How Does Nakamoto Set His Clock? Full Analysis of Nakamoto Consensus in Bounded Delay Networks

Juan Garay[1], Aggelos Kiayias[2], and Nikos Leonardos[3]

[1] Texas A&M University, garay@tamu.edu
[2] University of Edinburgh & IOHK, akiayias@inf.ed.ac.uk
[3] National and Kapodistrian University of Athens, nikos.leonardos@gmail.com

Abstract. Nakamoto consensus, arguably the most exciting development in distributed computing in the last few years, is in a sense a recasting of the traditional state-machine-replication problem in an unauthenticated setting, where furthermore parties come and go without warning. The protocol relies on a cryptographic primitive known as *proof of work* (PoW) which is used to throttle message passing. Importantly, the PoW difficulty level is appropriately adjusted throughout the course of the protocol execution relying on the blockchain's timekeeping ability.

While the original formulation was only accompanied by rudimentary analysis, significant and steady progress has been made in abstracting the protocol's properties and providing a formal analysis under various restrictions and protocol simplifications. Still, a full analysis of the protocol that includes its target recalculation and, notably, the *timestamp adjustment mechanism*—specifically, the protocol allows incoming block timestamps in the near future, as determined by a protocol parameter, and rejects blocks that have a timestamp in the past of the median time of a specific number of blocks on-chain (namely, 11)— which equip it to operate in its intended setting of bounded communication delays, imperfect clocks and dynamic participation, has remained open.

The gap is that Nakamoto's protocol fundamentally depends on the blockchain itself to be a consistent timekeeper that should advance roughly on par with real time. In order to tackle this question we introduce a new analytical tool that we call *hot-hand executions*, which capture the regular occurrence of high concentration of honestly generated blocks, and correspondingly put forth and prove a new blockchain property called *concentrated chain quality*, which may be of independent interest. Utilizing these tools and techniques we demonstrate that Nakamoto's protocol achieves, under suitable conditions, safety, liveness as well as (consistent) timekeeping.

1 Introduction

Nakamoto's blockchain protocol [24] is a consensus protocol where parties engage in the collection and organization of transactions in a ledger without having any information about each other or even precise knowledge of the number of parties running the protocol at any given time. This is in contrast to classical models and results in consensus (aka Byzantine agreement) [26,21] and other fundamental distributed computing tasks, where it is typically assumed that parties have pairwise authenticated communication channels or are initialized with the public keys of all participants. Instead, Nakamoto's blockchain protocol relies on the cryptographic primitive known as *proof of work* (PoW, aka cryptographic puzzles) [10,2,17]), to throttle message transmission and stochastically create opportunities for unifying the parties' possibly diverging views, despite the presence of a subset of them acting adversarially.

Given that the original protocol was presented with only a rudimentary analysis focusing solely on the application context of fund transfers, a number of works have attempted to isolate the protocol's properties and provide a formal analysis. The first analysis, presented in [12], focused on a synchronous execution model, and assuming the probability of the parties to solve a PoW over a single message-passing round is suitably restricted, proved that the protocol satisfies consistency and liveness as long as the total computational power in the system is in favor of the honest parties. Two limitations of this first analysis were that the target recalculation mechanism of the blockchain protocol which adjusts the hardness of PoWs was excluded, and that the execution model considered synchronous communication rounds.

Addressing the latter problem was undertaken in [25] (with further improvements in follow-up works [20,27]), where the blockchain protocol was analyzed in the so-called *bounded-delay model* (cf. [9])[4], showing the protocol secure for a favorable range of choices of network delay Δ with respect to its hard-coded PoW hardness parameter, and its insecurity in the general case where Δ is chosen adversarially. Technically, the main challenge to address in transitioning to the bounded-delay model is that the usefulness of a certain event in the protocol execution (e.g., the creation of a PoW at time t) is affected by events that are happening at times up to $t + \Delta$ forward in time (e.g., the creation of another PoW) and hence this dependence asks for additional care in the probabilistic analysis.

The problem of analyzing the target recalculation mechanism was addressed in [13], albeit again in the synchronous communication model, by introducing a setting where parties' participation is allowed to change round by round following a predetermined schedule that has a bounded rate of change. The main technical difficulty addressed in that setting was the fact that PoW successes are not independent events in the execution since the difficulty of the PoW primitive is determined by preceding execution events instead of being fixed throughout, as in [12,25,20,27]. Aside of this, given the synchronous formulation, the timekeeping adjustment features of the protocol were abstracted out.

While the above works have significantly improved our understanding regarding the behavior of Nakamoto consensus in successively more refined theoretical models, the full analysis of the actual protocol has remained elusive. Specifically, all previous works analyzed simplified versions of the protocol removing or adjusting protocol elements that deal with bounded delay networks and fluctuating participation. For example, all previous works ignore the way the protocol adjusts local (software) clocks based on on-chain timestamps and the actual timestamp rules for incoming blocks, mechanisms that Nakamoto (presumably) included to deal with the fact that no protocol can realistically assume perfectly synchronized clocks in an imperfect network. In fact, manipulating timestamps and exploiting the protocol's timestamp validation mechanisms are a well known attack vector in the Bitcoin community[5], so removing such adversarial capabilities from consideration in the formal threat model is an important deficiency.

Thus, the question remained whether Nakamoto's proposed blockchain protocol—*with all its adjustment mechanisms included*—retains its properties in a bounded-delay network. Importantly, we want to answer this question when the number of parties is dynamically changing without following a predetermined schedule, i.e., it is *adaptively* selected by the adversary, possibly even reacting to events that happen during the protocol execution, as long as the rate of change is bounded by a constant.

Our results. Our main result is the proof that Nakamoto's protocol achieves consistency, liveness, as well as a new *timekeeping* property in bounded-delay networks with adaptive dynamic participation. Importantly, our work for the first time takes into account the way that Nakamoto's protocol adjusts local clocks using on-chain timestamps and validates incoming block timestamps. Specifically, the protocol allows incoming block timestamps in the near future, as determined by a protocol parameter, and rejects blocks that have a timestamp in the past of the *median time* of a specific number of blocks on-chain—also a protocol parameter. These mechanisms open up an array of attack vectors that now must be considered in the security proof.

We observe that the "chain quality" property, proven in [12,25,13], is too weak to prove that the above clock adjustment mechanism is consistent with real time. To address this we introduce *concentrated chain quality* (CCQ), a property positing that honest parties' chains should include high concentrations of honest blocks at regular intervals. As we will see, CCQ will be the essential element to demonstrate that Nakamoto's design is a consistent timekeeper.

From a technical perspective, at the core of our analysis is the function $f(T, n)$, which determines the probability that n parties executing the protocol at a certain time find a PoW whose difficulty is determined by the "target" T (here, without loss of generality, n equates the number of parties with the number of CPUs of equal power running the protocol). Given that n is unknown and continuously changing, Nakamoto's pro-

[4] In this model, formulated by Dwork, Lynch and Stockmeyer [9], there is an upper bound Δ, unknown to the protocol, in the delay that the adversary may inflict on the delivery of messages.

[5] E.g., timejacking and poison-pill attacks or difficulty-raising by timestamp manipulation; see `https://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html`, and `https://bitcointalk.org/index.php?topic=43692.msg521772#msg521772`.

tocol adjusts T at regular intervals called *epochs* using the timestamps recorded on chain. As we show, the protocol's resilience to attacks will stem from its ability to keep $f(T, n)$ close to a suitable value that is favorably positioned with respect to the, otherwise unknown, network delay Δ. Starting with the assumption that the protocol is initiated at an appropriate f value, the blockchain protocol will recalculate T to approximate that initial value by estimating the number of active parties n per epoch. The estimation is based on the observed production of PoWs as recorded in the blockchain itself and the relative timings of their production. A complication here is that timestamps may be manipulated in various ways during the protocol execution and this is something that the analysis should take into account.

Thus, the first major technical challenge is the *analysis of the clock adjustment mechanism* and validation of incoming block timestamps, and the rule for determining the timestamp for the next block (namely, that the new time stamp must be greater than the *median* of a certain number of blocks—see Section 2 and eq. (6) in Section 4. The correctness of the rule critically depends on the existence on protocol executions of "winning streaks" by honest parties. Accordingly, we call such executions *hot hand* [6], and demonstrate how they result in regular high concentrations of honestly contributed blocks, which is necessary to ensure that honest parties's chains have timestamps that move forward. Our analysis also reveals a property of the Bitcoin blockchain that was before not formally understood and can be seen as a strengthening of chain quality (CQ) [12]: honest chains are not only guaranteed to have regular contributions by honest parties as CQ dictates, but these contributions, with overwhelming probability, will come in clusters, so that, regularly, it will happen that in a sequence of $2k_{\text{med}} - 1$ consecutive blocks at least k_{med} blocks will be contributed by the honest parties, where k_{med} is a small constant. This is our concentrated chain quality property. Prior to our work such statements could only be proved for large values of k_{med} bounding the adversary by $1/3$ of the total computational power, and hence they were unsuitable for arguing the security of Nakamoto's parameterization which imposes a lower bound on a block timestamp by the median of the last $2k_{\text{med}}-1 = 11$ blocks and seeks to argue protocol security for an adversarial bound below $1/2$.

The second major technical challenge is to work in a probabilistic setting where the random variables corresponding to the *cumulative difficulty* of PoWs (rather than their number) collected by the protocol participants capture the adaptive dynamic evolution of participants as well as the fact that some of these variables' values at a certain round may be affected by events in the future. The latter issue asks for a lower bound estimation of the aggregate difficulty (in terms of PoWs of different targets) collected by the honest parties over a period of time that is "isolated" from any future PoW event for a period of Δ rounds. At the same time, we need an upper bound on the aggregate difficulty amassed by adversarial parties while accounting for the fact that the adversary may choose to work on very difficult PoW instances for which it will be impossible to control their stochastic advantage via concentration bounds, due to high variance. We address this by introducing a suitable concept of "typical" execution where concentration bounds can be meaningfully applied to the relevant random variables.

