *ctr* does not exceed q. Furthermore it collects the inputs from all blocks, $\mathbf{x}_{\mathcal{C}}$, and tests them via the predicate $V(\mathbf{x}_{\mathcal{C}})$. Chains that fail these validation procedure are rejected. (Algorithm 1 in App. B.)

Chain comparison. The objective of the second algorithm, called maxvalid (Algorithm 2.), is to find the "best possible" chain when given a set of chains. The algorithm is straightforward and is parameterized by a $\mathsf{max}(\cdot)$ function that applies some ordering to the space of blockchains. The most important aspect is the chains' difficulty in which case $\mathsf{max}(\mathcal{C}_1, \mathcal{C}_2)$ will return the most *difficult* of the two. In case $\text{diff}(\mathcal{C}_1) = \text{diff}(\mathcal{C}_2)$, some other characteristic can be used to break the tie. In our case, $\mathsf{max}(\cdot, \cdot)$ will always return the first operand to reflect the fact that parties adopt the first chain they obtain from the network.

Proof of work. The third algorithm, called pow (Algorithm 3), is the proof of work-finding procedure. It takes as input a chain and attempts to extend it via solving a proof of work. This algorithm is parameterized by two hash functions $H(\cdot), G(\cdot)$ as well as the parameter q. Moreover, the algorithm calls the target calculation function $D(\cdot)$ in order to determine the value T that will be used for the proof of work. The procedure, given a chain $\mathcal{C}$ and a value x to be inserted in the chain, hashes these values to obtain h and initializes a counter *ctr*. Subsequently, it increments *ctr* and checks to see whether $H(ctr, h) < T$; in case a suitable *ctr* is found then the algorithm succeeds in solving the POW and extends chain $\mathcal{C}$ by one block.

The backbone protocol. The core of the backbone protocol with variable difficulty (Algorithm 4 in App. B) is similar to that in [GKL15], with several important distinctions. First is the procedure to follow when the parties become active. Parties check the `ready` flag they possess, which is false if and only if they have been inactive in the previous round. In case the `ready` flag is false, they diffuse a special message '**Join**' to request the most recent version of the blockchain(s). Similarly, parties that receive the special request message in their RECEIVE() tape broadcast their chains. As before parties, run "indefinitely" (our security analysis will apply when the total running time is polynomial in κ). The input contribution function $I(\cdot)$ and the chain reading function $R(\cdot)$ are applied to the values stored in the chain. Parties check their communication tape RECEIVE() to see whether any necessary update of their local chain is due; then they attempt to extend it via the POW algorithm pow. The function $I(\cdot)$ determines the input to be added in the chain given the party's state *st*, the current chain $\mathcal{C}$, the contents of the party's input tape INPUT() and communication tape RECEIVE(). The input tape contains two types of symbols, READ and (INSERT, *value*); other inputs are ignored. In case the local chain $\mathcal{C}$ is extended the new chain is diffused to the other parties. Finally, in case a READ symbol is present in the communication tape, the protocol applies function $R(\cdot)$ to its current chain and writes the result onto the output tape OUTPUT().

4.2 Properties of the Backbone Protocol with Variable Difficulty

Next, we define the two properties of the backbone protocol that the protocol will establish. They are close variants of the properties in [GKL15], suitably modified for the dynamic q-bounded syn-

chronous setting.

The *common prefix* property essentially remains the same. It is parameterized by a value $k \in \mathbb{N}$, considers an arbitrary environment and adversary, and it holds as long as any two parties' chains are different only in their most recent k blocks. It is actually helpful to define the property between an honest party's chain and another chain that may be adversarial. The definition is as follows.

Definition 3 (Common Prefix Property)**.** The *common-prefix* property Q_{cp} with parameter $k \in \mathbb{N}$ states that, at any round of the execution, if a chain $\mathcal{C}$ belongs to an honest party, then for any valid chain $\mathcal{C}'$ in the same round such that either $\mathrm{diff}(\mathcal{C}') > \mathrm{diff}(\mathcal{C})$, or $\mathrm{diff}(\mathcal{C}') = \mathrm{diff}(\mathcal{C})$ and $\mathrm{head}(\mathcal{C}')$ was computed no later than $\mathrm{head}(\mathcal{C})$, it holds that $\mathcal{C}^{\lceil k} \preceq \mathcal{C}'$ and $\mathcal{C}'^{\lceil k} \preceq \mathcal{C}$.

The second property, called *chain quality*, expresses the number of honest-party contributions that are contained in a sufficiently long and continuous part of a party's chain. Because we consider chains of variable difficulty it is more convenient to think of parties' contributions in terms of the total difficulty they add to the chain as opposed to the number of blocks they add (as done in [GKL15]). The property states that adversarial parties are bounded in the amount of difficulty they can contribute to any sufficiently long segment of the chain.

Definition 4 (Chain Quality Property)**.** The *chain quality* property Q_{cq} with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any party P with chain $\mathcal{C}$ in $\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$, and any segment of that chain of difficulty d such that the timestamp of the first block of the segment is at least ℓ smaller than the timestamp of the last block, the blocks the adversary has contributed in the segment have a total difficulty that is at most $\mu \cdot d$.

4.3 Application: Robust Transaction Ledger

We now come to the (main) application the Bitcoin backbone protocol was designed to solve. A *robust transaction ledger* is a protocol maintaining a ledger of transactions organized in the form of a chain $\mathcal{C}$, satisfying the following two properties. (Refer to App. C for a more detailed presentation of ledger terminology.)

- *Persistence:* Parameterized by $k \in \mathbb{N}$ (the "depth" parameter), if an honest party P, maintaining a chain $\mathcal{C}$, reports that a transaction tx is in $\mathcal{C}^{\lceil k}$, then it holds for every other honest party P' maintaining a chain $\mathcal{C}'$ that if $\mathcal{C}'^{\lceil k}$ contains tx, then it is in exactly the same position.
- *Liveness:* Parameterized by $u, k \in \mathbb{N}$ (the "wait time" and "depth" parameters, resp.), if a transaction tx is provided to all honest parties for u consecutive rounds, then it holds that for any player P, maintaining a chain $\mathcal{C}$, tx will be in $\mathcal{C}^{\lceil k}$.

We note that, as in [GKL15], Liveness is applicable to either "neutral" transactions (i.e., those that they are never in "conflict" with other transactions in the ledger), or transactions that are produced by an oracle Txgen that produces honestly generated transactions.

5 Overview of the Analysis

Our main goal is to show that the backbone protocol satisfies the properties common prefix and chain quality (Section 4.2) in a (γ, s)-respecting environment as an intermediate step towards proving, eventually, that the protocol implements a robust transaction ledger. In this section we present a high-level overview of our approach; the full analysis is then presented in Section 6. To prove the aforementioned properties we first characterize the set of *typical* executions. Informally, an execution is typical if for any set S of consecutive rounds the successes of the adversary and the honest parties

do not deviate too much from their expectations and no bad event occurs with respect to the hash function (which we model as a "random oracle"). Using the martingale bound of Theorem 30 we demonstrate that almost all polynomially bounded executions are typical. We then proceed to show that in a typical execution any chain that an honest party adopts (1) contains timestamps that are approximately accurate (i.e., no adversarial block has a timestamp that differs too much by its real creation time) and (2) has a target such that the probability of block production remains near a fixed constant f. Finally, these properties of a typical execution will bring us to our ultimate goal: to demonstrate that a typical execution enjoys the common prefix and the chain quality properties, and therefore one can build on the blockchain a robust transaction ledger (Section 4.3). Here we highlight the main steps and the novel concepts that we introduce.

"Good" executions. In order to be able to talk quantitavely about typical executions, we first introduce the notion of (η, θ)-*good* executions, which expresses how well the parties approximate f. Suppose at round r exactly n parties query the oracle with target T. The probability at least one of them will succeed is

$$f(T, n) = 1 - \left(1 - \frac{T}{2^\kappa}\right)^{qn}.$$

For the initial target T_0 and the initial estimate of the number of parties n_0, we denote $f_0 = f(T_0, n_0)$. Looking ahead, the objective of the target recalculation mechanism is to maintain a target T for each party such that $f(T, n_r) \approx f_0$ for all rounds r. (For succintness, we will drop the subscript and simply refer to it as f.)

Now, at a round r of an execution E the honest parties might be querying the random oracle for various targets. We denote by $T_r^{\min}(E)$ and $T_r^{\max}(E)$ the minimum and maximum over those targets. We say r is a target-recalculation point of a valid chain $\mathcal{C}$, if there is a block with timestamp r and m exactly divides the number of blocks up to (and including) this block. Consider constants $\eta \in (0, 1]$ and $\theta \in [1, \infty)$ and an execution E:

Definition 6 (Abridged). A round r is (η, θ)-*good* in E if $\eta f \leq f(T_r^{\min}(E), n_r)$ and $f(T_r^{\max}(E), n_r) \leq \theta f$. An execution E is (η, θ)-*good* if every round of E was (η, θ)-good.

We are going to study the progress of the honest parties only when their targets lie in a reasonable range. It will turn out that, with high probability, the honest parties always work with reasonable targets. The following bound will be useful because it gives an estimate of the progress the honest parties have made in an (η, θ)-good execution. We will be interested in the progress coming from *uniquely successful rounds*, where exactly one honest party computed a POW. Let Q_r be the random variable equal to the (maximum) difficulty of such rounds (recall a block with target T has difficulty $1/T$); 0 otherwise. We refer to Q_r also as "unique" difficulty. We are able to show the following.

Proposition 8 (Informal). If r is a (η, θ)-good round in E, then $\mathbf{E}[Q_r(E_{r-1})] \geq (1 - \theta f) p n_r$, where $Q_r(E_{r-1})$ is the unique difficulty conditioned on the execution so far, and $p = \frac{q}{2^\kappa}$.

"Per round" arguments regarding relevant random variables are not sufficient, as we need executions with "good" behavior over a sequence of rounds—i.e., variables should be concentrated around their means. It turns out that this is not easy to get, as the probabilities of the experiments performed per round depend on the history (due to target recalculation). To deal with this lack of concentration/variance problem, we introduce the following measure.

Typical executions. Intuitively, the idea that this notion captures is as follows. Note that at each round of a given execution E the parties perform Bernoulli trials with success probabilities possibly affected by the adversary. Given the execution, these trials are determined and we may calculate the expected progress the parties make given the corresponding probabilities. We then

compare this value to the actual progress and if the difference is "reasonable" we declare E *typical*. Note, however, that considering this difference by itself will not always suffice, because the variance of the process might be too high. Our definition, in view of Theorem 30 (App. D), says that either the variance is high with respect to the set of rounds we are considering, or the parties have made progress during these rounds as expected. A bit more formally, for a given random oracle query in an execution E, the history of the execution just before the query takes place, determines the parameters of the distribution that the outcome of this query follows as a POW (a Bernoulli trial). For the queries performed in a set of rounds S, let $V(S)$ denote the sum of the variances of these trials.

Definition 11 (Abridged). An execution E is (ϵ, η, θ)-*typical* if, for any given set S of consecutive rounds such that $V(S)$ is appropriately bounded from above:

- The average unique difficulty is *lower*-bounded by $\frac{1}{|S|}(\sum_{r\in S} \mathbf{E}[Q_r(E_{r-1})] - \epsilon(1-\theta f)p\sum_{r\in S} n_r)$;
- the average maximum difficulty is *upper*-bounded by $\frac{1}{|S|}(1+\epsilon)p\sum_{r\in S} n_r$;
- the adversary's average difficulty of "easy" targets is *upper*-bounded by $\frac{1}{|S|}(1+\epsilon)p\sum_{r\in S} t_r$, while the *number* of blocks with "hard" targets is bounded below m by a suitable constant; and
- no "bad events" with respect to the hash function occur (e.g., collisions).

The following is one of the main steps in our analysis.

Proposition 13 (Informal). Almost all polynomially bounded executions (in κ) are typical. The probability of an execution not being typical is bounded by $\exp(-\Omega(\min\{m, \kappa\}))$.

Recall (Remark 2) that the dynamic setting (specifically, the use of target recalculation functions) offers more opportunities for adversarial attacks [Bah13]. The following important intermediate lemma shows that if a typical execution is good up to a certain point, chains that are privately mined for long periods of time by the adversary will not be adopted by honest parties.

Lemma 14 (Informal). Let E be a typical execution in a (γ, s)-respecting environment. If E_r is (η, θ)-good, then, no honest party adopts at round $r+1$ a chain that has not been extended by an honest party for at least $O(\frac{m}{\tau f})$ consecutive rounds.

An easy corollary of the above is that in typical executions, the honest parties' chains cannot contain blocks with timestamps that differ too much from the blocks' actual creation times.