Putting together the properties of hot-hand and typical executions we distill a **set of conditions**, stated here at a high level (refer to Section 2 for the detailed description), under which consistency and liveness of Nakamoto consensus can be shown. First, we need an honest majority, i.e., the honest parties' hashing power exceeds the adversarial parties' power. Second, we need epochs to be long enough to be able to adequately measure the change in the hashing power so it is reflected in the target recalculation (refer to C1 in Section 2). Then we need an upper bound on the network delay in terms of the block production rate and the other parameters (C2), as well as a bound on the rate the parties dynamically change over time (C3). Finally, we also prove the timekeeping property of Nakamoto's ledger under these same conditions, namely, that the protocol records time in a consistent manner and can export timing information in the ledger.

Remark. One might wonder why the number of parties is allowed to dynamically (and adaptively) evolve following even an exponential increase while the upper bound on the delay Δ is assumed fixed throughout the execution and not allowed to dynamically evolve as well. The reason is that Bitcoin was designed with exactly this setting in mind. (To see evidence of this consider that the PoW production in the protocol is

[6] In basketball, to say a player has "hot hands" means the player is on a streak of making many consecutive shots. A question that has dogged researchers, coaches and fans for years is whether players on these streaks can defy random chance, or if hot hands are just an illusion and fit within statistical norms.

fixed to be at 10-minute intervals, irrespectively of the computational hashing power available to the network, which has increased significantly—and at periods of times even exponentially— since its initiation in 2009.)

As specified, the Bitcoin blockchain protocol [24] relies on a global clock being available to all parties, and this is the model we consider in this paper[7]. Recently, Garay, Kiayias and Shen [14] considered improving Nakamoto's protocol so that it does not depend on an external clock and it becomes its own timekeeper. We remark that results from our paper on properties of typical executions (cf. the "second technical challenge" outlined above) have already found applications in this recent timekeeper result, as well as on the analysis of Bitcoin Cash's alternative target recalculation functions [15]. We conclude by pointing out that directions for future work include proving the tightness of the necessary conditions for security. Recent work has achieved such tight bounds but for simplified variants of Nakamoto's protocol—without target recalculation and timestamp adjustments (cf. [16,7]).

Differences with previous versions of this paper. The current version of the paper is restructured to emphasize the clock adjustment and blockchain timestamp validation mechanisms consistent with Nakamoto's implementation, as well as the formulation of a new blockchain property we call *concentrated chain quality*, which might be of independent interest (see Section 3.2); this is reflected in the change of the paper's title.

Organization of the paper. The remainder of the paper is organized as follows. In Section 2 we present the network, protocol execution and adversarial model; in particular, we define the *dynamic bounded-delay setting* where our analysis is performed. We also present blockchain notation and Nakamoto consensus basics, and the protocol parameters and set of conditions mentioned above. In Section 3 we formalize the notion and prove the occurrence of hot-hand executions—i.e., the occurrence of concentrated clusters of honest blocks on the chain—, the concentrated chain quality property, and the ensuing timekeeping property of the ledger. Finally, in Section 4 we present an overview of the full analysis of Nakamoto's consensus protocol in the originally envisioned dynamic environment where parties—without synchronized clocks—come and go, resulting in the adjustment of the blocks' difficulty values but also of their clocks.

For the sake of readability, part of the material is presented in the appendix, including some standard mathematical facts (Appendix A), previously formulated blockchain and Nakamoto consensus properties—*Common Prefix* and *Chain Quality*, and *Consistency* and *Liveness*, respectively—(Appendix B), and the detailed specification of (an abstraction) of the protocol which includes in particular its clock adjustment mechanism, and its full analysis (Appendices C and D, respectively).

2 Model and Definitions

We describe our protocols in a "full" partially synchronous model where *both* communication and processors are partially synchronous. Specifically, in the model there is an upper bound Δ in the delay (measured in number of rounds) that the adversary may inflict to the delivery of any message, and an upper bound Φ on the potential difference in party's clocks (also measured in number of rounds—see below). The precise values of Δ and Φ will be *unknown* to the protocol (and in particular regular protocol participation will not rely on using Δ or Φ as a time-out parameter). However, the security of the protocol will be dependent on how specific protocol parameters relate to Δ and Φ in ways we will explicitly define. Observe that "rounds" still exist in the model, but now these are not synchronization rounds where messages are supposed to be delivered to honest parties. Next, we adapt Canetti's formulation of "real world" notion of protocol execution [4,5,6] for multi-party protocols, to the dynamic setting with a varying number of parties, bounded delays and partially synchronized clocks.

Round structure and protocol execution. As in [12,25], 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 adversary $\mathcal{A}$ is adaptive, and allowed to take control of parties on the fly, as well as rushing, meaning that in any given round the adversary gets to observe honest parties' actions before deciding how to react. Network

[7] The protocol implements such clock by having nodes querying other nodes in the network and possibly seeking user input—it has no way of deriving a clock from the protocol operation itself.

and hash function access is captured by a difussion ("gossiping") functionality and a random oracle (RO) functionality, respectively (see below). The diffusion functionality is similar to those in [12,25]; it allows the order of messages to be controlled by $\mathcal{A}$ (i.e., there is no atomicity guarantees when multiple messages are sent) and, furthermore, the adversary is allowed to "spoof" the source information on every message (i.e., communication is not authenticated). $\mathcal{A}$ can inject messages for selective delivery but cannot change the contents of the honest parties' messages nor prevent them from being delivered beyond Δ rounds of delay—a functionality parameter.

The environment program $\mathcal{Z}$ determines the protocol execution; it creates and interacts with other instances of programs at the discretion of a control program C. Following [5], $(\mathcal{Z}, C)$ forms of a *system of interactive Turing machines* (ITM's). The only instances allowed by C are those of the protocol program Π, an adversary $\mathcal{A}$. These are called ITI's (interactive Turing Machines Instances). We refer to [5] for further details on the mechanics of these aspects of the model. The only additional feature that is relevant to our setting is that we assume each instance is initialized with a special Boolean flag denoted by `active` which is set to false upon initialization.

Functionalities available to the parties. We next present the functionalities that are available to all parties running the protocol and the adversary and abstract the hash function (RO), the network and parties' clocks. Note that the functionalities below share a common state and realizing them by other protocols is outside the scope of the present work; in our exposition, they merely capture explicitly the assumptions we make about our execution model.

- *The RO functionality.* It accepts queries of the form (Compute, x) and (Verify, x, y). For the first type of query, assuming x was never queried before, a value y is sampled from $\{0,1\}^\kappa$ and it is entered to a table T_H. If x was queried before the pair (x, y) is recovered from T_H. The value y is provided as an answer. For the second type of query, a membership test is performed on the table. Honest parties are allowed to ask *one query per round* of the type Compute and unlimited queries of the type Verify.[8] The adversary $\mathcal{A}$ is given a bounded number of Compute queries per round and no Verify queries (the adversary can easily simulate those locally). The bound for the adversary is determined as follows. Whenever a corrupted party is activated the bound is increased by 1; whenever a query is asked the bound is decreased by 1 (it is not necessary that the specific corrupted party makes the query).
- *The diffusion functionality.* Message passing and round bookkeeping is maintained by this functionality. A round variable *globalclock* is initialized to 0. For each party a string denoted by RECEIVE() is maintained and the party is allowed to fetch the contents of its corresponding RECEIVE() at any time. The functionality records all messages of the form (Diffuse, m) it receives from the parties. Completion of a round for a party is indicated by sending a special message (RoundComplete). The adversary $\mathcal{A}$ is allowed to receive all the currently recorded Diffuse messages at any time and messages to the RECEIVE() strings as desired. The round is completed when the adversary submits its (RoundComplete) message. In such case, the functionality inspects the contents of all RECEIVE() strings and includes any messages m that were diffused by the parties Δ rounds ago but not contributed by the adversary to the RECEIVE() tapes (in this way guaranteeing message delivery up to Δ rounds). It also flushes any diffuse records that are placed in the RECEIVE() string of all parties. The variable *globalclock* is then incremented and a new round begins.
- *The clock functionality.* This functionality, parameterized by Φ, maintains parties clocks' values within this bound with respect to global time. The parties use (RequestTime) to obtain its local clock value from the functionality. The adversary $\mathcal{A}$ is allowed to issue (AddShift, P, d),with $d \in [-\Phi, \Phi]$, commands to the functionality, in order to alter party P's clock value, so long as $|d| \leq \Phi$, i.e., honest parties' clocks are roughly synchronized.[9]

The dynamic partially synchronous setting. Given the functionalities as described above observe that, contrary to prior formalizations, the adversary can choose the termination of the round thus deciding on the

[8] Note that we exclude denial-of-service attacks from our modeling, where $\mathcal{A}$ depletes the running time of parties by sending them too many messages for verification.

[9] Refer to [18] for further details on clock functionalities.

spot how many honest parties were activated adaptively. (In previous works, the adversary is restricted to a preset number of activations—cf. [12,25,13].) In each round, the number of parties that are active in the protocol is denoted by n_r and is equal to the total number of parties that have submitted the (RoundComplete) indicator to the diffusion functionality and have their internal flag `active` set to true. Determining n_r can only be done by examining the view of all honest parties and is not a quantity that is accessible to any of the honest parties individually. The number of parties controlled by $\mathcal{A}$ in a round r is similarly denoted by t_r.

Parties, when activated, are able to read their input tape INPUT() and communication tape RECEIVE() from the diffusion functionality. If a party finds that its `active` flag is false, it enters a "bootstrapping" mode where it will diffuse a discovery message and synchronize with the rest of the active parties in the network (in the case of Nakamoto consensus, the party will send a request for the latest blockchains, will collect all of them until a time-out parameter is reached and then will pick the most difficult one to start mining).[10] When the synchronization phase terminates, the party will set its `active` flag to true and after this point it will be counted among the honest parties. An honest party goes "offline" when it misses a round, i.e., the adversary issues a (RoundComplete) but that party misses the opportunity to complete its computation. To record this action, whenever this happens we assume that the party's `active` flag is set to false (in particular this means that a party is aware that it went offline; note, however, that the party does not need to report it to anyone). Also observe that parties are unaware of the set of active parties. As in previous works (e.g., [12]), we assume, without loss of generality, that each honest party has the same computational power.[11]

We will restrict the environment to fluctuate the number of parties in a certain limited fashion. Suppose $\mathcal{Z}$ with fixed coins produces a sequence of parties n_r, where r ranges over all rounds of the execution. We define the following property, which is a finite-sequence version of a similar property introduced in [13] for infinite sequences.

Definition 1. *For $\gamma \in \mathbb{R}^+$ we call $(n_r)_{r\in[0,B)}$, where $B \in \mathbb{N}$,* (γ, s)-respecting *if for any set $S \subseteq [0, B)$ of at most s consecutive integers,* $\max_{r\in S} n_r \leq \gamma \cdot \min_{r\in S} n_r$.

We say that $\mathcal{Z}$ is (γ, s)-*respecting* if for all $\mathcal{A}$ and coins for $\mathcal{Z}$ and $\mathcal{A}$ the sequence of parties n_r is (γ, s)-respecting.

The term $\{\text{VIEW}^P_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ 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 consider a "standalone" execution without any auxiliary information and we will thus restrict ourselves to executions with $z = 1^\lambda$. For this reason we will simply refer to the ensemble by $\text{VIEW}^P_{\Pi,\mathcal{A},\mathcal{Z}}$. The concatenation of the view of all parties ever activated in the execution is denoted by $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$.

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 $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$ 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 λ as well as in a parameter k that is selected from $\{1, \ldots, \lambda\}$. (With foresight, we note that in practice one would be able to choose k to be much smaller than λ, e.g., $k = 6$.)

Blockchain and Nakamoto consensus basics. The Nakamoto (ledger) consensus protocol can be abstracted in four main subroutines. The main subroutine has two modes operation: regular and bootstrapping. In the bootstrapping mode, the node listens to the network for a certain period of time to collect sufficient number of blocks so that it determines a legitimate chain to build upon. The node will remain in this mode for a number of rounds that is determined by a parameter Δ_{bootstr}. When that time window passes the node will switch to regular mode.

During regular mode, the node's state can be abstracted as a pair $(st, \mathcal{C})$. We follow previous work that focused on the core of the Bitcoin protocol, called the *Bitcoin backbone protocol* [13], and we abstract out

[10] Refer to Section 2 (specifically, the discussion on the Δ_{bootstr} parameter) for an adequate time-out value depending on the epoch's length and probability of at least one honest party out of the initial number of parties solving a PoW.

[11] A real-world mining pool or party of a certain hashing power can be thought of as a set of flat-model parties.

considerations related to managing secret-keys and transaction issuance on behalf of the node. These would be incorporated in an input selection function $I(\cdot)$ that acts and updates the node's state st. We focus on the chain data structure $\mathcal{C}$ that is defined as follows. 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 $B = \langle r, prev, x, ctr\rangle$ where $prev \in \{0,1\}^\kappa, x \in \{0,1\}^*$, and $r, ctr \in \mathbb{N}$ are such that they satisfy the predicate $\mathsf{validblock}^T(B)$ defined as $(H(ctr, G(r, st, x)) < T) \wedge (ctr < 2^{32})$. Observe that each block B is associated with a timestamp r.

A *blockchain*, or simply a *chain* $\mathcal{C}$ is a (possibly empty) sequence of *blocks*; the rightmost block by convention is denoted by $\mathrm{head}(\mathcal{C})$ (note $\mathrm{head}(\varepsilon) = \varepsilon$) The blocks in a chain are connected in the sense that if $B_i = \langle r_i, prev_i, x_i, ctr_i\rangle$ then $B_{i+1} = \langle r_{i+1}, prev_{i+1}, x_{i+1}, ctr_{i+1}\rangle$ that satisfies $prev_{i+1} = H(ctr_i, G(r_i, prev_i, x_i))$. By convention $B_0 = \varepsilon$ and $prev_0 = \varepsilon$. In practice, B_0, a.k.a. the "genesis" block, may be selected to be a valid block and furthermore contain some unpredictable string to ensure that no attacker could pre-mine blocks. Nevertheless, these features are not significant for our analysis and we ignore them.

We measure the length $\mathrm{len}(\mathcal{C})$ of a chain $\mathcal{C}$ as the number of blocks it contains. We will use the notation $\mathcal{C}^{\lceil k}$ to denote the chain that results after "pruning" the k latest blocks. Moreover, if $\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$ we write $\mathcal{C}_1 \preceq \mathcal{C}_2$. The data contained in the blocks of a chain, collected in a vector will be denoted by $\mathbf{x}_\mathcal{C}$. Similarly, $\mathbf{r}_\mathcal{C}$ is the sequence of timestamps of $\mathcal{C}$.

Next, we describe how the appropriate target T is determined for each block. The initial target is a parameter of the protocol denoted by T_0. In the Bitcoin parameterization this is set to 2^{224} and is the highest possible target. The target is updated every m blocks, which is another parameter of the protocol. In the Bitcoin parameterization it is set to 2016. The initial number of parties in the system, denoted by n_0, are assumed to be capable of producing m blocks in expectation over a length of time equal to Δ_{epoch}. In Bitcoin's parameterization Δ_{epoch} corresponds to 2 weeks. We will refer to this length of time as an "epoch." The difficulty of each block is measured in terms of how much harder it is to produce a block compared to a block using the initial target T_0; i.e., the blocks of the initial epoch have all difficulty 1. We take a slightly more general approach and define the difficulty of a block to be $1/T$. We will use the notation $\mathrm{diff}(\mathcal{C})$ to denote the sum of the difficulties of all blocks in chain $\mathcal{C}$.

The difficulty of the next block to be mined is determined by a function $D(\cdot)$ that takes as input the sequence of timestamps corresponding to a given chain $\mathcal{C}$. The function $D(\cdot)$ parses the timestamps and identifies the last complete epoch of m blocks; if no such epoch is defined the target is by definition equal to T_0. Let r be the timestamp corresponding to last block of the last complete epoch, T its target and r_0 the timestamp corresponding to the last block of the previous epoch (or the genesis block timestamp if such epoch does not exist). The function $D(\cdot)$ returns as the next target the value $T \cdot \frac{r-r_0}{\Delta_{\mathsf{epoch}}}$, unless $\frac{r-r_0}{\Delta_{\mathsf{epoch}}} \notin [\frac{1}{\tau}, \tau]$, in which case it returns $\tau \cdot T$, when $\frac{r-r_0}{\Delta_{\mathsf{epoch}}} > \tau$, or $1/\tau \cdot T$, otherwise. The parameter τ is a "dampening factor;" In the case of Bitcoin's parameterization it is set to 4.

Protocol parameters and their conditions. Our formalization of the Bitcoin protocol involves a number of parameters. We summarize them below so that they can be all found in one place.[12]

- δ: Advantage of honest parties[13] ($t_r < (1-\delta)n_r$ for all r);
- (γ, s) : It restricts the fluctuation of the number of parties across rounds (Definition 1); we set $s = \tau m/f$.
- Δ_{fwd} is how far ahead a clock of a party can be and still produce an acceptable bock. $\Delta_{\mathsf{bootstr}}$ is the time it takes for parties to bootstrap after joining.

An important parameter, which is a function of the protocol's initialization parameters n_0 (the number of parties at the onset) and T_0 (the initial difficulty target), determines the block production rate per round.

- $f \in (0,1)$: The probability at least one honest party out of n_0 computes a block for target T_0; more generally, for the case of target T and n parties, we will overload the notation and also use f as a function i.e., $f(T, n)$.

[12] Indeed, a thorough read and understanding, in particular of the more complicated expressions, is not needed for the general comprehension of our results.

[13] Note that we denote the number of honest parties at round r by n_r and the number of parties controlled by the adversary by t_r, so that the total number of parties is $n_r + t_r$. Although this is not standard, it simplifies several expressions and is also in agreement with notation in [13].

The protocol strives to maintain the probability of a successful round as close to f as possible. This is achieved via target recalculation, applied every epoch.

- $m \in \mathbb{N}$: The length of an epoch in number of blocks;
- $\tau \geq 1$: The dampening factor, used to curb the change of target recalculation.

Given the above, the value $\tau m/f$ is the longest an epoch might take to complete and $\gamma \geq 1$ is an estimate of how much the number of parties can change in such a time interval.

The following parameter is important for clock adjustment, one of the main subjects of this paper:

- $k_{\text{med}} \in \mathbb{N}$: The **median timestamp** will be determined by $2k_{\text{med}} - 1$ blocks.

The next two parameters are related to security:

- λ : The security parameter;
- $\kappa = \Omega(\lambda)$: The size of the hash function's range.

Note that we assume $k_{\text{med}} = O(\log \lambda)$ (in the actual protocol parameterization $k_{\text{med}} = 6$). To achieve security, we will argue concentration of several random variables. Furthermore, in any exponentially long (in the security parameters) execution bad events are bound to happen. We use the following notation.

- ϵ: Quality of concentration of random variables;
- L : The total number of rounds in the execution.

For L polynomially bounded in λ, our statements will fail with probability $\mathsf{poly}(\lambda)(2^{-\kappa} + e^{-\lambda})$.

In our analysis we will study events over intervals of rounds. In order to achieve desired probability of error (in the order of $e^{-\lambda}$) we will need to work with intervals of at least ℓ rounds, where

$$\ell = \left\lceil \frac{4(1+3\epsilon)}{\epsilon^2 f[1-(1+\delta)\gamma^2 f]^{\Delta+1}} \cdot \max\{\Delta, \tau\} \cdot \gamma^3 \cdot \lambda \right\rceil, \quad \text{for } \epsilon \leq \frac{\delta}{16}. \tag{1}$$

For our analysis to go through, the above parameters should satisfy certain **conditions** which we now discuss. First, we will require a lower bound on the epoch length which incorporates ℓ.

$$\frac{\epsilon^2 m}{f} \geq \max\left\{ \frac{\lambda^3 4^{k_{\text{med}}+4}}{\delta^3 f} + \ell + 2\Delta,\ 8(\Delta_{\mathsf{fwd}} + \Phi),\ \left(\ell + 3\Delta\right) \cdot \frac{2^5(1+\delta)^2\gamma^6}{(1-\epsilon)^3} \right\}. \tag{C1}$$

Second, the number of rounds Δ that the adversary may delay messages relative to the block production rate should be upper bounded.

$$[1-(1+\delta)\gamma^2 f]^{2\Delta} \geq 1-\epsilon. \tag{C2}$$

Third, we will require in our analysis that the fluctuation rate of the parties is absorbed by the honest parties' advantage.

$$\gamma \leq 1 + \delta/8. \tag{C3}$$

The time-out parameter for bootstrapping, $\Delta_{\mathsf{bootstr}}$, should allow enough time for the messages of a joining party to reach active parties and enough time for their response to come back. The drift between the clocks of two honest parties should not be greater than Δ_{fwd}, otherwise one honest party might not accept a block computed by another honest party. These amount to requiring $\Delta_{\mathsf{bootstr}} \geq 2\Delta$ and $\Delta_{\mathsf{fwd}} \geq 2\Phi$.