Corollary 15 (Informal). Let E be a typical execution in a (γ, s)-respecting environment. If E_{r-1} is (η, θ)-good, then the timestamp of any block in E_r is at most $O(\frac{m}{\tau f})$ away from its actual creation time (cf. the notion of *accuracy* in Definition 7).

Additional important results we obtain regarding (η, θ)-good executions are that their epochs last about as much as they should (Lemma 16), as well as a "self-correcting" property, which essentially says that if every chain adopted by an honest party is $(\eta\gamma, \frac{\theta}{\gamma})$-good in E_{r-1} (which we call "super good"), then E_r is (η, θ)-good (Corollary 19). The above (together with several smaller intermediate steps that we omit from this high-level overview) allow us to conclude:

Theorem 21 (Informal). A typical execution in a (γ, s)-respecting environment is $O(\frac{m}{\tau f})$-accurate and (η, θ)-good.

Common prefix and chain quality. Typical executions give us the two desired low-level properties of the blockchain:

Theorems 24 and 25 (Informal). Let E be a typical execution in a (γ, s)-respecting environment. Under the requirements of Table 1 (Section 6.1), common prefix holds for any $k \geq \theta\gamma m/8\tau$ and chain quality holds for $\ell = m/16\tau f$ and $\mu \leq 1 - \delta/2$, where for all r, $t_r < n_r(1-\delta)$.

Robust transaction ledger. Given the above we then prove the properties of the robust transaction ledger:

Theorems 26 and 27 (Informal) Under the requirements of Table 1, the backbone protocol satisfies persistence with parameter $k = \Theta(m)$ and liveness with wait time $u = \Omega(m + k)$ for depth k.

We refer to Section 6 for the full analysis of the protocol.

6 Full Analysis

In this section we present the full analysis and proofs of the backbone protocol and robust transaction ledger application with chains of variable difficulty. The analysis follows at a high level the roadmap presented in Section 5.

6.1 Additional notation, definitions, and preliminary propositions

Our probability space is over all executions of length at most some polynomial in κ. Formally, the set of elementary outcomes can be defined as a set of strings that encode every variable of every party during each round of a polynomially bounded execution. We won't delve into such formalism and leave the details unspecified. We will denote by $\mathbf{Pr}$ the probability measure of this space. Define also the random variable $\mathcal{E}$ taking values on this space and with distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle.

Suppose at round r exactly n parties query the oracle with target T. The probability at least one of them will succeed is

$$f(T, n) = 1 - \left(1 - \frac{T}{2^\kappa}\right)^{qn}.$$

For the initial target T_0 and the initial estimate of the number of parties n_0, we denote $f_0 = f(T_0, n_0)$. Looking ahead, the objective of the target recalculation mechanism would be to maintain a target T for each party such that $f(T, n_r) \approx f_0$ for all rounds r. For this reason, we will drop the subscript from f_0 and simply refer to it as f; to avoid confusion, whenever we refer to the function $f(\cdot, \cdot)$, we will specify its two operands.

Note that $f(T, n)$ is concave and increasing in n and T. In particular, Fact 2 applies. The following proposition provides useful bounds on $f(T, n)$. For convenience, define $p = q/2^\kappa$.

Proposition 5. *For positive integers κ, q, T, n and $f(T, n)$ defined as above,*

$$\frac{pTn}{1 + pTn} \leq f(T, n) \leq pTn \leq \frac{f(T, n)}{1 - f(T, n)}, \quad \textit{where } p = \frac{q}{2^\kappa}.$$

Proof. The bounds can be obtained using the inequalities $(1 - x)^\alpha \geq 1 - x\alpha$, valid for $x \leq 1$ and $\alpha \geq 1$, and $e^{-x} \leq \frac{1}{1+x}$, valid for $x \geq 0$. □

At a round r of an execution E the honest parties might be querying the random oracle for various targets. We denote by $T_r^{\min}(E)$ and $T_r^{\max}(E)$ the minimum and maximum over those targets. We say r is a target-recalculation point of a valid chain $\mathcal{C}$, if there is a block with timestamp r and m exactly divides the number of blocks up to (and including) this block.

We now define two desirable properties of executions which will be crucial in the analysis. We will show later that most executions have these properties.

Definition 6. Consider an execution E and constants $\eta \in (0,1]$ and $\theta \in [1,\infty)$. A target-recalculation point r in a chain $\mathcal{C}$ in E is (η,θ)-*good* if the new target T satisfies $\eta f \leq f(T,n_r) \leq \theta f$. A chain $\mathcal{C}$ in E is (η,θ)-*good* if all its target-recalculation points are (η,θ)-*good*. A round r is (η,θ)-*good* in E if $\eta f \leq f(T_r^{\min}(E),n_r)$ and $f(T_r^{\max}(E),n_r) \leq \theta f$. We say that E is (η,θ)-*good* if every round of E was (η,θ)-good.

For a round r, the following set of chains is of interest. It contains, besides the chains that the honest parties have, those chains that could potentially belong to an honest party.

$$\mathcal{S}_r = \left\{ \mathcal{C} \in E_r \;\middle|\; \begin{array}{l} \text{“}\mathcal{C}\text{ belongs to an honest party” or} \\ \text{“for some chain } \mathcal{C}' \text{ of an honest party } \mathrm{diff}(\mathcal{C}) > \mathrm{diff}(\mathcal{C}')\text{” or} \\ \text{“for some chain } \mathcal{C}' \text{ of an honest party } \mathrm{diff}(\mathcal{C}) = \mathrm{diff}(\mathcal{C}') \text{ and} \\ \qquad\qquad \mathrm{head}(\mathcal{C}) \text{ was computed no later than } \mathrm{head}(\mathcal{C}')\text{”} \end{array} \right\},$$

where $\mathcal{C} \in E_r$ means that $\mathcal{C}$ exists and is valid at round r.

Definition 7. Consider an execution E. For $\epsilon \in [0,\infty)$, a block created at round r is ϵ-*accurate* if it has a timestamp r' such that $|r'-r| \leq \epsilon \frac{m}{f}$. We say that E_r is ϵ-*accurate* if no chain in $\mathcal{S}_r$ contains a block that is not ϵ-accurate. We say that E is ϵ-*accurate* if for every round r in the execution, E_r is ϵ-accurate.

Our next step is to define the typical set of executions. To this end we define a few more quantities and random variables.

In an actual execution E the honest parties may be split across different chains with possibly different targets. We are going to study the progress of the honest parties only when their targets lie in a reasonable range. It will turn out that, with high probability, the honest parties always work with reasonable targets. For a round r, a set of consecutive rounds S, and constant $\eta \in (0,1)$, let

$$T^{(r,\eta)} = \frac{\eta f}{p n_r} \quad \text{and} \quad T^{(S,\eta)} = \min_{r \in S} T^{(r,\eta)}.$$

To expunge the mystery from the definition of $T^{(r,\eta)}$, note that in an (η,θ)-good round all honest parties query for target at least $T^{(r,\eta)}$. We now define for each round r a real random variable D_r equal to the *maximum difficulty* among all blocks with targets at least $T^{(r,\eta)}$ computed by honest parties at round r. Define also Q_r to equal D_r when exactly one block was computed by an honest party and 0 otherwise.

Regarding the adversary, we are going to be interested in periods of time during which he has gathered a number of blocks in the order of m. Given that the targets of blocks are variable themselves, it is appropriate to consider the difficulty acquired by the adversary not in a set of consecutive rounds but rather in a set of consecutive adversarial queries that may span a number of rounds but do are not necessarily a multiple of q.

For a set of consecutive queries indexed by a set J, we define the following value that will act as a threshold for targets of blocks that are attempted adversary.

$$T^{(J)} = \frac{\eta(1-\delta)(1-2\epsilon)(1-\theta f)}{32\tau^3\gamma} \cdot \frac{m}{|J|} \cdot 2^{\kappa}.$$

Given the above threshold, for $j \in J$, if the adversary computed at his j-th query a block of difficulty at most $1/T^{(J)}$, then let the random variable $A_j^{(J)}$ be equal to the difficulty of this block; otherwise, let $A_j^{(J)} = 0$. The above definition suggests that we collect in $A_j^{(J)}$ the difficulty acquired by the adversary as long as it corresponds to blocks that are not too difficult (i.e., those with targets less

than $T^{(J)}$). With foresight we note that this will enable a concentration argument for random variable $A_j^{(J)}$. We will usually drop the superscript (J) from A.

Let $\mathcal{E}_{r-1}$ contain the information of the execution just before round r. In particular, a value E_{r-1} of $\mathcal{E}_{r-1}$ determines the targets against which every party will query the oracle at round r, but it does not determine D_r or Q_r. If E is a fixed execution (i.e., $\mathcal{E} = E$), denote by $D_r(E)$ and $Q_r(E)$ the value of D_r and Q_r in E. If a set of consecutive queries J is considered, then, for $j \in J$, $A_j^{(J)}(E)$ is defined analogously. In this case we will also write $\mathcal{E}_j^{(J)}$ for the execution just before the j-th query of the adversary.

With respect to the random variables defined above, the following bound will be useful because it gives an estimate of the progress the honest parties have made in an (η, θ)-good execution. Note that we are interested in the progress coming from *uniquely successful rounds*, where exactly one honest party computed a POW. The expected difficulty that will be computed by the n_r honest parties at round r is pn_r. However, the easier the POW computation is, the smaller $\mathbf{E}[Q_r|\mathcal{E}_{r-1} = E_{r-1}]$ will be with respect to this value. Since the execution is (η, θ)-good, a POW is computed by the honest parties with probability at most θf. This justifies the appearance of $(1 - \theta f)$ in the bound.

Proposition 8. *If round r is (η, θ)-good in E, then* $\mathbf{E}[Q_r|\mathcal{E}_{r-1} = E_{r-1}] \geq (1 - \theta f)pn_r$.

Proof. Let us drop the subscript r for convenience. Suppose that the honest parties were split into k chains with corresponding targets $T_1 \leq T_2 \leq \cdots \leq T_k = T^{\max}$. Let also $n_1, n_2, \ldots, n_k$, with $n_1 + \cdots + n_k = n$, be the corresponding number of parties with each chain. First note that

$$\prod_{j\in[k]} \big[1 - f(T_j, n_j)\big] \geq \prod_{j\in[k]} \big[1 - f(T^{\max}, n_j)\big] = 1 - f(T^{\max}, n) \geq 1 - \theta f,$$

where the first inequality holds because $f(T, n)$ is increasing in T. Proposition 5 now gives

$$\mathbf{E}[Q_r|\mathcal{E}_{r-1} = E_{r-1}] = \sum_{i\in[k]} \frac{1}{T_i} \cdot \frac{f(T_i, n_i)}{1 - f(T_i, n_i)} \cdot \prod_{j\in[k]} \big[1 - f(T_j, n_j)\big] \geq (1 - \theta f) \sum_{i\in[k]} pn_i.$$

□

The properties we have defined will be shown to hold in a (γ, s)-respecting environment, for suitable γ and s. The following simple fact is a consequence of the definition.

Fact 1. *In a (γ, s)-respecting environment, for any set S of consecutive rounds with $|S| \leq s$, any $S' \subseteq S$, and any $n \in \{n_r : r \in S\}$,*

$$\frac{1}{\gamma} \cdot n \leq \frac{1}{|S'|} \cdot \sum_{r\in S'} n_r \leq \gamma \cdot n.$$

Proof. The average of several numbers is bounded by their min and max. Furthermore, the definition of (γ, s)-respecting implies $\min_{r\in S} n_r \geq \frac{1}{\gamma} \max_{r\in S} n_r \geq \frac{1}{\gamma} n$ and $\max_{r\in S} n_r \leq \gamma \min_{r\in S} \leq \gamma n$. Thus,

$$\frac{1}{\gamma} \cdot n \leq \min_{r\in S} n_r \leq \min_{r\in S'} n_r \leq \frac{1}{|S'|} \cdot \sum_{r\in S'} n_r \leq \max_{r\in S'} n_r \leq \max_{r\in S} n_r \leq \gamma \cdot n.$$

□

Our analysis involves a number of parameters that are suitably related. Table 1 summarizes them, recalls their definitions and lists all the constraints that they should satisfy.

(R0) $\forall r : t_r < (1-\delta)n_r$ (R1) $s \geq \frac{\tau m}{f} + \frac{m}{8\tau f}$ (R2) $\frac{\delta}{2} \geq 2\epsilon + \theta f$ (R3) $\tau - 1/8\tau > 1/(1-\epsilon)(1-\theta f)\eta$ (R4) $17(1+\epsilon)\theta \leq 8\tau(\gamma - \theta f)$ (R5) $9(1+\epsilon)\eta\gamma^2 \leq 4(1-\eta\gamma f)$ (R6) $7\theta(1-\epsilon)(1-\theta f) \geq 8\gamma^2$	n_r: number of honest parties mining in round r. t_r: number of activated parties that are corrupted. δ: advantage of honest parties, $\forall r(t_r/n_r < 1-\delta)$ (γ, s): determines how the number of parties fluctuates across rounds, cf. Definition 1. f: probability at least one honest party succeeds in a round assuming n_0 parties and target T_0 (the protocol's initialization parameters). τ: the dampening filter, see Definition 2. (η, θ): lower and upper bound determining the goodness of an execution, cf. Definition 6. ϵ: quality of concentration of random variables in typical executions, cf. Definition 11. m: the length of an epoch in number of blocks.