3 Hot-Hand Executions and *Concentrated* Chain Quality

In previous works the security of the Bitcoin protocol was argued by reducing the failure of a property (such as common prefix or chain quality) to an event over a large set of rounds which could be shown to be of negligible probability using standard concentration techniques. In contrast to previous works, it is not sufficient in our setting to consider large sequences of rounds. In fact, over any sufficiently large sequence of rounds an adversary can control most consecutive blocks in an honest party's chain. It follows that any function that is based on the contents of consecutive blocks on the chain cannot be guaranteed to have most inputs contributed by honest parties. On the other hand, it may be possible to obtain concentrated clusters of honest blocks occasionally on the chain. To investigate these issues, in this section we study a new class of protocol executions that we call *hot hand* executions and introduce a new property for blockchain protocols called *concentrated chain quality*.

3.1 Hot-Hand Executions

We now define a new class of executions, called *hot-hand*. The structure such executions are required to have will allow us to show that concentrated chain quality holds with high probability. At a high level, a hot-hand execution contains special streaks of honest successes. What is special about these streaks is the structure of the surrounding rounds, which is such that the honest blocks that correspond to the streak are guaranteed to form a segment of consecutive honest blocks in any chain.

We define a sequence of random variables $\{(V_i, R_i) : i > 0\}$ taking values in $\mathbb{Z} \times \mathbb{N}$, with respect to a sequence of rounds $r_1, r_2, \ldots$ and a target T. Set $R_1 = r_1$, and for $i > 0$ set $R_{i+1} = r_{j+1}$, where $r_j \geq R_i$ is the least round such that exactly one of the following three events occurred:

- At least one adversarial block was created in r_j. In this case V_i is minus the number of blocks acquired by the adversary during round r_j.
- An honest block was created in $r_j - \Delta + 1 \geq R_i + \Delta - 1$ and no other block was created in $[R_i, r_j]$. In this case $V_i = 1$.
- Otherwise $V_i = 0$ and either there was an honest block created in $r_j < R_i + \Delta - 1$, or two honest blocks were created at rounds $r \leq r_j$ with $r_j < r + \Delta$.

For an interval of rounds $S = [u, v]$, we write $V(S) = \sum V_i$, where the sum is over $\{i : u \leq R_i \leq v\}$ and the sequence $\{(V_i, R_i) : i > 0\}$ is defined with respect to the sequence of rounds $u, u + 1, \ldots$. Before presenting the definition of a "winning streak," we present the definition of *strongly* isolated successful rounds (a stronger version of an analogous definition that can be found in the full version of [12]).

Definition 2. *A round r is called* strongly-isolated successful *if an honest query was successful for target T in round r and no other query was successful for target T in the interval of rounds $(r - \Delta, r + \Delta)$.*

Definition 3 ((Winning) Streak). *An interval that contains at least k_{med} strongly-isolated successful rounds and no other honest or adversarial blocks were computed in it is called a* streak. *A streak $[u, v]$ such that $V(S) \geq 0$ for any $S = (v, v']$ and $V(S) \geq 0$ for any $S = [u', u)$ is a* winning streak.

Note that $V_i = 1$ indicates a strongly-isolated successful round and the number of such rounds in a set S is at least $V(S)$.

For a given execution E we associate with each set $S = [u, v]$ of at most s consecutive rounds the target T of the first honest query in S. We want to say that T is good if E_u forces T to be good for each round in S. Formally, we say T is *good* for $S = [u, v]$ in E, if there is $n' \in [v - s, u]$ such that $f/2\gamma \leq pn'T \leq (1 + \delta)\gamma f$. Note that this implies $\gamma^2 f/2 \leq pn_r T \leq (1 + \delta)\gamma^2 f$ for each $r \in S$.

Definition 4 (Hot-hand execution). *An execution E is* hot-hand *if for any interval S of at least $4^{k_{\text{med}}+4}\lambda^3/\delta^3 f$ rounds such that the associated target T is good for S in E, there is a winning streak in S with respect to T.*

We wish to show that an execution is hot-hand with overwhelming probability. Since we are studying a small set of rounds each time, we are going to absorb the fluctuation γ in the number of parties in the advantage δ of the honest parties. Specifically, recall the property of n' in the paragraph before Definition 4 and set $n = \lceil n'/\gamma \rceil$. In a (γ, s)-respecting environment, the number of honest parties in S may fluctuate between n and $\gamma^2 n$ (because $\gamma^2 n \geq \gamma n' \geq \lfloor \gamma n' \rfloor$). Our plan is to reduce the question of the existence of a winning streak in S to a question in a *static* execution with n honest parties and a non-adaptive adversary controlling a fixed number of $t < (1 - \delta/2)n$ parties. This is going to work because, for $\gamma \leq 1 + \delta/8$,

$$(\gamma^2 - 1)n + (1 - \delta)\gamma^2 n \leq [(2 - \delta)(1 + \delta/8)^2 - 1]n < (1 - \delta/2)n.$$

Thus, we may handle the fluctuation of parties above n by allowing the adversary to control all of the excess parties.

Winning streaks in the *static* setting. We make the above argument formal via a coupling argument. We show that the probability that an interval of rounds is a winning streak in this static setting is at most that in the dynamic setting where the honest parties fluctuate between n and $\gamma^2 n$ against an adaptive adversary.

To prove this, we generate both distributions by starting with the maximum number $\lfloor (2-\delta/2)\gamma^2 n \rfloor$ of oracle outputs per round. In the dynamic setting, the strategy of the adaptive adversary determines in the beginning of each round how many of these outputs will be discarded, while in the static setting this number is fixed. Note that in the dynamic setting there are at least as many honest parties as in the static one. Furthermore, we distribute this excessive number of at most $(\gamma^2-1)n$ honest queries to the static adversary. Under this construction, every honest query in the static setting is also an honest query in the dynamic setting. Conversely, every query in the dynamic setting is a query in the static one. As a result, the following properties are satisfied:

- Every successful round in the static setting is also successful in the dynamic.
- If there are no successful queries in a round of the static setting, then the same holds for this round in the dynamic setting.

We observe now that if an interval $[u, v]$ is a winning streak for a given sequence of oracle outputs in the static setting, then this interval is also a winning streak for this sequence in the dynamic setting. Indeed, it follows directly from the above properties that any (strongly) isolated successful round r in the static setting is still (strongly) isolated in the dynamic one. Now consider an interval $S = [u', u)$. If $V(S)$ is non-negative in the static setting, then it is also non-negative in the dynamic setting, as only more honest queries are introduced.

Having reduced the dynamic setting to the static setting, we now proceed with the analysis in the static setting outlined above.

Lemma 1. *For any round u, the probability that $V(S) \geq 0$ for all $S = [u, v]$ is at least $\delta/8$.*

Proof. This follows from Theorem 13 applied to the sequence $(V_i : i > 0)$. We need to show that $\mathbf{E}[V_i] \geq \delta/8$ for any $i > 0$.

For each nonnegative integer k set $q_k = \mathbf{Pr}[V_i = -k]$ and $q = \mathbf{Pr}[V_i = 1]$. Setting $p_T = pT$ and recalling $\sum_{k\geq 0} x^k = 1/(1-x)$,

$$q = \sum_{r\geq 0} (1-p_T)^{(n+t)r} \cdot \frac{p_T n}{1-p_T} \cdot (1-p_T)^{(n+t)(2\Delta-1)} = \frac{1}{1-(1-p_T)^{(n+t)}} \cdot \frac{p_T n}{1-p_T} \cdot (1-p_T)^{(n+t)(2\Delta-1)}.$$

Under the hypothesis that T is good, it holds that $p_T n \leq (1+\delta)\gamma^2 f$. Using this, the fact that $t < (1-\delta/2)n$, Bernoulli's inequality, and Condition C2, we obtain the lower bound

$$q > \frac{p_T n (1-p_T)^{2(n+t)\Delta}}{p_T(n+t)} > \frac{[1-(1+\delta)\gamma^2 f]^{(4-\delta)\Delta}}{2-\delta/2} > \frac{2(1-\epsilon)^2}{4-\delta} > \frac{1}{2} + \frac{\delta}{16} \tag{2}$$

Since $\mathbf{E}[V_i] = q - \sum_{k>0} k q_k$, we now turn to the sum. Note that in the event $V_i = -k$, there is a round with k adversarial blocks that is preceded by at most one honest block computed less than Δ rounds ago. Thus,

$$\begin{aligned}
\sum_{k>0} k q_k &= \sum_{k>0} k \sum_{r\geq 0} (1-p_T)^{(n+t)r} \left(1 + \frac{p_T n(\Delta-1)}{1-p_T}\right) \binom{t}{k} p_T^k (1-p_T)^{t-k} \\
&< \sum_{r\geq 0} (1-p_T)^{(n+t)r} \cdot \frac{1+p_T n(\Delta-1)}{1-p_T} \cdot \sum_{k>0} k \binom{t}{k} p_T^k (1-p_T)^{t-k} \\
&< \frac{p_T t}{1-(1-p_T)^{n+t}} \cdot \frac{(1+p_T)^{n(\Delta-1)}}{1-p_T} < \frac{p_T t (1-p_T)^{-n(\Delta-1)-1}}{1-(1-p_T)^{n+t}}
\end{aligned}$$

For the second inequality, note that the sum over k is the expected value of a binomial and equals $p_T t$; also recall $\sum_{k\geq 0} x^k = 1/(1-x)$ and Bernoulli's inequality. Now,

$$\sum_{k>0} kq_k < \frac{2-\delta}{4-\delta} \cdot \frac{p_T(n+t)(1-p_T)^{-n(\Delta-1)-1}}{1-(1-p_T)^{n+t}} < \frac{2-\delta}{4-\delta} \cdot \frac{(1-p_T)^{-n(\Delta-1)-1}}{(1-p_T)^{n+t}}$$
$$\leq \frac{2-\delta}{4-\delta} \cdot \frac{1}{(1-p_T n)^{2\Delta}} \leq \frac{2-\delta}{4-\delta} \cdot \frac{1}{1-\epsilon}.$$