Table 1: *Requirements on the parameters.* The parameters are as follows: positive integers s, m; positive reals $f, \gamma, \delta, \epsilon, \tau, \eta, \theta$, where $f, \epsilon, \delta \in (0,1)$, and $0 < \eta \leq 1 \leq \theta$.

Remark 3. We remark that for the actual parameterization of the parameters τ, m, f of Bitcoin[5], i.e., $\tau = 4, m = 2016, f = 0.03$, vis-à-vis the constraints of Table 1, they can be satisfied for $\delta = 0.99, \eta = 0.268, \theta = 1.995, \epsilon = 2.93 \cdot 10^{-8}$, for $\gamma = 1.281$ and $s = 2.71 \cdot 10^5$. Given that s measures the number of rounds within which a fluctuation of γ may take place, we have that the constraints are satisfiable for a fluctuation of up to 28% every approximately 2 months (considering a round to last 18 seconds).

6.2 Chain-Growth Lemma

We now prove the Chain-growth lemma. This lemma appears already in [GKL15], but it refers to number of blocks instead of difficulty. In [KP15] the name "chain growth" appears for the first time and the authors explicitly state a chain-growth property.

Informally, this lemma says that the honest parties will make as much progress as how many POWs they obtain. Although simple to prove, the chain-growth lemma is very important, because it shows that no matter what the adversary does the honest parties will advance (in terms of accumulated difficulty) by at least the difficulty of the POWs they have acquired.

Lemma 9. *Let E be any execution. Suppose that at round u an honest party has a chain of difficulty d. Then, by round $v+1 \geq u$, every honest party will have received a chain of difficulty at least $d + \sum_{r=u}^{v} D_r(E)$.*

Proof. By induction on $v - u$. For the basis, $v + 1 = u$ and $d + \sum_{r=u}^{v} D_r(E) = d$. Observe that if at round u an honest party has a chain $\mathcal{C}$ of difficulty d, then that party broadcast $\mathcal{C}$ at a round earlier than u. It follows that every honest party will receive $\mathcal{C}$ by round u.

For the inductive step, note that by the inductive hypothesis every honest party has received a chain of difficulty at least $d' = d + \sum_{r=u}^{v-1} D_r$ by round v. When $D_v = 0$ the statement follows

[5]Note that in order to calculate f, we can consider that a round of full interaction lasts 18 seconds; If this is combined with the fact that the target is set for a POW to be discovered approximately every 10 minutes, we have that $18/600 = 0.3$ is a good estimate for f.

directly, so assume $D_v > 0$. Since every honest party queried the oracle with a chain of difficulty at least d' at round v, if follows that an honest party successful at round v broadcast a chain of difficulty at least $d' + D_v = d + \sum_{r=u}^{v} D_r$. □

6.3 Typical Executions: Definition and Related Proofs

We can now define formally our notion of *typical* executions. Intuitively, the idea that this definition captures is as follows. Suppose that we examine a certain execution E. Note that at each round of E the parties perform Bernoulli trials with success probabilities possibly affected by the adversary. Given the execution, these trials are determined and we may calculate the expected progress the parties make given the corresponding probabilities. We then compare this value to the actual progress and if the difference is reasonable we declare E *typical*. Note, however, that considering this difference by itself will not always suffice, because the variance of the process might be too high. Our definition, in view of Theorem 30, says that either the variance is high with respect to the set of rounds we are considering, or the parties have made progress during these rounds as expected.

Beyond the behavior of random variables described above, a typical execution will also be characterized by the absence of a number of bad events about the underlying hash function $H(\cdot)$ which is used in proofs of work and is modeled as a random oracle. The bad events that are of concern to us are defined as follows.

Definition 10. An *insertion* occurs when, given a chain $\mathcal{C}$ with two consecutive blocks B and B', a block B^* created after B' is such that B, B^*, B' form three consecutive blocks of a valid chain. A *copy* occurs if the same block exists in two different positions. A *prediction* occurs when a block extends one with earlier creation time.

Given the above we are not ready to specify what is a typical execution.

Definition 11 (Typical execution)**.** An execution E is (ϵ, η, θ)-*typical* if the following hold:

(a) If, for any set S of consecutive rounds, $pT^{(S,\eta)} \sum_{r\in S} n_r \geq \frac{\eta m}{16\tau\gamma}$, then

$$\sum_{r\in S} Q_r(E) \geq \sum_{r\in S} \mathbf{E}[Q_r|\mathcal{E}_{r-1} = E_{r-1}] - \epsilon(1-\theta f)p\sum_{r\in S} n_r \quad \text{and} \quad \sum_{r\in S} D_r(E) \leq (1+\epsilon)p\sum_{r\in S} n_r.$$

(b) For any set J indexing a set of consecutive queries of the adversary we have

$$\sum_{j\in J} A_j(E) \leq (1+\epsilon)2^{-\kappa}|J|$$

and during these queries the adversary has acquired (strictly) less than $\frac{\eta(1-\epsilon)(1-\theta f)}{32\tau^2\gamma} \cdot m$ blocks with targets (strictly) less than $\tau T^{(J)}$.

(c) No insertions, no copies, and no predictions occurred in E.

Remark 4. Note that if J indexes the queries of the adversary in a set S of consecutive rounds, then $|J| = q\sum_{r\in S} t_r$ and the inequality in Definition 11(b) reads $\sum_{j\in J} A_j(E) \leq (1+\epsilon)p\sum_{r\in S} t_r$.

The next proposition simplify our applications of Definition 11(a).

Proposition 12. *Assume E is a typical execution in a (γ, s)-respecting environment. For any set S of consecutive rounds with $|S| \geq \frac{m}{16\tau f}$,*

$$\sum_{r\in S} D_r \leq (1+\epsilon)p\sum_{r\in S} n_r.$$

If in addition, E is (η, θ)-good, then

$$\sum_{r \in S} Q_r \geq (1-\epsilon)(1-\theta f) p \sum_{r \in S} n_r$$

and any block computed by an honest party at any round r corresponds to target at least $T^{(r,\eta)}$, and so contributes to the random variables D_r and Q_r (if the r was uniquely successful).

Proof. We first partition S into several parts with size at least $\frac{m}{16\tau f}$ and at most s. In view of Proposition 8, for both of the inequalities, we only need to verify the 'if' part of Definition 11(a) for each part S' of S. Indeed, by the definition of $T^{(S',\eta)}$ and Fact 1, $pT^{(S',\eta)} \sum_{r \in S'} n_r \geq \eta f |S'|/\gamma \geq \frac{\eta m}{16\tau\gamma}$. The last part, in view of the definition of $T^{(r,\eta)}$, is equivalent to r being (η, θ)-good. □

Almost all polynomially bounded executions (in κ) are typical

Proposition 13. *Assuming the ITM system $(\mathcal{Z}, C)$ runs for L steps, the event "$\mathcal{E}$ is not typical" is bounded by* $\exp\{-\frac{\eta\epsilon^2(1-2\delta)m}{64\tau^3\gamma} + 2(\ln L + \ln 2)\} + 2^{-\kappa+1+2\log L}$.

Proof. Since the length of the execution, L, is fixed we will prove the stated bound for a fixed set of consecutive rounds S and then apply a union bound over all such sets in the length of the execution. Let k be the size of S and identify it, without loss of generality, with $[k] = \{1, 2, \ldots, k\}$. For part (a), define a sequence of random variables by

$$X_0 = 0; \;\; X_r = \sum_{i \in [r]} Q_i - \sum_{i \in [r]} \mathbf{E}[Q_i | \mathcal{E}_{i-1}], \;\; r \in [k].$$

This forms a martingale with respect to the sequence $\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_k$, because (recalling basic properties of conditional expectation [McD98]),

$$\mathbf{E}[X_r | \mathcal{E}_{r-1}] = \mathbf{E}\big[Q_r - \mathbf{E}[Q_r|\mathcal{E}_{r-1}]\big|\mathcal{E}_{r-1}\big] + \mathbf{E}[X_{r-1}|\mathcal{E}_{r-1}] = X_{r-1}.$$

Specifically, the above follows from linearity of conditional expectation, and the fact that X_{r-1} is a deterministic function of $\mathcal{E}_{r-1}$.

Now suppose the first inequality of Definition 11(a) fails. Note that the probability of this event is equal to $\mathbf{Pr}[X_k < X_0 - t]$, for $t = \epsilon(1-\theta f)p\sum_{r\in S} n_r$. For b and V defined with respect to Theorem 30, we have $b \leq 1/T^{(S,\eta)}$ and $V \leq v$, where $v = p\sum_{r\in S} n_r / T^{(S,\eta)}$. To prove the bound on V, note that

$$\mathrm{var}(X_r - X_{r-1}|\mathcal{E}_{r-1}) = \mathbf{E}\big[\big(Q_r - \mathbf{E}[Q_r|\mathcal{E}_{r-1}]\big)^2\big|\mathcal{E}_{r-1}\big] = \mathbf{E}[Q_r^2|\mathcal{E}_{r-1}] - \big(\mathbf{E}[Q_r|\mathcal{E}_{r-1}]\big)^2.$$

Thus, it suffices to show $\mathbf{E}[Q_r^2|\mathcal{E}_{r-1}] \leq pn_r/T^{(r,\eta)} \leq pn_r/T^{(S,\eta)}$. To this end, suppose that the honest parties at round r were split into ℓ chains with corresponding targets $T^{(r,\eta)} \leq T_1 \leq T_2 \leq \cdots \leq T_\ell$. Let also $\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_\ell$, with $\hat{n}_1 + \cdots + \hat{n}_\ell \leq n_r$, be the corresponding number of parties with each chain. Then, for any E_{r-1},

$$\mathbf{E}[Q_r^2|\mathcal{E}_{r-1} = E_{r-1})] = \sum_{i\in[\ell]} \frac{1}{T_i^2} \cdot f(T_i, \hat{n}_i) \cdot \prod_{j\neq i}[1 - f(T_j, \hat{n}_j)] \leq \sum_{i\in[\ell]} \frac{p\hat{n}_i}{T_i} \leq \frac{pn_r}{T^{(r,\eta)}}.$$

Applying Theorem 30 on $-X_0, -X_1, \ldots$, note that $V \leq v$ always holds, and recalling the condition $pT^{(S,\eta)}\sum_{r\in S} n_r \geq \frac{\eta m}{16\tau\gamma}$, we obtain $\exp\{-\frac{3\epsilon^2(1-\theta f)^2\eta m}{32(3+\epsilon)\tau\gamma}\} \leq \exp\{-\frac{\epsilon^2(1-\delta)\eta m}{32\tau\gamma}\}$ (using Requirement (R2)).

For the bounds on $\sum_{r\in S} D_r(E)$ and $\sum_{j\in J} A_j(E)$ the proof follows the same lines. In particular, replace Q by D and A (in the case of A the martingale will be indexed by J) and note that in these cases the martingale need not be negated.

In more details, regarding the bound on D in part (a), using the same notation as above, we have that

$$\mathbf{E}[D_r|\mathcal{E}_{r-1}=E_{r-1})] = \sum_{i\in[\ell]} \frac{1}{T_i}\cdot f(T_i,\hat{n}_i)\cdot \prod_{j=1}^{i-1}\big[1-f(T_j,\hat{n}_j)\big] \leq \sum_{i\in[\ell]} p\hat{n}_i \leq pn_r$$

and so

$$\sum_{r\in S}\mathbf{E}[D_r|\mathcal{E}_{r-1}] \leq p\sum_{r\in S} n_r.$$

A similar argument provides the bound $\mathbf{E}[D_r^2|\mathcal{E}_{r-1}=E_{r-1})] \leq pn_r/T^{(r,\eta)}$ from which we can obtain the bound on variance $V \leq v = p\sum_{r\in S} n_r/T^{(S,\eta)}$.