The first inequality follows from $t < (1-\delta/2)n$; the second inequality can be argued as (7); next, apply Bernoulli's inequality and for the last inequality combine $p_T n \leq (1+\delta)\gamma^2 f$ and Condition C2. Finally, since $\epsilon \leq \delta/16$,

$$\mathbf{E}[V_i] > \frac{2(1-\epsilon)^2}{4-\delta} - \frac{2-\delta}{4-\delta} \cdot \frac{1}{1-\epsilon} \geq \frac{\delta}{8}.$$

□

Next, let X_r and Z_r denote the number of successful queries by honest parties and the adversary in round r, respectively. Consider an interval S of at least $4^{k_{\text{med}}+4}\lambda^3/\delta^3 f$ rounds and for $(X, Z) = (X_r, Z_r : r \in S)$, let $h(X, Z)$ be equal to the number of winning streaks beginning at a round in S. We claim that

$$\mathbf{E}[h] > \delta^2 |S|/2^{k_{\text{med}}+6}. \tag{3}$$

First note that, since $p_T n \geq f/2\gamma^2$, the expected number of blocks in S is at least $4^{k_{\text{med}}+4}\lambda^3/2\delta^3\gamma^2 \geq 4^{k_{\text{med}}}\lambda^3$. It follows that the probability that less than k_{med} blocks are produced in S is less than $2^{-k_{\text{med}}\lambda^3/\delta^3}$. By (2), the probability a streak begins at a round u is at least

$$\left(\frac{1}{2} + \frac{\delta}{16}\right)^{k_{\text{med}}} \frac{1}{2^{k_{\text{med}}\lambda^3/\delta^3}} \geq \frac{1}{2^{k_{\text{med}}}}.$$

Irrespectively of the round u, it follows by Lemma 1 with respect to the (backward) sequence of rounds $u, u-1, \ldots,$ that the probability that $V([u,r]) \geq 0$ for all $r \leq u$ is at least $\delta/8$. Similarly, for any round v where the final block of the streak is computed, the probability that $V([v,r]) \geq 0$ for all $r \geq v$ is at least $\delta/8$. Since these events are independent, the bound follows.

We will now show that $h(X, Z)$ is concentrated around its expected value. To that end, it will be convenient to assume that each X_r and Z_r never surpass λ. The probability that more than λ queries out of n are successful can be bounded as in [23, Theorem 4.4] by $(ep_T n/\lambda)^\lambda < e^{-\lambda}$ (note that Condition C2 implies $(1+\delta)\gamma^2 f \leq \epsilon \leq \delta/16$ and so $\lambda \geq 1 \geq 16 p_T n > e^2 p_T n$).

Lemma 2. *Assuming that neither the honest parties nor the adversary acquired more than λ blocks in a single round of an interval S, then f is $\max\{k_{\text{med}}+2, 2\lceil\lambda/k_{\text{med}}\rceil\}$-Lipschitz over S.*

Proof. We are interested in the maximum value of $|h(x,z) - h(x',z')|$ over $(x,z),(x',z') \in \{0,1\}^{|S|}$ that differ only on the i-th position. Consider winning streaks $W = [u,v]$ and $W' = [u',v']$ and a round r. We say W' lies between W and r if $v < u'$ and $v' < r$. Suppose $x_i > x'_i$ and $x_i > 1$. We claim that any winning streak W of (x,z) with the property that at least k/k_{med} winning streaks lie between W and i is also a winning streak of (x',z'). This is because we can assume $|x_i - x'_i| \leq k$ and so the winning streaks in between counter-balance the difference $|x_i - x'_i|$. Note also that $x_i > 1$ implies that the blocks computed in round i do not themselves belong to a streak. It follows that even if $x_i = k$ and $x'_i = 0$, the winning streaks that are "lost" in (x',z') are at most $\lceil k/k_{\text{med}}\rceil$ to the right of i and at most the same number on the left. In the case $x_i = 1$ and $x'_i = 0$, these streaks can be at most 2. However, the block computed at i can belong to at most k_{med} streaks. The statement follows. □

We are now ready to prove this section's main result, that an execution is hot-hand with overwhelming probability in the security parameter λ.

Theorem 1. *The probability an execution is not hot-hand is* $\mathsf{poly}(\lambda) \cdot e^{-\lambda}$.

Proof. We apply Theorem 11 to the sequences of random variables $(X_j, Z_j)_{j\in[N]}$ that correspond to $N = \lceil \delta^{-4} 4^{k_{\text{med}}+6} \lambda^3 \rceil$ rounds of a static execution with target T, n honest parties, $t < (1-\delta/2)n$ adversarial parties, and Γ_j the event that neither the honest parties nor the adversary acquired more than λ blocks in round j. We apply the theorem with $c_j = \lambda$, $d = N$, $\mathbf{Pr}[(X,Z) \notin \Gamma] \leq Ne^{-\lambda}$ (as discussed before Lemma 2), and $t = \delta^2 N/2^{k_{\text{med}}+6} - Ne^{-\lambda}$. We obtain

$$\mathbf{Pr}[h(x) \leq 0] \leq \exp\Big\{-\frac{\delta^4 N - O(N^2)e^{-\lambda}}{\lambda^2 4^{k_{\text{med}}+6}}\Big\} + N^2 e^{-\lambda} = \mathsf{poly}(\lambda)\cdot e^{-\lambda},$$

where the last equality follows from Condition C1. We finish the proof with a union bound over the relevant intervals. □

3.2 Concentrated CQ and Timekeeping: New Blockchain Properties

As mentioned in the introduction, in this paper we observe that the "chain quality" property [12,25,13] is too weak to prove that the above clock adjustment mechanism is consistent with real time. Next, we introduce *concentrated chain quality* (CCQ), a property positing that honest parties' chains should include high concentrations of honest blocks at regular intervals. As we will show, CCQ constitutes the essential element to demonstrate that Nakamoto's design is a consistent timekeeper. Further, the connection to hot-hand executions from the previous subsection should now be apparent.

Definition 5 (Concentrated Chain Quality). *The* concentrated chain quality *property* Q_{ccq}*, with parameters* $K, k \in \mathbb{N}$*, states that for any honest party* P *with chain* $\mathcal{C}$ *in* $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$ *and any interval of at least* K *rounds,* $\mathcal{C}$ *contains* k *consecutive blocks that were computed in this interval of which at least half are honest.*

Our goal is to show that Q_{ccq} holds for appropriate K and $k = 2k_{\text{med}} - 1$ with high probability. This suffices, because the median timestamp of such a segment of k blocks is preceded by the timestamp of an honest block and therefore cannot be smaller.

CCQ will be essential in proving the following ledger property, which will be in addition to consistency and liveness (recall that parties in Nakamoto's protocol define a ledger $\mathcal{L}$ from their blockchains; see Section B in the supplementary material). The property requires the ledger to regularly report timestamps that are near the actual time by at most an offset $\Phi_{\text{drift}} \in \mathbb{N}$:

Definition 6 (Timekeeping). *For any honest party* P*, the timestamp* t *reported in the ledger* $\mathcal{L}$ *at round* r *satisfies* $|t - r| \leq \Phi_{\text{drift}}$.

Towards the existence of segments with an honest majority. We now state and prove a couple of lemmas that will be useful in the forthcoming analysis in Section 4 in proving that CCQ holds. These lemmas describe what happens when an honest block is "orphaned" and will be useful in arguing that blocks in winning streaks which are deep in an epoch cannot be orphaned.

Lemma 3. *Suppose that for round* v *there exist chains* $\mathcal{C}$ *and* $\mathcal{C}'$ *in* $\mathcal{S}_v$ *such that* $\mathcal{C} \setminus (\mathcal{C} \cap \mathcal{C}')$ *contains an honest block computed in round* r*. Let* u *the round that the last honest block on* $\mathcal{C} \cap \mathcal{C}'$ *was computed. Then, for the interval* $S = (u, v]$ *containing* r*,* $V(S) \leq 0$.

Proof. Let $\mathcal{B}$ be the set of honest blocks that contributed to $V(S)$. We prove $V(S) \leq 0$ by exhibiting the existence of a set of adversarial blocks $\mathcal{B}'$ computed in S such that $\{d \in B : B \in \mathcal{B}\} \subseteq \{d \in B : B \in \mathcal{B}'\}$.

Consider a block $B \in \mathcal{B}$ extending a chain $\mathcal{C}^*$ and let $d = \text{diff}(\mathcal{C}^* B)$. If $d \leq \text{diff}(\mathcal{C} \cap \mathcal{C}')$, let B' be the block of $\mathcal{C} \cap \mathcal{C}'$ containing d. Such a block clearly exists on $\mathcal{C} \cap \mathcal{C}'$ and was computed after round u by the adversary (due the definition of u). If $d > \text{diff}(\mathcal{C} \cap \mathcal{C}')$, note that by the definition of a strongly-isolated successful round there is a unique $B \in \mathcal{B}$ such that $d \in B$. Since $\mathcal{C}$ and $\mathcal{C}'$ are in $\mathcal{S}_v$ they are at least as heavy as $\mathcal{C}^*$. It follows that there is $B' \notin \mathcal{B}$ either on $\mathcal{C}$ or on $\mathcal{C}'$ that contains d. □

Lemma 4. *Suppose a block was computed by an honest party at round r and does not belong to the chain of an honest party at round $r' \geq r + \Delta$. Then r is contained in an interval S such that $V(S) \leq 0$.*

Proof. Let B be the block computed at round r by an honest party P. We are going to apply the previous lemma for a round v and chains $\mathcal{C}, \mathcal{C}' \in \mathcal{S}_v$ that we will define according to the following cases. If P adopted a chain $\mathcal{C}'$ not containing B, then let v be the previous round. Otherwise, let v be the least round not earlier than $r + \Delta$ such that an honest party P' does not have B in its chain $\mathcal{C}'$. Let $\mathcal{C}$ be the chain of P at round v. In both cases, $\mathcal{C}$ and $\mathcal{C}'$ belong in $\mathcal{S}_v$ and the previous lemma supplies the set S. □

4 Full Analysis of Nakamoto Consensus: Overview

In this section we present an overview of the full analysis of Nakamoto's consensus protocol in the originally envisioned dynamic environment where parties—without synchronized clocks—come and go, resulting in the adjustment of the blocks' difficulty values but also of their clocks, and which makes use of the new notions presented thus far. The detailed analysis is presented in Appendix D.