We next focus on part (b). First we will show that for the martingale $X_0, X_1, X_2, \ldots$ with respect to $\mathcal{E}_0^{(J)}, \mathcal{E}_1^{(J)}, \mathcal{E}_2^{(J)}, \ldots$ that is defined as

$$X_0 = 0;\;\; X_j = \sum_{i\in[j]} A_i - \sum_{i\in[j]} \mathbf{E}[A_i|\mathcal{E}_{i-1}^{(J)}],\;\; j\in J,$$

it holds that $b \leq 1/T^{(J)}$ and $V \leq v$ for $v = 2^{-\kappa}|J|/T^{(J)}$, for the quantities b, V defined as in Theorem 30. Consider an execution prefix $E_{j-1}^{(J)}$ and the target that is selected by the adversary in its j-th query. Note that we can associate such value to any query of the form (ctr, g) where $g = G(r, st, x)$ by recovering the chain that corresponds to st. If such value is below $T^{(J)}$, or is not defined, $A_j = 0$. Thus, we have $\mathbf{E}[A_j|\mathcal{E}_{j-1}^{(J)} = E_{j-1}^{(J)}] \leq 2^{-\kappa}$ and $\mathbf{E}[A_j|\mathcal{E}_{j-1}^{(J)} = E_{j-1}^{(J)}] \leq 2^{-\kappa}$. From the above follows the inequality

$$\sum_{j\in J}\mathbf{E}[A_j|\mathcal{E}_{j-1}^{(J)}] \leq 2^{-\kappa}|J|$$

and the bound on V. We now have the following by setting $t = \epsilon 2^{-\kappa}|J|$.

$$\begin{aligned}\mathbf{Pr}\Big[\sum_{j\in J} A_j > (1+\epsilon)2^{-\kappa}|J|\Big] &= \mathbf{Pr}\Big[\sum_{j\in J} A_j > t + 2^{-\kappa}|J|\Big] \leq \mathbf{Pr}\Big[\sum_{j\in J} A_j > t + \sum_{j\in J}\mathbf{E}[A_j|\mathcal{E}_{j-1}^{(J)}]\Big] \\ &\leq \mathbf{Pr}\Big[\sum_{j\in J}(A_j - \mathbf{E}[A_j|\mathcal{E}_{j-1}^{(J)}]) > t] \leq \exp\Big\{-\frac{t^2}{2v+2bt/3}\Big\} \leq \exp\Big\{-\frac{3\epsilon^2 2^{-\kappa}|J|T^{(J)}}{(6+2\epsilon)}\Big\}.\end{aligned}$$

Recalling $T^{(J)} = \frac{\eta(1-\delta)(1-2\epsilon)(1-\theta f)}{32\tau^3\gamma}\cdot\frac{m}{|J|}\cdot 2^{\kappa}$, we obtain the bound $\exp\{-\frac{3\eta\epsilon^2(1-\delta)(1-\theta f)(1-2\epsilon)m}{64(3+\epsilon)\tau^3\gamma}\}$. Using Requirement (R2) this can be shown to be at most $\exp\{-\frac{\eta\epsilon^2(1-2\delta)m}{64\tau^3\gamma}\}$.

Regarding the second part of part (b), in order to bound the number of blocks of target less than $T^{(J)}$ the adversary can acquire, define a Boolean random variable Z_j, for each $j\in J$ as follows. If the corresponding target is less than $T^{(J)}$ and the query was successful, then $Z_j = 1$, otherwise $Z_j = 0$. We can then define a martingale as in part (a), by letting $k = |J|$ and replacing Q with Z. We have $b \leq 1$ and $V \leq 2^{-\kappa}\tau T^{(J)}$. Since

$$\sum_{j\in[|J|]}\mathbf{E}[Z_j|\mathcal{E}_{j-1}^{(J)}] \leq 2^{-\kappa}\tau T^{(J)}|J| = \frac{\eta(1-2\epsilon)(1-\theta f)}{32\tau^2\gamma}\cdot m$$

and $(1+\epsilon)(1-2\epsilon) < (1-\epsilon)$, for $t = \epsilon \cdot \frac{\eta(1-2\epsilon)(1-\theta f)}{32\tau^2\gamma} \cdot m$ we have (using Requirement (R2) to simplify in the last step)

$$\mathbf{Pr}\Big[\sum_{j\in[|J|]} Z_j > \frac{\eta(1-\epsilon)(1-\theta f)}{32\tau^2\gamma} \cdot m\Big] \leq \mathbf{Pr}[X_k > X_0 + t] \leq \exp\Big\{-\frac{\eta\epsilon^2(1-\delta/2)m}{32\tau^2\gamma}\Big\}.$$

For part (c) and $i \in \{0,1,2,3\}$, let $B_i = \langle r_i, st_i, x_i, ctr_i\rangle$ and $g_i = G(r_i, st_i, x_i)$. If a block extends two distinct blocks, then a collision has occurred. To see this, suppose block B_3 extents two distinct blocks B_1 and B_2. Then $st_3 = H(ctr_1, g_1) = H(ctr_2, g_2)$; implying a collision either in H or in G, since B_1 and B_2 are distinct.

The existence of an insertion or a copy implies a collision as well. Suppose the adversary inserts a block B_2 among two existing blocks B_1 and B_3. Then, B_3 extends both B_1 and B_2 and since B_2 extends B_1, $r_1 < r_2$ and the blocks are distinct. Similarly, if B_3 is a copy of B_1 (i.e., $B_3 = B_1$), then there exist two distinct blocks B_2 and B_0 that are both extended by the same block. To see this, note that either B_0 and B_2 are the ones that B_1 and B_3 extend, or if these are not distinct, then B_2 is a copy of B_0 and so on. Eventually, two distinct blocks will be reached, since B_1 and B_3 are assumed to be on different chains. If the total running time of the system of ITM's is L then it holds that there are at most L queries posed to G, H. It follows that the probability of a collision occurring is $\binom{L}{2}2^{-\kappa+1} \leq 2^{-\kappa+1+2\log L}$.

Finally, note that, for polynomially many rounds in κ, the probability that a guessed block occurs is exponentially small in κ. □

6.4 Typical Executions are Good and Accurate

Lemma 14. *Let E be a typical execution in a (γ, s)-respecting environment. If E_r is (η, θ)-good, then $\mathcal{S}_{r+1}$ contains no chain that has not been extended by an honest party for at least $\frac{m}{16\tau f}$ consecutive rounds.*

Proof. Suppose—towards a contradiction—$\mathcal{C} \in \mathcal{S}_{r+1}$ and has not been extended by an honest party for at least $\frac{m}{16\tau f}$ rounds. Without loss of generality we may assume that $r+1$ is the first such round.

Let $r^* \leq r$ denote the greatest timestamp among the blocks of $\mathcal{C}$ computed by honest parties ($r^* = 0$ if none exists). Define $S = \{r^*+1, \ldots, r\}$ with $|S| \geq \frac{m}{16\tau f}$ and the index-set of the corresponding set of queries $J = \{1, \ldots, q\sum_{r\in S} t_r\}$. Suppose that the blocks of $\mathcal{C}$ with timestamps in S span k epochs with corresponding targets $T_1, \ldots, T_k$. For $i \in [k]$ let m_i be the number of blocks with target T_i and set $M = m_1 + \cdots + m_k$.

Our plan is to contradict the assumption that $\mathcal{C} \in \mathcal{S}_{r+1}$, by showing that the honest parties have accumulated more difficulty than the adversary. To be precise, note that the blocks $\mathcal{C}$ has gained in S sum to $\sum_{i\in[k]} \frac{m_i}{T_i}$ difficulty. On the other hand, by the Chain-Growth Lemma 9, all the honest parties have advanced during the rounds in S by $\sum_{r\in S} D_r(E) \geq \sum_{r\in S} Q_r(E)$. Since $|S| \geq \frac{m}{16\tau f}$, Proposition 12 implies that $\sum_{r\in S} Q_r(E)$ is at least $(1-\epsilon)(1-\theta f)p\sum_{r\in S} n_r$. Therefore, to obtain a contradiction, it suffices to show that

$$\sum_{i\in[k]} \frac{m_i}{T_i} < (1-\epsilon)(1-\theta f)p\sum_{r\in S} n_r. \tag{1}$$

We proceed by considering cases on M.

First, suppose $M \geq 2M'$, where $M' = \frac{\eta(1-\epsilon)(1-\theta f)}{32\tau^2\gamma} \cdot m$ (see Definition 11(b)). Partition the part of $\mathcal{C}$ with these M blocks into ℓ parts, so that each part has the following properties: (1) it contains at most one target-calculation point, and (2) it contains at least M' blocks with the

same target. Note that such a partition exists because $M \geq 2M'$ and $M' < m$. For $i \in [\ell]$, let $j_i \in J$ be the index of the query during which the last block of the i-th part was computed. Set $J_i = \{j_{i-1}+1, \ldots, j_i\}$, with $j_0 = 0$. Note that Definition 11(c) implies $j_{i-1} < j_i$, and this is a partition of J. Recalling Definition 11(b), the sum of the difficulties of all the blocks in the i-th part is at most $\sum_{j \in J_i} A_j(E)$. This holds because one of the targets is at least $\tau T^{(J_i)}$ (since more than M' blocks have been computed in J_i with this target) and so both are at least $T^{(J_i)}$ (since targets with at most one calculation point between them can differ by a factor at most τ). Thus,

$$\sum_{i \in [k]} \frac{m_i}{T_i} \leq \sum_{i \in [\ell]} \sum_{j \in J_i} A_j(E) \leq \sum_{i \in [\ell]} (1+\epsilon) 2^{-\kappa} |J_i| = (1+\epsilon) p \sum_{r \in S} t_r < (1+\epsilon)(1-\delta) p \sum_{r \in S} n_r,$$

where in the last step we used Requirement (R0). Requirement (R1) implies $(1+\epsilon)(1-\delta) \leq (1-\epsilon)(1-\theta f))$; thus, Equation (1) holds concluding the case $M \geq 2M'$.

Otherwise, $k \leq 2$ and $m_1 + m_2 < 2M'$. Let S' consist of the first $\frac{m}{16\tau f}$ rounds of S. We are going to argue that in this case Equation (1) holds even for S' in the place of S. Since we are in a (γ, s)-respecting environment, by Fact 1, $\gamma \sum_{r \in S'} n_r \geq n_{r^*} |S'|$. Furthermore, since r^* is (η, θ)-good, $T_1 \geq T^{(r^*, \eta)} = \eta f / p n_{r^*}$. Recalling also that $T_2 \geq T_1 / \tau$, we have

$$\frac{m_1}{T_1} + \frac{m_2}{T_2} \leq \frac{m_1 + \tau m_2}{T_1} \leq \frac{\tau M}{T^{(r^*,\eta)}} < \frac{2\tau M' p n_{r^*}}{\eta f} \leq \frac{2\tau\gamma M' p \sum_{r \in S'} n_r}{\eta f |S'|} \leq \frac{32 \tau^2 \gamma M' p \sum_{r \in S} n_r}{\eta m}.$$

and, after substituting M', Equation (1) holds concluding this case as well as the proof. □

Corollary 15. *Let E be a typical execution in a (γ, s)-respecting environment. If E_{r-1} is (η, θ)-good, then E_r is $\frac{m}{16\tau f}$-accurate.*

Proof. Suppose—towards a contradiction—that, for some $r^* \leq r$, $\mathcal{C} \in \mathcal{S}_{r^*}$ contains a block which is not $\frac{m}{16\tau f}$-accurate and let $u \leq r^* \leq r$ be the timestamp of this block and v its creation time. If $u - v > \frac{m}{16\tau f}$, then every honest party would consider $\mathcal{C}$ to be invalid during rounds $v, v+1, \ldots, u$. If $v - u > \frac{m}{16\tau f}$, then in order for $\mathcal{C}$ to be valid it should not contain any honest block with timestamp in $u, u+1, \ldots, v$. (Note that we are using Definition 11(c) here as a block could be inserted later.) In either case, $\mathcal{C} \in \mathcal{S}_{r^*}$, but has not been extended by an honest party for at least $\frac{m}{16\tau f}$ rounds. Since E_{r^*-1} is (η, θ)-good, the statement follows from Lemma 14. □

Lemma 16. *Let E be a typical execution in a (γ, s)-respecting environment and r^* an $(\eta\gamma, \frac{\theta}{\gamma})$-good target-recalculation point of a valid chain $\mathcal{C}$. For $r > r^* + \frac{\tau m}{f}$, assume E_{r-1} is (η, θ)-good. Then, either the duration Δ of the epoch of $\mathcal{C}$ starting at r^* satisfies*

$$\frac{m}{\tau f} \leq \Delta \leq \frac{\tau m}{f},$$

or $\mathcal{C} \notin \mathcal{S}_u$ for each $u \in \{r^* + \frac{\tau m}{f}, \ldots, r\}$.

Proof. Let T be the target of the epoch in question.

For the upper bound, assume $\Delta > \frac{\tau m}{f}$. We show first that in the rounds $S = \{r^* + \frac{m}{16\tau f}, \ldots, r^* + \frac{\tau m}{f} - \frac{m}{16\tau f}\}$ the honest parties have acquired more than $\frac{m}{T}$ difficulty. Note that the rounds of S are (η, θ)-good as they come before r. Thus, by Proposition 12, the difficulty acquired in S by the honest parties is at least

$$(1-\epsilon)(1-\theta f) p \sum_{r \in S} n_r \geq (1-\epsilon)(1-\theta f) p \cdot \frac{|S| n_{r^*}}{\gamma} \geq (1-\epsilon)(1-\theta f) |S| \frac{\eta f}{T} > \frac{m}{T}.$$

For the first inequality, we used Fact 1. For the second, recall that r^* is $(\eta\gamma, \theta/\gamma)$-good and so $pTn_{r^*} \geq f(T, n_{r^*}) \geq \eta\gamma f$. For the last inequality observe that $|S| = \frac{m}{f}(\tau - 1/8\tau)$ and thus follows from Requirement (R3).

Next, we observe that chain $\mathcal{C}$ either has a block within the epoch in question that is computed by an honest party in a round within the period $[r^*, r^* + \frac{m}{16\tau f})$, or by Lemma 14, $\mathcal{C} \notin \mathcal{S}_u$ for each $u \in \{r^* + \frac{m}{16\tau f}, \ldots, r\} \supseteq \{r^* + \frac{\tau m}{f}, \ldots, r\}$. Assuming the first happens, it follows that by round $r^* + \frac{\tau m}{f} - \frac{m}{16\tau f}$ the honest parties' chains have advanced by an amount of difficulty which exceeds the total difficulty of the epoch in question. This means that no honest party will extend $\mathcal{C}$ during the rounds $\{r^* + \frac{\tau m}{f} - \frac{m}{16\tau f} + 1, \ldots, \Delta\}$. Since it is assumed $\Delta > r^* + \frac{\tau m}{f}$, Lemma 14 can then be applied to imply that $\mathcal{C} \notin \mathcal{S}_u$ for $u \in \{r^* + \frac{\tau m}{f}, \ldots, r\}$.

For the lower bound, we assume $\Delta < \frac{m}{\tau f}$ and that $\mathcal{C} \in \mathcal{S}_u$ for some $u \in \{r^* + \Delta + 1, \ldots, r\}$, and seek a contradiction. Clearly, the honest parties contributed only during the set of rounds $S = \{r^*, \ldots, r^* + \Delta\}$. The adversary, by Lemma 14, may have contributed only during $S' = \{r^* - \frac{m}{16\tau f}, \ldots, r^* + \Delta + \frac{m}{16\tau f}\}$. Let J be the set of queries available to the adversary during the rounds in S'. We show that in a typical execution the honest parties together with the adversary cannot acquire difficulty $\frac{m}{T}$ in the rounds in the sets S and S' respectively. With respect to the honest parties, Proposition 12 applies. Regarding the adversary, assume first $T \geq T^{(J)}$ (it is not hard to verify that the case $T < T^{(J)}$ leads to a more favorable bound). It follows that the total difficulty contributed to the epoch is at most

$$(1+\epsilon)p\left(\sum_{r\in S} n_r + \sum_{r\in S'} t_r\right) \leq (1+\epsilon)p\gamma n_{r^*}(|S| + |S'|) < (1+\epsilon)p\gamma n_{r^*} \cdot \frac{17m}{8\tau f} \leq \frac{17(1+\epsilon)\theta}{8\tau(\gamma - \theta f)} \cdot \frac{m}{T}.$$

The first inequality follows from Fact 1 using $t_r < (1-\delta)n_r$. For the second substitute the upper bounds on the sizes of S and S'. Next, note that r^* is an $(\eta\gamma, \theta/\gamma)$-good recalculation point and so $f(T, n_{r^*}) \leq \theta f/\gamma$. By Proposition 5, $pTn_{r^*} < f(T, n_{r^*})/(1 - f(T, n_{r^*})) \leq (\theta f/\gamma)/(1 - \theta f/\gamma)$. From this the last inequality follows and Requirement (R4) makes this less than $\frac{m}{T}$ as desired. □

Proposition 17. *Assume E is a typical execution in a (γ, s)-respecting environment. Consider a round r and a set of consecutive rounds S with $|S| \geq \frac{m}{32\tau^2 f}$. If E_{r-1} is (η, θ)-good, then the adversary, during the rounds in S, has contributed at most $(1-\delta)(1+\epsilon)p\sum_{r\in S} n_r$ difficulty to $\mathcal{S}_r$.*

Proof. Without loss of generality, we will assume in this proof that $t_r = (1-\delta)n_r$ for each $r \in S$. Furthermore, we assume $|S| \leq \frac{\tau m}{f}$. If this is not the case, then we can partition S to parts of appropriate sizes and apply the arguments that follow to each sum. The statement will follow upon summing over all parts.

By Lemma 14, for any block B in $\mathcal{S}_r$, there is a block in the same chain and computed at most $\frac{m}{16\tau f}$ rounds earlier than it. By Lemma 16, there is at most one recalculation point between them. Let u be the round the honest party computed this block and T its target. Note that since E is (η, θ)-good, $T \geq T^{(u,\eta)} = \frac{\eta f}{pn_u}$ and the target of B is at least $\tau^{(-1)}T$. We are going to show that, with J the set of queries that correspond to S, we have $\tau^{-1}T \geq T^{(J)}$. This will suffice, because $(1-\delta)(1+\epsilon)p\sum_{r\in S} n_r \geq (1+\epsilon)p\sum_{r\in S} t_r$, and this is at least $\sum_{j\in J} A_j$ in a typical execution (Definition 11(b)).

Note first that, using Fact 1 and the lower-bound on $|S|$,

$$2^{-\kappa}|J| = (1-\delta)p\sum_{r\in S} n_r \geq (1-\delta)p\frac{|S|n_u}{\gamma} \geq (1-\delta)p\frac{mn_u}{32\tau^3 f\gamma}.$$

Recalling the definition of $T^{(J)}$ and using this bound,

$$T^{(J)} = \frac{\eta(1-\delta)(1-2\epsilon)(1-\theta f)}{32\tau^3\gamma} \cdot \frac{m}{|J|} \cdot 2^{\kappa} \leq \frac{\eta f(1-2\epsilon)(1-\theta f)}{\tau p n_u} < \frac{T^{(u,\eta)}}{\tau} \leq \frac{T}{\tau},$$

as desired. □

Lemma 18. *Let E be a typical execution in a (γ, s)-respecting environment and assume E_{r-1} is (η, θ)-good. If $\mathcal{C} \in \mathcal{S}_r$, then $\mathcal{C}$ is $(\eta\gamma, \theta/\gamma)$-good in E_r.*

Proof. Note that it is our assumption that every chain is $(\eta\gamma, \theta/\gamma)$-good at the first round. Therefore, to prove the statement, it suffices to show that if a chain is $(\eta\gamma, \theta/\gamma)$-good at a recalculation point r^*, then it will also be $(\eta\gamma, \theta/\gamma)$-good at then next recalculation point $r^* + \Delta$.

Let r^* and $r^* + \Delta \leq r$ be two consecutive target-calculation points of a chain $\mathcal{C}$ and T the target of the corresponding epoch. By Lemma 16 and Definition 2 of the target-recalculation function, the new target will be

$$T' = \frac{\Delta}{m/f} \cdot T,$$

where Δ is the duration of the epoch.

We wish to show that

$$\eta\gamma f \leq f(T', n_{r^*+\Delta}) \leq \theta f/\gamma.$$

To this end, let $S = \{r^*, \ldots, r^* + \Delta\}$, $S' = \big\{\max\{0, r^* - \frac{m}{16\tau f}\}, \ldots, \min\{r^* + \Delta + \frac{m}{16\tau f}, r\}\big\}$, and let J index the queries available to the adversary in S'. Note that, by Corollary 15, every block in the epoch was computed either by an honest party during a round in S or by the adversary during a round in S'.

Suppose—towards a contradiction—that $f(T', n_{r^*+\Delta}) < \eta\gamma f$. Using the definition of $f(T, n)$, this implies $q n_{r^*+\Delta} \ln\big(1 - \frac{T'}{2^{\kappa}}\big) > \ln(1 - \eta\gamma f)$. Applying the inequality $-\frac{x}{1-x} < \ln(1-x) < -x$, valid for $x \in (0, 1)$, substituting the expression for T' above and rearranging, we obtain

$$\frac{m}{T} > \frac{1 - \eta\gamma f}{\eta\gamma} \cdot p\Delta n_{r^*+\Delta}.$$

By Propositions 12 and 17 it follows that

$$\frac{m}{T} \leq 2(1+\epsilon)p \sum_{r \in S'} n_r \leq 2(1+\epsilon)p \cdot \frac{\Delta + \frac{m}{8\tau f}}{|S'|} \cdot \sum_{r \in S'} n_r.$$

By Lemma 16, $\Delta \geq \frac{m}{\tau f}$. Thus, $\frac{\Delta + \frac{m}{8\tau f}}{\Delta} \leq \frac{9}{8}$. Using this, Requirement (R5), and combining the inequalities on $\frac{m}{T}$,

$$\gamma n_{r^*+\Delta} < \frac{9(1+\epsilon)\eta\gamma^2}{4(1-\eta\gamma f)} \cdot \frac{1}{|S'|} \sum_{r \in S'} n_r \leq \frac{1}{|S'|} \sum_{r \in S'} n_r,$$

contradicting Fact 1.

For the upper bound, assume $f(T', n_{r^*+\Delta}) > \theta f/\gamma$, which (see Proposition 5) implies

$$\frac{m}{T} < \frac{\gamma}{\theta} \cdot p\Delta n_{r^*+\Delta}.$$

Set $S = \{r^* + \frac{m}{16\tau f}, \ldots, r^* + \Delta - \frac{m}{16\tau f}\}$. Since an honest party posses $\mathcal{C}$ at round r, it follows by Lemma 14 that there is a block computed by an honest party in $\mathcal{C}$ during $\{r^*, \ldots, r^* + \frac{m}{16\tau f} - 1\}$

and one during $\{r^* + \Delta - \frac{m}{16\tau f} + 1, \ldots, r^* + \Delta\}$. By the Chain-Growth Lemma 9, it follows that the honest parties computed less than $\frac{m}{T}$ difficulty during S. In particular,

$$\frac{m}{T} > (1-\epsilon)(1-\theta f)p \sum_{r \in S} n_r \geq (1-\epsilon)(1-\theta f)p \cdot \frac{\Delta - \frac{m}{8\tau f}}{|S|} \cdot \sum_{r \in S} n_r.$$

By Lemma 16, $\Delta \geq \frac{m}{\tau f}$. Thus, $\frac{\Delta - \frac{m}{8\tau f}}{\Delta} \geq \frac{7}{8}$. Using this, Requirement (R6), and combining the inequalities on $\frac{m}{T}$,

$$\frac{n_{r^*+\Delta}}{\gamma} > \frac{7\theta}{8\gamma^2}(1-\epsilon)(1-\theta f) \cdot \frac{1}{|S|} \sum_{r \in S} n_r \geq \frac{1}{|S|} \sum_{r \in S} n_r,$$

contradicting Fact 1. □

Corollary 19. *Let E be a typical execution in a (γ, s)-respecting environment and E_{r-1} be (η, θ)-good. If every chain in $\mathcal{S}_{r-1}$ is $(\eta\gamma, \frac{\theta}{\gamma})$-good, then E_r is (η, θ)-good.*

Proof. We use notations and definitions of Lemma 16. Let $\mathcal{CS}_r$ and let r^* be its last recalculation point in E_{r-1}. Let T be the target after r^* and T' the one at r. We need to show that $f(T', n_r) \in [\eta f, \theta f]$. Note that if r is a recalculation point, this follows by Lemma 18. Otherwise, $T' = T$ and $\eta\gamma \leq f(T, n_{r^*}) \leq \theta f/\gamma$. Using Lemma 16, $r - r^* \leq \Delta \leq \frac{\tau m}{f}$. Thus, $\frac{1}{\gamma} n_{r^*} \leq n_r \leq \gamma n_{r^*}$. By Fact 2 we have $f(T, n_r) \leq f(T, \gamma n_{r^*}) \leq \gamma f(T, n_{r^*}) \leq \theta f$ and $f(T, n_r) \geq f(T, \frac{1}{\gamma} n_{r^*}) \geq \frac{1}{\gamma} f(T, n_{r^*}) \geq \eta f$. □

Corollary 20. *Let E be a typical execution in a (γ, s)-respecting environment. Then every round is (η, θ)-good in E.*

Proof. For the sake of contradiction, let r be the smallest round of E that is not (η, θ)-good. This means that there is a chain $\mathcal{C}$ and an honest party that possesses this chain in round r and the corresponding target T is such that $f(T, n_r) \notin [\eta f, \theta f]$. Note that E_{r-1} is (η, θ)-good, and so, by Corollary 15, E_r is $\frac{m}{16\tau f}$-accurate. Let $r^* < r$ be the last $(\eta\gamma, \theta/\gamma)$-good recalculation point of $\mathcal{C}$ (let r^* be 0 in case there is no such point).