Regarding the timestamp sequence of a chain, the protocol specifies that it should satisfy the condition:

$$r_{i+1} > \mathsf{median}(r_i, \ldots, r_{i-2k_{\text{med}}+2}), \text{for } i > 2k_{\text{med}} - 1, \tag{4}$$

where k_{med} is a protocol parameter which, in the case of Bitcoin's parameterization, is set to 6 blocks. Refer to Appendix C for the full specification of the protocol.

The main challenge that arises in all the previous analyses of the protocol is that adjusting the blockchain clock in Nakamoto's protocol is influenced by the above median calculation over the timestamps of a sequence of consecutive blocks. Unfortunately, applying the "plain" chain quality property [12,25,13] over the sequence of k_{med} blocks will not result in anything meaningful as the property is too weak to ensure that the majority of medians is honest. It is for this reason that we need the stronger concentrated chain quality property to show that the honest parties can "push" the median forward sufficiently often. This is where our analysis of a hot-hand execution from the previous section will be crucial.

In the Bitcoin protocol, at any point in the execution, a node needs to determine the current time. We abstract this by a query $\mathsf{RequestTime}$ to the clock functionality, which responds with a reading that is within Φ of the correct time. Note that, in practice, Bitcoin achieves that by querying the system time, the median time of its neighbors in the peer-to-peer network as well as the human operator if a substantial deviation exists between the first two readings.[14] Such considerations are abstracted out in our modeling by the slack that is adversarially introduced in the $\mathsf{RequestTime}$ response from the clock functionality. In terms of determining the timestamp to use for the next block, the node should take into account the rule of the median of the past $2k_{\text{med}} - 1$ blocks (see equation (6) above): the current time will be "pushed" forward to ensure that it is ahead of the median.[15]

Additional notation and definitions. Following [13], our probability space is over all executions of length at most some polynomial in κ and λ and we denote by $\mathbf{Pr}$ the probability measure of this space. Furthermore, let $\mathcal{E}$ be a random variable taking values on this space and with a distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle.

If at round r exactly n parties query the oracle for target T, the probability at least one of them will succeed is

$$f(T, n) = 1 - (1 - pT)^n \leq pTn, \quad \text{where } p = 1/2^\kappa.$$

Note that Δ_{epoch} and the initial target T_0 implies in our model an initial estimate of the number of parties n_0; specifically, $n_0 = 2^\kappa m/(T_0 \Delta_{\mathsf{epoch}})$, i.e., the number of parties it takes to produce m blocks of difficulty

[14] As stated in `https://github.com/bitcoin/bitcoin/blob/master/src/timedata.cpp`, "never go to sea with two chronometers; take one or three" (cf. *triple modular redundancy*).

[15] Refer to the source code, `https://github.com/bitcoin/blob/master/src/miner.cpp`, line 30: `nNewTime = std::max(pindexPrev->GetMedianTimePast()+1, GetAdjustedTime());`.

$1/T_0$ in time Δ_{epoch}. We denote $f_0 = f(T_0, n_0)$ and we drop the subscript from f_0 and simply refer to it as f. Note the inequality

$$\frac{f(T,n)}{1-f(T,n)} = \frac{1-(1-pT)^n}{(1-pT)^n} = (1-pT)^{-n} - 1 > (1+pT)^n - 1 > pTn. \tag{5}$$

The first inequality is $1/(1-x) > 1+x$ and the second is Bernoulli's.

Next, in the full analysis (Appendix D) we present some definitions which allow us to introduce properties that we call "good," as applied to rounds, recalculation points, and chains. These properties are an intermediate step towards proving common prefix and chain quality, but are also interesting in their own right. In a nutshell, the underlying notion of "goodness" is concerned with the targets that the honest parties are querying the random oracle with. Refer to Definition 10 in the appendix.

Next, at a certain round of an execution, we would like to prove that the chain of every honest party has several desirable properties (along the notions just defined), and for that purpose we define a series of useful predicates with respect to such set of chains (Definition 11). Our goal is to show that, with high probability, an execution satisfies the blockchain properties defined in Section D.5. To fulfill this goal we first focus on showing that the execution satisfies the predicates defined above. In particular, we argue first that none of these predicates can fail, assuming proper initialization. We call such executions *typical*.

Typical executions. This notion follows the analogous definition in [13], but it is substantially simplified. 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 10, 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 suggested above, a typical execution will also be characterized by the absence of a number of bad events about the underlying hash function $H(\cdot)$ used to generate PoWs and is modeled as a random oracle. This section's main result is that almost all polynomially bounded in κ and λ executions are typical.

Theorem 2. *Assuming the ITM system $(\mathcal{Z}, C)$ runs for L steps, the probability of the event "$\mathcal{E}$ is not typical" is bounded by $O(L^2)(e^{-\lambda} + 2^{-\kappa})$.*

Properties of typical *and* hot-hand executions. Next, we study the validity of the predicates of Definition 11 over the space of typical *and* hot-hand executions in a (γ, s)-respecting environment. All statements assume a (γ, s)-respecting environment for $s \geq 2(1+\delta)\gamma^2 m/f$. Furthermore, Conditions (C1-3) (Section 2) are assumed to hold for the initialization parameters n_0 and T_0. The analysis first focuses on properties that require only an execution to be typical and subsequently properties that require the execution to be in addition hot hand. The first part follows [13], but the proofs use our simplified definition of a typical execution. Of the second part, in this overview we highlight the validity of the predicate related to the correct calculation of the block's timestamp:

Lemma 11. *In a typical and hot-hand execution and a (γ, s)-respecting environment,* $\textsc{GoodRounds}(r-1) \wedge \textsc{GoodChains}(r-1) \implies \textsc{MedianTime}(r)$.

This subsection concludes by showing that when executions are both typical and hot-hand, all predicates from Definition 11 are satisfied. Refer to Appendices D.3 and D.4 for details.

Theorem 3. *Consider a typical and hot-hand execution in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment. If the Conditions C1-3 (Section 2) are satisfied, then all predicates of Definition 11 hold.*

Blockchain properties. For parameters that satisfy Conditions C1-3 (Section 2) we can now show that a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment enjoys common prefix, chain

quality, and the new concentrated chain quality property. Here we just present the statements with the relevant parameters' values; refer to Appendix D.5 for details.

Theorem 4 (Common Prefix). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the common-prefix property holds for parameter $\epsilon^2 m$.*

Theorem 5 (Chain Quality). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the chain-quality property holds with parameters $\ell + 2\Delta$ and $\mu = \delta - 3\epsilon$.*

Theorem 6 (Concentrated Chain Quality). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the concentrated chain quality property holds for parameters $k \leq 2k_{\text{med}}$ and $K = 4^{k_{\text{med}}+4} \cdot \frac{\lambda^3}{\delta^3 f} + 2\ell + 4\Delta$.*

Ledger properties. Finally, for parameters that satisfy Conditions C1-3 as above we can show that a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment enjoys consistency, liveness, and the new timekeeping property. For details refer to Appendix D.6. Consistency follows directly from the common prefix property, that we showed to hold in the above circumstances.

Theorem 7 (Consistency). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than ϵm blocks deep.*

Liveness follows easily from Lemmas 5 and 7.

Theorem 8 (Liveness). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, Liveness is satisfied for depth $\epsilon^2 m$ with wait-time $(4\gamma^2+1)\epsilon^2 m/f$.*

To conclude, using the new *concentrated chain quality property*, we show that the timestamps on the blockchain are approximately accurate.

Theorem 9 (Timekeeping). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the Timekeeping property holds with $\Phi_{\text{drift}} = \max\{K + \Phi, \Delta_{\mathsf{fwd}} + \ell + 2\Delta\}$.*

Proof. Consider a block B in a chain $\mathcal{C}$ computed in round r and with timestamp t. Suppose $t > r + \Delta_{\mathsf{fwd}} + \ell + 2\Delta$. Then no honest party would adopt $\mathcal{C}$ for more than $\ell + 2\Delta$ rounds and it would become stale.

We now argue $t \geq r - K - \Phi$ and consider the K rounds preceding r. By concentrated chain quality there are $2k_{\text{med}} - 1$ blocks such that the median of the timestamps of these blocks is preceded by the timestamp of an honest block. This timestamp can be at most Φ rounds away from the round it was computed in and the timestamp of B cannot be smaller than this. □

References

1. L. Addario-Berry and B. A. Reed. *Ballot Theorems, Old and New*, pages 9–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
2. Adam Back. Hashcash. http://www.cypherspace.org/hashcash, 1997.
3. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Dynamic ad hoc clock synchronization. *Advances in Cryptology - EUROCRYPT 2021 - pages 399–428. Springer, 2021.*
4. Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
5. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, pages 136–145. IEEE Computer Society, 2001.*

7. Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and Nakamoto always wins. *CCS '20: 2020 ACM SIGSAC CCS, November 9-13, 2020*. pages 819–838.
8. Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, 2009.
9. Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.
10. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO*, 1992. pages 139–147.
11. Juan A. Garay and Aggelos Kiayias. Sok: A consensus taxonomy in the blockchain era. *Topics in Cryptology - CT-RSA 2020 - pages 284–318. Springer, 2020.*
12. 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 - pages 281–310. Springer, 2015.*
13. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. *CRYPTO* 2017 - pages 291–323.
14. Juan A. Garay, Aggelos Kiayias, and Yu Shen. Permissionless Clock Synchronization with Public Setup. *TCC 2022* - pages 181-211.
15. Juan A. Garay and Yu Shen. On Bitcoin Cash's Target Recalculation Functions. *AFT '21: 3rd ACM Conference on Advances in Financial Technologies* - pages 192–204.
16. Peter Gazi, Aggelos Kiayias, and Alexander Russell. Tight consistency bounds for bitcoin. *CCS '20: 2020 ACM SIGSAC November 9-13, 2020*. pages 819–838.
17. Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*. The Internet Society, 1999.
18. Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. TCC 2013, pages 477–498. 2013.
19. Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. *IACR Cryptology ePrint Archive*, 2015:1019, 2015.
20. Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. A better method to analyze blockchain consistency. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pages 729–744. ACM, 2018.*
21. Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
22. Colin McDiarmid. *Probabilistic Methods for Algorithmic Discrete Mathematics*, chapter Concentration, pages 195–248. Springer 1998.
23. Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
24. Satoshi Nakamoto. Bitcoin open source implementation of p2p currency. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source, 2009.
25. Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - pages 643–673, 2017.*
26. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
27. Ling Ren. Analysis of nakamoto consensus. *IACR ePrint Archive*, 2019:943, 2019.
28. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.
29. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.
30. J.M. Steele and Mathematical Association of America. *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. MAA problem books series. Cambridge University Press, 2004.
31. Lutz Warnke. On the method of typical bounded differences. *Combinatorics, Probability and Computing*, 25(2):269–299, 2016.