First suppose that there is another recalculation point $r' \in (r^*, r]$. By the definition of r^*, r' is not $(\eta\gamma, \theta/\gamma)$-good. However, the assumptions of Lemma 18 hold, implying that $\mathcal{C}$ is $(\eta\gamma, \theta/\gamma)$-good. We have reached a contradiction.

We may now assume that there is no recalculation point in $(r^*, r]$ and so the points r^* and r correspond to the same target T with $\eta\gamma \leq f(T, n_{r^*}) \leq \theta f/\gamma$. Note that since r^* is an $(\eta\gamma, \theta/\gamma)$-good recalculation point and E_{r-1} is (η, θ)-good, we have $r - r^* \leq \frac{\tau m}{f}$. This follows from Lemma 16, because $\mathcal{C}$ belongs to an honest party at round r. Thus, $\frac{1}{\gamma} n_{r^*} \leq n_r \leq \gamma n_{r^*}$, and so (by Fact 2) $f(T, n_r) \leq f(T, \gamma n_{r^*}) \leq \gamma f(T, n_{r^*}) \leq \theta f$ and $f(T, n_r) \geq f(T, \frac{1}{\gamma} n_{r^*}) \geq \frac{1}{\gamma} f(T, n_{r^*}) \geq \eta f$. □

Theorem 21. *A typical execution in a (γ, s)-respecting environment is $\frac{m}{16\tau f}$-accurate and (η, θ)-good.*

Proof. This follows from Corollaries 20 and 15. □

6.5 Common Prefix and Chain Quality

Proposition 22. *Let E be a typical execution in a (γ, s)-respecting environment. Any $\frac{\theta\gamma m}{8\tau}$ consecutive blocks in an epoch of a chain $\mathcal{C} \in \mathcal{S}_r$ have been computed in at least $\frac{m}{16\tau f}$ rounds.*

Proof. Suppose—towards a contradiction—that the blocks of $\mathcal{C}$ where computed during the rounds in S^*, for some S^* such that $|S^*| < \frac{m}{16\tau f}$. Consider an S such that $S^* \subseteq S$ and $|S| = \frac{m}{16\tau f}$ and the property that a block of target T in $\mathcal{C}$ was computed by an honest party in some round $v \in S$. Such an S exists by Lemmas 14 and 16. By Propositions 12 and 17, the number of blocks of target T computed in S is at most

$$(1+\epsilon)(2-\delta)pT\sum_{u\in S} n_u \leq (1+\epsilon)(2-\delta)pT\gamma n_v|S| \leq (1+\epsilon)(2-\delta)\gamma|S| \cdot \frac{\theta f}{1-\theta f} \leq \frac{\theta\gamma m}{8\tau}.$$

For the first inequality we used Fact 1, for the second Fact 5 and that round v is (η,θ)-good, and for the last one Requirement (R2). □

Let us say that two chains $\mathcal{C}$ and $\mathcal{C}'$ *diverge* before round r, if the timestamp of the last block on their common prefix is less than r.

Lemma 23. *Let E be a typical execution in a (γ, s)-respecting environment. Any $\mathcal{C}, \mathcal{C}' \in \mathcal{S}_r$ do not diverge before round $r - \frac{m}{16\tau f}$.*

Proof. Consider the last block on the common prefix of $\mathcal{C}$ and $\mathcal{C}'$ that was computed by an honest party and let r^* be the round on which it was computed (set $r^* = 0$ if no such block exists). Denote by $\mathcal{C}^*$ the common part of $\mathcal{C}$ and $\mathcal{C}'$ up to (and including) this block and let $d^* = \mathrm{diff}(\mathcal{C}^*)$ and $S = \{i : r^* < u < r\}$. We claim that

$$(1+\epsilon)(1-\delta)p\sum_{u\in S} n_u \geq \sum_{u\in S} Q_u. \tag{2}$$

In view of Proposition 17, it suffices to show that the difficulty which the adversary contributed to $\mathcal{C}$ and $\mathcal{C}'$ is at least the right-hand side of (2). The proof of this rests on the following observation.

Consider any block B extending a chain $\mathcal{C}_1$ that was computed by an honest party in a uniquely successful round $u \in S$. Consider also an arbitrary $d \in \mathbb{R}$ such that $\mathrm{diff}(\mathcal{C}_1) \leq d < \mathrm{diff}(\mathcal{C}_1 B)$. We are going to argue that if another chain of difficulty at least d exists, then the block that "contains" the point of difficulty d was computed by the adversary. More formally, suppose a chain $\mathcal{C}_2 B'$ exists such that $B' \neq B$ and $\mathrm{diff}(\mathcal{C}_2) \leq d < \mathrm{diff}(\mathcal{C}_2 B')$. We observe that B' was computed by the adversary. This is because no honest party would extend $\mathcal{C}_2$ at a round later than u since $\mathrm{diff}(\mathcal{C}_2) \leq d < \mathrm{diff}(\mathcal{C}_1 B)$; on the other hand, if an honest party computed B' at some round $u' < u$, then no honest party would have extended $\mathcal{C}_1$ at round u since $\mathrm{diff}(\mathcal{C}_1) \leq d < \mathrm{diff}(\mathcal{C}_2 B')$; finally, note that u is also ruled out since it was a uniquely successful round by assumption.

Returning to the proof of (2) note that, by the Chain-Growth Lemma 9, $\mathrm{diff}(\mathcal{C}')$ and $\mathrm{diff}(\mathcal{C})$ are at least $d^* + \sum_{u\in S} Q_u$. To show (2) it suffices to argue that for all $d \in (d^*, \sum_{u\in S} Q_u]$ there is always a B' as above that lies either on $\mathcal{C}$, or on $\mathcal{C}'$, or on their common prefix. But this is always possible since B cannot be both on $\mathcal{C}$ and $\mathcal{C}'$ (note that by the definition of r^*, B cannot be on their common prefix). To finish the proof note that (2) contradicts Proposition 12 for large enough S. □

Theorem 24 (Common-Prefix)**.** *Let E be a typical execution in a (γ, s)-respecting environment. For any round r and any two chains in $\mathcal{S}_r$, the common-prefix property holds for $k \geq \frac{\theta\gamma m}{4\tau}$.*

Proof. Suppose common-prefix fails for two chains $\mathcal{C}$ and $\mathcal{C}'$ at round r. At least $k/2$ of the blocks in each chain after their common prefix, lie in a single epoch. Proposition 22 implies that $\mathcal{C}$ and $\mathcal{C}'$ diverge before round $r - \frac{m}{16\tau f}$, contradicting Lemma 23. □

Theorem 25 (Chain-Quality)**.** *Let E be a typical execution in a (γ, s)-respecting environment. For the chain of any honest party at any round in E, the chain-quality property holds with parameters $\ell = \frac{m}{16\tau f}$ and $\mu = (1 + \delta/2)\lambda < (1 - \delta/2)$, where $\lambda = \max\{t_r/n_r\} < (1 - \delta)$.*

Proof. Let us denote by B_i the i-th block of $\mathcal{C}$ so that $\mathcal{C} = B_1 \dots B_{\text{len}(\mathcal{C})}$ and consider L consecutive blocks $B_u, \dots, B_v$. Define L' as the least number of consecutive blocks $B_{u'}, \dots, B_{v'}$ that include the L given ones (i.e., $u' \leq u$ and $v \leq v'$) and have the properties (1) that the block $B_{u'}$ was computed by an honest party or is B_1 in case such block does not exist, and (2) that there exists a round at which an honest party was trying to extend the chain ending at block $B_{v'}$. Observe that number L' is well defined since $B_{\text{len}(\mathcal{C})}$ is at the head of a chain that an honest party is trying to extend. Denote by d' the total difficulty of these L' blocks. Define also r_1 as the round that $B_{u'}$ was created (set $r_1 = 0$ if $B_{u'}$ is the genesis block), r_2 as the first round that an honest party attempts to extend $B_{v'}$, and let $S = \{r : r_1 \leq r \leq r_2\}$. Note that $|S| \geq \frac{m}{16\tau f}$.

Now let x denote the total difficulty of all the blocks from honest parties that are included in the L blocks and—towards a contradiction—assume that

$$x < \Big[1 - \Big(1 + \frac{\delta}{2}\Big)\lambda\Big]d \leq \Big[1 - \Big(1 + \frac{\delta}{2}\Big)\lambda\Big]d'. \tag{3}$$

Suppose first that all the L' blocks $\{B_j : u' \leq j \leq v'\}$ have been computed during the rounds in the set S. Recalling Proposition 17, we now argue the following sequence of inequalities.

$$(1 + \epsilon)(1 - \delta)p \sum_{u \in S} n_u \geq d' - x \geq \Big(1 + \frac{\delta}{2}\Big)\lambda d' \geq \Big(1 + \frac{\delta}{2}\Big)\lambda \sum_{u \in S} Q_u. \tag{4}$$

The first inequality follows from the definition of x and d' and Proposition 17. The second one comes from the relation between x and d' outlined in (3). To see the last inequality, assume $\sum_{u \in S} Q_u > d'$. But then, by the Chain-Growth Lemma 9, the assumption than an honest party is on $B_{v'}$ at round r_2 is contradicted as all honest parties should be at chains of greater length. We now observe that (4) contradicts Proposition 12, since

$$\Big(1 + \frac{\delta}{2}\Big)\lambda \sum_{u \in S} Q_u > (1 - \epsilon)(1 - \theta f)\Big(1 - \frac{\delta}{2}\Big)p \sum_{u \in S} n_u \geq (1 + \epsilon)(1 - \delta)p \sum_{u \in S} n_u,$$

where the middle inequality follows by Requirement (R2).

To finish the proof we need to consider the case in which these L' blocks contain blocks that the adversary computed in rounds outside S. It is not hard to see that this case implies either a prediction or an insertion and cannot occur in a typical execution. □

6.6 Persistence and Liveness

Theorem 26. *Let E be a typical execution in a (γ, s)-respecting environment. Persistence is satisfied with depth $k \geq \frac{\theta\gamma m}{4\tau}$.*

Proof. Suppose an honest party P has at round r a chain $\mathcal{C}$ such that $\mathcal{C}^{\lceil k}$ contains a transaction tx.

We first show that the $k \geq \frac{\theta\gamma m}{4\tau}$ blocks of $\mathcal{C}$ cannot have been computed in less than $\frac{m}{16\tau f}$ rounds. Suppose—towards a contradiction—that this was the case. By Lemma 16, at least $\frac{\theta\gamma m}{8\tau}$ of the k blocks belong to a single epoch and Proposition 22 is contradicted.

To show persistence, note that if any party $P' \neq P$ has a chain $\mathcal{C}'$ at round r and $\mathcal{C}^{\lceil k}$ is not a prefix of $\mathcal{C}'$, then Lemma 23 is contradicted. Next, let $r' > r$ be the first round after r such that

an honest party P' has a chain $\mathcal{C}'$ such that $\mathcal{C}^{\lceil k}$ is not a prefix of $\mathcal{C}'$. By the note above and the minimality of r' it follows that no honest party had a prefix of $\mathcal{C}'$ at round $r'-1$. Thus, $\mathcal{C}'$ existed at round $r'-1$ and P' had another chain $\mathcal{C}''$ at that round such that $\mathcal{C}^{\lceil k} \preceq \mathcal{C}''$ and $\mathrm{diff}(\mathcal{C}'') < \mathrm{diff}(\mathcal{C}')$. We now observe that $\mathcal{C}'$ and $\mathcal{C}''$ contradict Lemma 23 at round $r'-1$. □

Theorem 27. *Let E be a typical execution in a (γ, s)-respecting environment. Liveness is satisfied for depth k with wait-time $\frac{m}{16\tau f} + \frac{\gamma k}{\eta f(1-\epsilon)(1-\theta f)}$.*

Proof. Suppose a transaction tx is included in any block computed by an honest party for $\frac{m}{16\tau f}$ consecutive rounds and let S denote the set of $\frac{\gamma k}{\eta f(1-\epsilon)(1-\theta f)}$ rounds that follow these rounds. Consider now the chain $\mathcal{C}$ of an arbitrary honest party after the rounds in S. By Lemma 14, $\mathcal{C}$ contains an honest block computed in the $\frac{m}{16\tau f}$ rounds. This block contains tx. Furthermore, after the rounds in the set S, on top of this block there has been accumulated at least $\sum_{r\in S} Q_r$ amount of difficulty. We claim that this much difficulty corresponds to at least k blocks. To show this, assume $|S| \leq s$ (or consider only the first s rounds of S). Let T be the smallest target computed by an honest party during the rounds in S and let u be such a round. It suffices to show $T\sum_{r\in S} Q_r \geq k$. Indeed,

$$T\sum_{r\in S} Q_r \geq (1-\epsilon)(1-\theta f)pT\sum_{r\in S} n_r \geq (1-\epsilon)(1-\theta f)\frac{pTn_u|S|}{\gamma} \geq k.$$

The first inequality follows from Proposition 12, the second by Fact 1, and for the last one we substitute the size of S and use that $pTn_u \geq f(T, n_u) \geq \eta f$ (since u is (η, θ)-good). □

References

[Bac97] Adam Back. Hashcash. http://www.cypherspace.org/hashcash, 1997.