A Mathematical Facts

All the following definitions and statements assume finite probability spaces and random variables with finite means.

Definition 7. *[8, Definition 5.3] A sequence of random variables $(X_0, X_1, \dots)$ is a martingale with respect to the sequence $(Y_0, Y_1, \dots)$, if, for all $n \geq 0$, X_n is determined by $Y_0, \dots, Y_n$ and $\mathbf{E}[X_{n+1}|Y_0, \dots, Y_n] = X_n$.*

The following is closer to Theorem 3.15 in [22], but see also Theorems 8.1 and 8.2 in [8].

Theorem 10. *Let $(X_0, X_1, \dots)$ be a martingale with respect to the sequence $(Y_0, Y_1, \dots)$. Suppose an event G implies*

$$X_k - X_{k-1} \leq b \text{ (for all } k) \quad \text{and} \quad V = \textstyle\sum_k \mathbf{var}[X_k - X_{k-1}|Y_1, \dots, Y_{k-1}] \leq v,$$

Then, for non-negative n and t,

$$\mathbf{Pr}\big[X_n \geq X_0 + t \wedge G\big] \leq \exp\Big\{-\frac{t^2}{2v + 2bt/3}\Big\}.$$

Theorem 11. *[31, Theorem 1.6] Let $X = (X_1, \dots, X_N)$ be a family of independent random variables with X_j taking values in a set Λ_j and let $\Gamma = \prod_{j\in[N]} \Gamma_j$ where $\Gamma_j \subseteq \Lambda_j$. Assume there are numbers $(c_j)_{j\in[N]}$ so that $f : \prod_{j\in[N]} \Lambda_j \to \mathbb{R}$ satisfies the following. Whenever $x, x' \in \prod_{j\in[N]} \Gamma_j$ differ only in the j-th coordinate and $x, x' \in \Gamma$ we have $|f(x) - f(x')| \leq c_j$ and $|f(x) - f(x')| \leq d$ for all $x, x' \in \prod_{j\in[N]} \Lambda_j$ that differ in at least one coordinate. Then, for all $t \geq 0$,*

$$\mathbf{Pr}\Big[f(x) \leq \mathbf{E}[f(X)] - t - d\mathbf{Pr}[X \notin \Gamma]\Big] \leq \exp\Big\{-\frac{2t^2}{\sum_{j\in[N]} c_j^2}\Big\} + \mathbf{Pr}[X \notin \Gamma].$$

The following Cauchy-Schwarz converse (see [30]) will be of use.

Theorem 12. *For all non-negative real numbers $a_k, b_k, k = 1, 2, \dots, n$ that satisfy $m \leq a_k/b_k \leq M$ for some constants $0 < m \leq M < \infty$,*

$$\sqrt{\sum_{k=1}^{n} a_k^2}\sqrt{\sum_{k=1}^{n} b_k^2} \leq \frac{A}{G}\sum_{k=1}^{n} a_k b_k,$$

where $A = (m + M)/2$ and $G = \sqrt{mM}$.

Theorem 13 (Theorem 3 in [1]). *Let $X_1, X_2, \dots$ be an infinite sequence of iid integer random variables with mean $\mu > 0$ and maximum value 1 and for any $i \geq 1$ let $S_i = X_1 + \cdots + X_i$. Then*

$$\mathbf{Pr}[S_i > 0 \text{ for } n = 1, 2, \dots] = \mu.$$

B Blockchain and Nakamoto Consensus Properties

Blockchain properties. The blockchain data structure's two fundamental properties, adapted from [12,13], are related to the Nakamoto consensus properties of Consistency and Liveness (see below).

The *common prefix* property, parameterized by a value $k \in \mathbb{N}$, considers an arbitrary environment and adversary, and holds as long as any two parties' chains at two rounds have the earlier one subsumed in the former as long as k blocks are removed.

Definition 8 (Common Prefix). *The* common prefix *property Q_{cp} with parameter $k \in \mathbb{N}$ states that for any two honest parties P_1, P_2 holding chains $\mathcal{C}_1, \mathcal{C}_2$ at rounds r_1, r_2, with $r_1 \leq r_2$, it holds that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.*

The second property, called *chain quality*, quantifies the honest-party contributions in terms of aggregate difficulty that are contained in a sufficiently long and continuous portion of a party's chain. As a result, the property restricts the amount of difficulty (and hence number of blocks) contributed by the adversary to any sufficiently long segment of the chain.

Definition 9 (Chain Quality). *The* chain quality *property Q_{cq}, with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$, states that for any honest party P with chain $\mathcal{C}$ in* $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$*, and any segment of that chain of difficulty d such that the first block of the segment was computed at least ℓ rounds earlier than the last block, the blocks the honest parties have contributed in the segment have total difficulty at least $\mu \cdot d$.*

Nakamoto consensus properties. As mentioned in Section 1, *Nakamoto consensus* (aka "ledger consensus") is the problem where a set of servers (nodes, parties) operate continuously accepting inputs ("transactions") and incorporate them in a public data structure called the *ledger*. More specifically, the problem is to maintain a ledger of transactions serialized in the form of a transaction sequence $\mathcal{L}$; satisfying the following two properties [12,13]. Below we make the distinction between $\mathcal{L}$ and $\widetilde{\mathcal{L}}$, with the first denoting the settled ledger in the view of the party, and the second denoting the settled ledger with a sequence of transactions appended that are still not settled in the view of the party. In the context of Nakamoto's Bitcoin protocol, we note that $\widetilde{\mathcal{L}}$ will be the sequence of transactions defined by the chain $\mathcal{C}$ held by the party, while $\mathcal{L}$ will be the sequence of transactions defined by the prefix $\mathcal{C}^{\lceil k}$, where k is a security parameter.

- **Consistency** (cf. **Persistence** [12]): For any two honest parties P_1, P_2, reporting $\mathcal{L}_1, \mathcal{L}_2$ at rounds $r_1 \leq r_2$, respectively, it holds that $\mathcal{L}_1$ is a prefix of $\widetilde{\mathcal{L}}_2$.
- **Liveness** (parameterized by $u \in \mathbb{N}$, the "wait time" parameter): If a transaction tx is provided to all honest parties for u consecutive rounds, then it holds that for any player P, tx will be in $\mathcal{L}$.

We remark that the problem is a variant of the *state machine replication* problem [29] (see also [11]).

C The Bitcoin Backbone Protocol with Clock Adjustment: Specification

In this section we give a detailed description of an abstraction of Nakamoto's blockchain protocol with chains of variable difficulty and clock adjustment mechanism. The presentation is based on the description given in [12], called the "Bitcoin backbone" protocol.

C.1 The Protocol

As in [12], in our description of the 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 $\mathsf{validate}$ 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 2). 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 $V(\mathbf{x}_{\mathcal{C}})$; note that $V(\varepsilon) = \mathbf{true}$. Chains that fail these validation procedure are rejected. (Algorithm 1.)

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 straighrward 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 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 validate($r_{\mathsf{now}}, \mathcal{C}$)
  $valid \leftarrow V(\mathbf{x}_{\mathcal{C}})$
  if $valid = \text{True} \wedge (C \neq \varepsilon)$ **then** ▷ C is non-empty and meaningful w.r.t. $V(\cdot)$
    $r_{\text{R}} \leftarrow r_{\mathsf{now}} + \Delta_{\mathsf{fwd}}$
    $r_{\text{L}} \leftarrow 0$
    $len \leftarrow |\mathcal{C}|$
    $\langle r', st', x', ctr' \rangle \leftarrow \langle 0, \perp, \text{"genesis"}, \perp \rangle$
    $i \leftarrow 1$
    $T \leftarrow T_{\text{initial}}$
    while $(i \leq len) \wedge valid$ **do**
      $\langle r, st, x, ctr \rangle \leftarrow \text{head}(\mathcal{C}^{\lceil len-i})$ ▷ Get the i-th block; note $\mathcal{C}^{\lceil 0} = \mathcal{C}$
      $st_{\mathsf{prev}} \leftarrow H(ctr', G(r', st', x'))$ ▷ Calculate the hash of previous block
      if $\mathsf{validblock}_q^T(\langle r, st, x, ctr \rangle) \wedge (st_{\mathsf{prev}} = st) \wedge (r_{\text{L}} < r < r_{\text{R}})$ **then**
        $\langle r', st', x', ctr' \rangle \leftarrow \langle r, st, x, ctr \rangle$ ▷ Retain current block
        $timeseq \leftarrow \langle r \rangle || timeseq$ ▷ Prepend timestamp
        if $|timeseq| > 2k_{\text{med}}$ **then** ▷ We have enough timestamps for median
          $timeseq \leftarrow timeseq^{\lceil 1}$ ▷ Remove the last element
        end if
        $r_{\text{L}} \leftarrow \max\{r_{\text{L}}, \textbf{median}(timeseq) + 1\}$ ▷ Advance left time bound
      else
        $valid \leftarrow$ False ▷ Blockchain is not valid
      end if
      $T \leftarrow D(\mathbf{r}_{\mathcal{C}^{\lceil len-i}})$ ▷ Calculate next target
      $i \leftarrow i + 1$
    end while
  end if
  return $valid$
end function

Note that in the Bitcoin implementation it holds that $\Delta_{\mathsf{fwd}} = 2$ hours and $k_{\text{med}} = 6$.