[Bah13] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *IACR Cryptology ePrint Archive*, 2013:868, 2013.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[ES14] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

[GKL14] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. *IACR Cryptology ePrint Archive*, 2014:765, 2014.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[HT94] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical report, 1994.

[JB99] Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*. The Internet Society, 1999.

[KKKT16] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In Vincent Conitzer, Dirk Bergemann, and Yiling Chen, editors, *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382. ACM, 2016.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. *IACR Cryptology ePrint Archive*, 2015:1019, 2015.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[McD98] Colin McDiarmid. *Probabilistic Methods for Algorithmic Discrete Mathematics*, chapter Concentration, pages 195–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[Nak09] Satoshi Nakamoto. Bitcoin open source implementation of p2p currency. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source, February 2009.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[PSS16] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. *IACR Cryptology ePrint Archive*, 2016:454, 2016.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.

[SSZ15] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *CoRR*, abs/1507.06183, 2015.

A Model and Definitions (cont'd)

As mentioned in Section 2, we describe our protocols in a model that extends the synchronous communication network model presented in [GKL14, GKL15] for the analysis of the Bitcoin backbone protocol in the static setting with a fixed number of parties (which in turn is based on Canetti's formulation of "real world" notion of protocol execution [Can00a, Can00b, Can01] for multi-party protocols) to the dynamic setting with a varying number of parties.

Round structure and protocol execution. As in [GKL14], the protocol execution proceeds in rounds with inputs provided by an environment program denoted by $\mathcal{Z}$ to parties that execute the protocol Π, and our adversarial model in the network is "adaptive," meaning that the adversary ($\mathcal{A}$) is allowed to take control of parties on the fly, and "rushing," meaning that in any given round the adversary gets to see all honest players' messages before deciding his strategy. The parties' access to the hash function and their communication mechanism are captured by a joint random oracle /

diffusion functionality which reflects Bitcoin's peer structure. The diffusion functionality, [GKL14], allows the order of messages to be controlled by $\mathcal{A}$, i.e., there is no atomicity guarantees in message broadcast [HT94], and, furthermore, the adversary is allowed to spoof the source information on every message (i.e., communication is not authenticated). Still, the adversary cannot change the contents of the messages nor prevent them from being delivered. We will use DIFFUSE as the message transmission command that captures this "send-to-all" functionality.

The parties that *may* become active in a protocol execution are encoded as part of a control program C and come from a universe $\mathcal{U}$ of parties.

The protocol execution is driven by an environment program $\mathcal{Z}$ that interacts with other instances of programs that it spawns at the discretion of the control program C. The pair $(\mathcal{Z}, C)$ forms of a *system of interactive Turing machines* (ITM's) in the sense of [Can00b]. The execution is with respect to a program Π, an adversary $\mathcal{A}$ (which is another ITM) and the universe of parties $\mathcal{U}$. Assuming the control program C allows it, the environment $\mathcal{Z}$ can activate a party by writing to its input tape. Note that the environment $\mathcal{Z}$ also receives the parties' outputs when they are produced in a standard subroutine-like interaction. Additionally, the control program maintains a flag for each instance of an ITM, (abbreviated as ITI in the terminology of [Can00b]), that is called the `ready` flag and is initially set to false for all parties.

The environment $\mathcal{Z}$, initially is restricted by C to spawn the adversary $\mathcal{A}$. Each time the adversary is activated, it may send one or more messages of the form $(\mathsf{Corrupt}, P_i)$ to C and C will mark the corresponding party as corrupted.

Functionalities available to the protocol. The ITI's of protocol Π will have access to a joint ideal functionality capturing the random oracle and the diffusion mechanism which is defined in a similar way as [GKL14] and is explained below.

- The random oracle functionality. Given a query with a value x marked for "calculation" for the function $H(\cdot)$ from a honest party P_i and assuming x has not been queried before, the functionality returns a value y which is selected at random from $\{0,1\}^\kappa$; furthermore, it stores the pair (x, y) in the table of $H(\cdot)$, in case the same value x is queried in the future. Each honest party P_i is allowed to ask q queries in each round as determined by the diffusion functionality (see below). On the other hand, each honest party is given unlimited queries for "verification" for the function $H(\cdot)$. The adversary $\mathcal{A}$, on the other hand, is given a bounded number queries in each round as determined by diffusion functionality with a bound that is initialized to 0 and determined as follows: whenever a corrupted party is activated, the party can ask the bound to be increased by q; each time a query is asked by the adversary the bound is decreased by 1. No verification queries are provided to $\mathcal{A}$. Note that the value q is a polynomial function of κ, the security parameter. The functionality can maintain tables for functions other than $H(\cdot)$ but, by convention, the functionality will impose query quotas to function $H(\cdot)$ only.

- The diffusion functionality. This functionality keeps track of rounds in the protocol execution; for this purpose it initially sets a variable *round* to be 1. It also maintains a RECEIVE() string defined for each party P_i in $\mathcal{U}$. A party that is activated is allowed to query the functionality and fetch the contents of its personal RECEIVE() string. Moreover, when the functionality receives a message $(\mathsf{Diffuse}, m)$ from party P_i it records the message m. A party P_i can signal when it is complete for the round by sending a special message ($\mathsf{RoundComplete}$). With respect to the adversary $\mathcal{A}$, the functionality allows it to receive the contents of all contents sent in $\mathsf{Diffuse}$ messages for the round and specify the contents of the RECEIVE() string for each party P_i. The adversary has to specify when it is complete for the current round. When all parties

are complete for the current round, the functionality inspects the contents of all RECEIVE() strings and includes any messages m that were diffused by the parties in the current round but not contributed by the adversary to the RECEIVE() tapes (in this way guaranteeing message delivery). It also flushes any old messages that were diffused in previous rounds and not diffused again. The variable *round* is then incremented.

The dynamic q-bounded synchronous setting. Consider $\mathbf{n} = \{n_r\}_{r\in\mathbb{N}}$ and $\mathbf{t} = \{t_r\}_{r\in\mathbb{N}}$ two series of natural numbers. As mentioned, the first instance that is spawned by $\mathcal{Z}$ is the adversary $\mathcal{A}$. Subsequently the environment may spawn (or activate if they are already spawned) parties $P_i \in \mathcal{U}$. The control program maintains a counter in each sequence of activations and matches it with the current round that is maintained by the diffusion functionality. Each time a party diffuses a message containing the label "`ready`" the control program C increases the ready counter for the round. In round r, the control program C will enable the adversary $\mathcal{A}$ to complete the round, only provided that (i) exactly n_r parties have transmitted `ready` message, (ii) the number of ("corrupt") parties controlled by $\mathcal{A}$ should match t_r.

Parties, when activated, are able to read their input tape INPUT() and communication tape RECEIVE() from the diffusion functionality. Observe that parties are unaware of the set of activated parties. The Bitcoin backbone protocol requires from parties (miners) to solve a "proof of work" (POW, aka "cryptographic puzzle"— see, e.g., [DN92, RSW96, Bac97, JB99]), which, as mentioned earlier, essentially amounts to brute-forcing a hash inequality based on SHA-256, in order to generate new blocks for the blockchain. This is modeled in [GKL15] as parties having access to the oracle $H(\cdot)$. The fact that (active) parties have limited ability to produce such POWs, is captured as in [GKL15] by the random oracle functionality and the fact that it paces parties to query a limited number of queries *per round*. The bound, q, is a function of the security parameter κ; in this sense the parties may be called q-bounded[6]. We refer to the above restrictions on the environment, the parties and the adversary as the *dynamic q-bounded synchronous setting*.

The term $\{\text{VIEW}^{P,\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ denotes the random variable ensemble describing the view of party P after the completion of an execution running protocol Π with environment $\mathcal{Z}$ and adversary $\mathcal{A}$, on input $z \in \{0,1\}^*$. We will only consider a "standalone" execution without any auxiliary information and we will thus restrict ourselves to executions with $z = 1^\kappa$. For this reason we will simply refer to the ensemble by $\text{VIEW}^{P,\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$. The concatenation of the view of all parties ever activated in the execution is denoted by $\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$.

Properties of protocols. In our theorems we will be concerned with *properties* of protocols Π running in the above setting. Such properties will be defined as predicates over the random variable $\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}}$ by quantifying over all possible adversaries $\mathcal{A}$ and environments $\mathcal{Z}$. 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., $k = 6$).

The protocol class that we will analyze will not be able to preserve its properties for arbitrary sequences of parties. To restrict the way the sequence $\mathbf{n}$ is fluctuating we will introduce the following class of sequences.

Definition 28. For $\gamma \in \mathbb{R}^+$, we call a sequence $(n_r)_{r\in\mathbb{N}}$ (γ, s)-*respecting* if, it holds that in a sequence of rounds S with $|S| \leq s$ rounds it holds that $\max_{r\in S} n_r \leq \gamma \cdot \min_{r\in S} n_r$.

[6] In [GKL15] this is referred to as the "flat-model" in terms of computational power, where all parties are assumed equal. In practice, different parties may have different "hashing power"; note that this does not sacrifice generality since one can imagine that real parties are simply clusters of some arbitrary number of flat-model parties.

Observe that the above sequence is fairly general and also can capture exponential growth by setting e.g., $\gamma = 2$ and $s = 10$, it follows that every 10 rounds the number of ready parties may double. Note that this will not lead to an exponential running time overall since the total run time is bounded by a polynomial in κ, (due to the fact that $(\mathcal{Z}, C)$ is a system of ITM's, $\mathcal{Z}$ is locally polynomial bounded, C is a polynomial-time program, and thus [Can00b, Proposition 3] applies).

More formally, a protocol Π would satisfy a property Q for a certain class of sequences $\mathbf{n}, \mathbf{t}$, provided that for all PPT $\mathcal{A}$ and locally polynomial bounded $\mathcal{Z}$, it holds that $Q(\text{VIEW}^{\mathbf{t},\mathbf{n}}_{\Pi,\mathcal{A},\mathcal{Z}})$ is true with overwhelming probability of the coins of $\mathcal{A}, \mathcal{Z}$ and the random oracle functionality.

In this paper, we will be interested in (γ, s)-respecting sequences $\mathbf{n}$, sequences $\mathbf{t}$ suitably restricted by $\mathbf{n}$, and protocols Π that are suitably parameterized given $\mathbf{n}, \mathbf{t}$.

B The Bitcoin Backbone Protocol with Variable Difficulty (cont'd)

In this section we give a a more detailed description of the Bitcoin backbone protocol with chains of variable difficulty. The presentation is based on the description in [GKL15].

B.1 The protocol

As in [GKL15] in our description oof the backbone protocol we intentionally avoid specifying the type of values/content 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. These checks and operations are handled by the external functions $V(\cdot), I(\cdot)$ and $R(\cdot)$ (the *content validation function*, the *input contribution function* and the *chain reading function*, resp.) which are specified by the application that runs "on top" of the backbone protocol.

The Bitcoin backbone protocol in the dynamic setting is specified as Algorithm 4 and depends on three sub-procedures.

Chain validation. The validate algorithm performs a validation of the structural properties of a given chain $\mathcal{C}$. It is given as input the value q, as well as hash functions $H(\cdot), G(\cdot)$. It is parameterized by the content validation predicate predicate $V(\cdot)$ as well as by $D(\cdot)$, the *target calculation function* (see Section 3). For each block of the chain, the algorithm checks that the proof of work is properly solved (with a target that is suitable as determined by the target calculation function), and that the counter *ctr* does not exceed q. Furthermore it collects the inputs from all blocks, $\mathbf{x}_{\mathcal{C}}$, and tests them via the predicate $V(\mathbf{x}_{\mathcal{C}})$. Chains that fail these validation procedure are rejected. (Algorithm 1.)

Algorithm 1 The *chain validation predicate*, parameterized by q, D, the hash functions $G(\cdot), H(\cdot)$, and the *input validation predicate* $V(\cdot)$. The input is chain $\mathcal{C}$.

function $\mathsf{validate}(r_{\mathsf{now}}, \mathcal{C})$
  $valid \leftarrow V(\mathbf{x}_{\mathcal{C}}) \wedge (C \neq \varepsilon)$
  if $valid =$ **true then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
    $r' \leftarrow r_{\mathsf{now}}$
    $\langle r, st, x, ctr\rangle \leftarrow \text{head}(\mathcal{C})$
    $st' \leftarrow H(ctr, G(r, st, x))$
    repeat
      $\langle r, st, x, ctr\rangle \leftarrow \text{head}(\mathcal{C})$
      $T \leftarrow D(\mathbf{r}_{\mathcal{C}^{\lceil 1}})$ ▷ Calculate target based on $\mathcal{C}^{\lceil 1}$
      if $\mathsf{validblock}_q^T(\langle st, x, ctr\rangle) \wedge (H(ctr, G(r, st, x)) = st') \wedge (r < r')$ **then**
        $r' \leftarrow r$ ▷ Retain round timestamp
        $st' \leftarrow st$ ▷ Retain hash value
        $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove the head from $\mathcal{C}$
      else
        $valid \leftarrow$ False
      end if
    until $(\mathcal{C} = \varepsilon) \vee (valid =$ False$)$
  end if
  return $valid$
end function

Chain comparison. The objective of the second algorithm, called $\mathsf{maxvalid}$, is to find the "best possible" chain when given a set of chains. The algorithm is straightforward and is parameterized by a $\mathsf{max}(\cdot)$ function that applies some ordering in the space of chains. The most important aspect is the chains' difficulty in which case $\mathsf{max}(\mathcal{C}_1, \mathcal{C}_2)$ will return the most *difficult* of the two. In case $\text{diff}(\mathcal{C}_1) = \text{diff}(\mathcal{C}_2)$, some other characteristic can be used to break the tie. In our case, $\mathsf{max}(\cdot, \cdot)$ will always return the first operand to reflect the fact that parties adopt the first chain they obtain from the network. (Algorithm 2.)

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

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

Proof of work. The third algorithm, called pow, is the proof of work-finding procedure. It takes as input a chain and attempts to extend it via solving a proof of work. This algorithm is parameterized by two hash functions $H(\cdot), G(\cdot)$ as well as the parameter q. Moreover, the algorithm calls the target

calculation function $D(\cdot)$ om prder to determine the value T that will be used for the proof of work. The procedure, given a chain $\mathcal{C}$ and a value x to be inserted in the chain, hashes these values to obtain h and initializes a counter ctr. Subsequently, it increments ctr and checks to see whether $H(ctr, h) < T$; in case a suitable ctr is found then the algorithm succeeds in solving the POW and extends chain $\mathcal{C}$ by one block. (Algorithm 3.)

Algorithm 3 The *proof of work* function, parameterized by q and hash functions $H(\cdot), G(\cdot)$. The input is $(x, \mathcal{C})$.

function $\mathsf{pow}(r, x, \mathcal{C})$
  if $\mathcal{C} = \varepsilon$ **then** ▷ Determine proof of work instance.
    $st \leftarrow 0$
  else
    $\langle r', st', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$
    $st \leftarrow H(ctr', G(r', st', x'))$
  end if
  $ctr \leftarrow 1$
  $B \leftarrow \varepsilon$
  $T \leftarrow D(\mathbf{r}_{\mathcal{C}})$ ▷ Calculate target for next block based on timestamps.
  $h \leftarrow G(r, st, x)$
  while $(ctr \leq q)$ **do**
    if $(H(ctr, h) < T)$ **then** ▷ Proof of work succeeds and a new block is created.
      $B \leftarrow \langle r, st, x, ctr \rangle$
      break
    end if
    $ctr \leftarrow ctr + 1$
  end while
  $\mathcal{C} \leftarrow \mathcal{C}B$ ▷ Chain is extended
  return $\mathcal{C}$
end function

The backbone protocol. The core of the protocol is similar to that of [GKL15], with several important distinctions. First is the procedure to follow when they become active. Parties check the `ready` flag they possess that is false if and only if they have been inactive in the previous round. In case the `ready` flag is false, they broadcast a special message '**Join**' to request the most recent version of the blockchain(s). Similarly, parties that receive the special request message in their Receive() tape they broadcast their chain. As before parties, run "indefinitely" (our security analysis will apply when the total running time is polynomial in κ). The input contribution function $I(\cdot)$ and the chain reading function $R(\cdot)$ are applied to the values stored in the chain. Parties check their communication tape Receive() to see whether any necessary update of their local chain is due; then they attempt to extend it via the POW algorithm pow. The function $I(\cdot)$ determines the input to be added in the chain given the party's state st, the current chain $\mathcal{C}$, the contents of the party's input tape Input() and communication tape Receive(). The input tape contains two types of symbols, Read and (Insert, *value*); other inputs are ignored. In case the local chain $\mathcal{C}$ is extended the new chain is broadcast to the other parties. Finally, in case a Read symbol is present in the communication tape, the protocol applies function $R(\cdot)$ to its current chain and writes the result onto the output tape Output(). The pseudocode of the backbone protocol is presented in Algorithm 4.

Algorithm 4 The Bitcoin backbone protocol in the dynamic setting at round "`round`" on local state $(st, \mathcal{C})$ parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$. The `ready` flag is **false** if and only if the party was inactive in the previous round.

if `ready` = **true then**
  DIFFUSE('`ready`')
  $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{all chains } \mathcal{C}' \text{ found in } \text{RECEIVE}())$
  $\langle st, x\rangle \leftarrow I(st, \tilde{\mathcal{C}}, \texttt{round}, \text{INPUT}(), \text{RECEIVE}())$
  $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(\texttt{round}, x, \tilde{\mathcal{C}})$
  if $(\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}) \vee$ ('**Join**' $\in$ RECEIVE()) **then**
    $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
    DIFFUSE($\mathcal{C}$) ▷ chain is diffused when it is updated or when someone wants to join.
  end if
  if INPUT() contains READ **then**
    write $R(\mathbf{x}_{\mathcal{C}})$ to OUTPUT()
    DIFFUSE(RoundComplete)
  end if
else
  `ready` $\leftarrow$ **true**
  DIFFUSE(Join, RoundComplete)
end if

C Robust Public Transaction Ledgers

In this section we reproduce the presentation of public transaction ledgers given in [GKL14, GKL15]. A *public transaction ledger* is defined with respect to a set of valid ledgers $\mathcal{L}$ and a set of valid transactions $\mathcal{T}$, each one possessing an efficient membership test. A ledger $\mathbf{x} \in \mathcal{L}$ is a vector of sequences of transactions $\text{tx} \in \mathcal{T}$. Each transaction tx may be associated with one or more *accounts*, denoted $a_1, a_2, \ldots$ etc.

The backbone protocol parties, called *miners* in the context of this section, process sequences of transactions of the form $x = \text{tx}_1 \ldots \text{tx}_e$ that are supposed to be incorporated into their local chain $\mathcal{C}$. The input inserted at each block of the chain $\mathcal{C}$ is the sequence x of transactions. Thus, a ledger is a vector of transaction sequences $\langle x_1, \ldots, x_m\rangle$, and a chain $\mathcal{C}$ of length m contains the ledger $\mathbf{x}_{\mathcal{C}} = \langle x_1, \ldots, x_m\rangle$ if the input of the j-th block in $\mathcal{C}$ is x_j.

The description and properties of the ledger protocol will be expressed relative to an oracle Txgen which will control a set of accounts by creating them and issuing transactions on their behalf. In an execution of the backbone protocol, the environment $\mathcal{Z}$ as well as the miners will have access to Txgen. Specifically, Txgen is a stateful oracle that responds to two types of queries (which we purposely only describe at a high level):

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

We also consider a symmetric relation on $\mathcal{T}$, denoted by $C(\cdot,\cdot)$, which indicates when two transactions tx_1, tx_2 are conflicting. Valid ledgers $\mathbf{x} \in \mathcal{L}$ can never contain two conflicting transactions. We call oracle Txgen *unambiguous* if it holds that for all PPT $\mathcal{A}$, the probability that $\mathcal{A}^{\mathsf{Txgen}}$ produces a transaction tx' such that $C(\text{tx}', \text{tx}) = 1$, for tx issued by Txgen, is negligible in κ.

Finally, a transaction tx is called *neutral* if $C(\text{tx}, \text{tx}') = 0$ for any other transaction tx'. The presence of neutral transactions in the ledger can be helpful for a variety of purposes, as we will see next and in the BA protocol that we build on top of the ledger. For convenience we will assume that a single random nonce $\rho \in \{0,1\}^\kappa$ is also a valid transaction. Nonces will be neutral transactions and may be included in the ledger for the sole purpose of ensuring independence between the POW instances solved by the honest parties.

Next, we determine the three functions $V(\cdot), I(\cdot), R(\cdot)$ that will turn the backbone protocol into Π_{PL}, a protocol realizing a public transaction ledger. See Figure 1.

Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_m\rangle)$ is true if and only if the vector $\langle x_1, \ldots, x_m\rangle$ is a valid ledger, i.e., $\langle x_1, \ldots, x_m\rangle \in \mathcal{L}$.
Chain reading function $R(\cdot)$	If $V(\langle x_1, \ldots, x_m\rangle) =$ True, the value $R(\mathcal{C})$ is equal to $\langle x_1, \ldots, x_m\rangle$; undefined otherwise.
Input contribution function $I(\cdot)$	$I(st, \mathcal{C}, round, \text{INPUT}())$ operates as follows: if the input tape contains (INSERT, v), it parses v as a sequence of transactions and retains the largest subsequence $x' \preceq v$ that is valid with respect to $\mathbf{x}_\mathcal{C}$ (and whose transactions are not already included in $\mathbf{x}_\mathcal{C}$). Finally, $x = \text{tx}_0 x'$ where tx_0 is a neutral random nonce transaction.

Figure 1: *The public transaction ledger protocol Π_{PL}, built on the Bitcoin backbone.*

In Section 4.3 we introduced two essential properties for a protocol maintaning a public transaction ledger: (i) *Persistence* and (ii) *Liveness*. In a nutshell, Persistence states that once an honest player reports a transaction "deep enough" in the ledger, then all other honest players will report it indefinitely whenever they are asked, and at exactly the same position in the ledger (essentially, this means that all honest players agree on all the transactions that took place and in what order). In a more concrete Bitcoin-like setting, Persistence is essential to ensure that credits are final and that they happened at a certain "time" in the system's timeline (which is implicitly defined by the ledger itself).

Persistence is useful but not enough to ensure that the ledger makes progress, i.e., that transactions are eventually inserted in a chain. This is captured by the Liveness property, which states that as long as a transaction comes from an honest account holder and is provided by the environment to all honest players, then it will be inserted into the honest players' ledgers, assuming the environment keeps providing it as an input for a sufficient number of rounds.

For more details about the specification of a robust transaction ledger, in particular Bitcoin-like transactions and ledger, refer to [GKL14, GKL15].

D Martingale Sequences and Other Mathematical Facts

Definition 29. [MU05, Chapter 12] A sequence of random variables $X_0, X_1, \ldots$ is a martingale with respect to the sequence $Y_0, Y_1, \ldots$, if, for all $n \geq 0$, (1) X_n is a function of $Y_0, \ldots, Y_n$, (2) $\mathbf{E}[|X_n|] < \infty$, and (3) $\mathbf{E}[X_{n+1}|Y_0, \ldots, Y_n] = X_n$.

Theorem 30. *[McD98, Theorem 3.15] Let $X_0, X_1, \ldots$ be a martingale with respect to the sequence $Y_0, Y_1, \ldots$. For $n \geq 0$, let*

$$V = \sum_{i=1}^{n} \text{var}(X_i - X_{i-1}|Y_0, \ldots, Y_{i-1}) \quad \textit{and} \quad b = \max_{1 \leq i \leq n} \sup(X_i - X_{i-1}|Y_0, \ldots, Y_{i-1}),$$

where sup is taken over all possible assignments to $Y_0, \ldots, Y_{i-1}$. *Then, for any* $t, v \geq 0$,

$$\mathbf{Pr}\big[(X_n \geq X_0 + t) \wedge (V \leq v)\big] \leq \exp\left\{-\frac{t^2}{2v + 2bt/3}\right\}.$$

Fact 2. *Suppose* $f : [0, \infty) \to [0, \infty)$ *is concave and* $f(0) \geq 0$. *Then, for any* $x, y \in [0, \infty)$ *and* $\lambda \in [1, \infty)$,

$$f(x/\lambda) \geq f(x)/\lambda, \quad f(\lambda x) \leq \lambda f(x), \quad f(x + y) \leq f(x) + f(y).$$

The following well-known e-related inequalities are used throughout the analysis, possibly without reference.

Fact 3. *(1)* $1 + x < e^x$, *for all* x. *(2)* $-\frac{x}{1-x} < \ln(1 - x)$, *for* $x \in (0, 1)$. *(3)* $\frac{x}{1+x/2} < \ln(1 + x) < x$, *for* $x > 0$.