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

function maxvalid($r_{\mathsf{now}}, \mathcal{C}_1, \ldots, \mathcal{C}_k$)
  $temp \leftarrow \varepsilon$
  for $i = 1$ to k **do**
    if validate($r_{\mathsf{now}}, \mathcal{C}_i$) **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 *Proof of work* single step based on hash functions $H(\cdot), G(\cdot)$ and target calculation function $D(\cdot)$.

function $\mathsf{pow}(r_{\mathsf{now}}, x, \mathcal{C})$
  if $\mathcal{C} = \varepsilon$ **then** ▷ Determine proof of work instance.
    $prev \leftarrow 0$
    $ctr \leftarrow 0$
  else
    $\langle r', prev', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$
    $prev \leftarrow H(ctr', G(r', prev', x'))$
  end if
  $B \leftarrow \varepsilon$
  $T \leftarrow D(\mathbf{r}_{\mathcal{C}})$ ▷ Calculate target for next block based on timestamps.
  $h \leftarrow G(r, prev, x)$
  if $(H(ctr, h) < T)$ **then** ▷ This $H(\cdot)$ invocation is subject to 1 query/round.
    $B \leftarrow \langle r, prev, x, ctr \rangle$
  end if
  $ctr \leftarrow ctr + 1 \mod 2^{32}$
  $\mathcal{C} \leftarrow \mathcal{C}B$ ▷ Chain is extended, if $B \neq \varepsilon$
  return $\mathcal{C}$
end function

The backbone protocol. The core of the protocol is similar to that of [13]. We recall some of the main functions as well as point to the new elements we have added compared to previous abstractions. Parties always check the `active` flag to make sure they detect they have missed one or more rounds. In case the `active` flag is false, they broadcast a special message '**Join**' (that requests the most recent version of the blockchain from other other parties) and enter into bootstrapping mode which lasts for a certain period of time denoted by $\Delta_{\mathsf{bootstr}}$. To respond to such request, when online parties receive the special request message in their Receive() tape they broadcast their chain. The input contribution function $I(\cdot)$ and the chain reading function $R(\cdot)$ are applied to the values stored in the chain; we are not concerned with the way these functions are defined.

As in past work, [12,13] the input tape of a party contains two types of symbols, Read and (Insert, *value*); other inputs are ignored. A Read results to the party applying function $R(\cdot)$ to its current chain and writing the result onto the output tape Output(), while a (Insert, *value*) symbol is taken into account by the $I(\cdot)$ function when it determines the contribution of the party in extending its chain.

One of the novel elements in the current treatment is the fact that the current time is always determined by querying the clock functionality, which allows the adversary to apply a drift on a party's clock up to the Φ bound.

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 "`active`" flag is False if and only if the party was inactive in the previous round. `bootstrapping` is initially False.

$currenttime \leftarrow \mathsf{RequestTime}$ ▷ request time from the clock functionality
`bootstrapping` $\leftarrow$ `bootstrapping` $\wedge$ `active`
if `active` $=$ True $\wedge \neg$`bootstrapping` **then** ▷ Ledger maintenance mode
  DIFFUSE(Ready)
  `round` $\leftarrow currenttime$
  $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\texttt{round}, \mathcal{C}, \text{all chains } \mathcal{C}' \text{ found in } \text{RECEIVE}())$
  `round` $\leftarrow \max\{\texttt{round}, 1 + \text{median of } \tilde{\mathcal{C}} \text{ last } 2k_{\text{med}} - 1 \text{ blocks}\}$
  if INPUT() contains READ **then**
    write $R(\mathbf{x}_{\mathcal{C}})$ to OUTPUT()
  end if
  $\langle st, x\rangle \leftarrow I(st, \tilde{\mathcal{C}}, \texttt{round}, \text{INPUT}(), \text{RECEIVE}())$
  $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(\texttt{round}, x, \tilde{\mathcal{C}})$
  if $(\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}) \vee (\mathsf{Join} \in \text{RECEIVE}())$ **then**
    $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
    DIFFUSE($\mathcal{C}$) ▷ chain is diffused when updated or when someone joins.
  end if
  DIFFUSE(RoundComplete)
else ▷ Bootstrapping mode
  `active` $\leftarrow$ True
  if `bootstrapping` $=$ True **then** ▷ Node is in the process of bootstrapping
    $timelapsed \leftarrow timelapsed + \max\{currenttime - timelapsed, 0\}$
  else ▷ Node just woke up and needs to bootstrap
    `bootstrapping` $\leftarrow$ True
    $timelapsed \leftarrow 0$
  end if
  if $timelapsed > \Delta_{\text{bootstr}}$ **then**
    `bootstrapping` $\leftarrow$ False ▷ Node is ready to engage
  else
    DIFFUSE(Join, RoundComplete) ▷ Node is asking for blocks to synchronize
  end if
end if

D The Full Analysis of Nakamoto Consensus in Detail

In this section we present the full analysis and proof of Nakamoto's consensus protocol in the originally envisioned dynamic environment where parties—without synchronized clocks—come and go, resulting in the adjusment of of the blocks' difficulty values but also of their clocks.

Regarding the timestamp sequence of a chain, the protocol specifies that it should satisfy the condition:

$$r_{i+1} > \mathsf{median}(r_i, \ldots, r_{i-2k_{\text{med}}+2}), \text{for } i > 2k_{\text{med}} - 1, \tag{6}$$

where k_{med} is a protocol parameter which, in the case of Bitcoin's parameterization, is set to 6 blocks. Refer to Appendix C for the full specification of the protocol.

The main challenge that arises in all the previous analyses of the protocol is that adjusting the blockchain clock in Nakamoto's protocol is influenced by the above median calculation over the timestamps of a sequence of consecutive blocks. Unfortunately, applying "plain" chain quality over the sequence of k_{med} blocks will not result in anything meaningful as the property is too weak to ensure that the majority of medians is honest. It is for this reason that we need the stronger concentrated chain quality property to show that the honest parties can "push" the median forward sufficiently often. This is where our analysis of a hot-hand execution from the previous section will be crucial.

In the Bitcoin protocol, at any point in the execution, a node needs to determine the current time. We abstract this by a query RequestTime to the clock functionality, which responds with a reading that is within Φ of the correct time. Note that, in practice, Bitcoin achieves that by querying the system time, the median time of its neighbors in the peer-to-peer network as well as the human operator if a substantial deviation exists between the first two readings.[16] Such considerations are abstracted out in our modeling by the slack that is adversarially introduced in the RequestTime response from the clock functionality. In terms of determining the timestamp to use for the next block, the node should take into account the rule of the median of the past $2k_{\text{med}} - 1$ blocks (see equation (6) above): the current time will be "pushed" forward to ensure that it is ahead of the median.[17]

D.1 Additional notation, definitions, and preliminary propositions

Following [13], our probability space is over all executions of length at most some polynomial in κ and λ and we denote by $\mathbf{Pr}$ the probability measure of this space. Furthermore, let $\mathcal{E}$ be a random variable taking values on this space and with a distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle.

If at round r exactly n parties query the oracle for target T, the probability at least one of them will succeed is

$$f(T, n) = 1 - (1 - pT)^n \leq pTn, \quad \text{where } p = 1/2^\kappa.$$

Note that Δ_{epoch} and the initial target T_0 implies in our model an initial estimate of the number of parties n_0; specifically, $n_0 = 2^\kappa m/(T_0 \Delta_{\mathsf{epoch}})$, i.e., the number of parties it takes to produce m blocks of difficulty $1/T_0$ in time Δ_{epoch}. We denote $f_0 = f(T_0, n_0)$ and we drop the subscript from f_0 and simply refer to it as f. Note the inequality

$$\frac{f(T, n)}{1 - f(T, n)} = \frac{1 - (1 - pT)^n}{(1 - pT)^n} = (1 - pT)^{-n} - 1 > (1 + pT)^n - 1 > pTn. \tag{7}$$

The first inequality is $1/(1 - x) > 1 + x$ and the second is Bernoulli's.

We will next present some definitions which will allow us to introduce a few ("good") properties. These properties are an intermediate step towards proving common prefix and chain quality, but are also interesting

[16] As stated in `https://github.com/bitcoin/bitcoin/blob/master/src/timedata.cpp`, "never go to sea with two chronometers; take one or three" (cf. *triple modular redundancy*).

[17] Refer to the source code, `https://github.com/bitcoin/blob/master/src/miner.cpp`, line 30: `nNewTime = std::max(pindexPrev->GetMedianTimePast()+1, GetAdjustedTime());`.

in their own. The next two definitions are about the notions of "good chain" and "good round." The underlying notion of "goodness" is concerned with the targets that the honest parties are querying the random oracle for. At a round r of an execution the n_r honest parties might be querying the random oracle for various targets. We denote by $T_r^{\min}$ and $T_r^{\max}$ the minimum and maximum of those targets.

With respect to parameters that appear as "free" in the following definitions (such as γ, Δ, ℓ), please refer to the next subsection.

Definition 10.

- *Round r is* good *if $f/2\gamma^2 \leq pn_rT_r^{\min}$ and $pn_rT_r^{\max} \leq (1+\delta)\gamma^2 f$.*
- *Round r is a* target-recalculation point *(or simply a* recalculation point*) of a chain $\mathcal{C}$, if $\mathcal{C}$ has a block created in r and with height a multiple of m.*
- *A target-recalculation point r is* good *if the target T for the next block satisfies $f/2\gamma \leq pn_rT \leq (1+\delta)\gamma f$.*
- *A chain is* good *if all its target-recalculation points are* good*.*
- *A chain is* stale *if for some u it does not contain an honest block computed in $[u-\ell-2\Delta, u]$.*
- *The blocks between two consecutive target recalculation points u and v on a chain $\mathcal{C}$ are an* epoch *of $\mathcal{C}$ and the* duration *of the epoch is $u-v$.*

At a certain round of an execution, we would like to prove that the chain of every honest party has several desirable properties (along the notions just defined). This, however, entails a stronger statement in the following sense. At any given round there might exist chains which do not belong to any honest party (perhaps because the adversary kept them private), but have the potential to be adopted by one (i.e., have sufficient difficulty). With that in mind we define the following set of chains of a round r.

$$\mathcal{S}_r = \left\{ \mathcal{C} \in E_r \;\middle|\; \begin{array}{c} (\mathcal{C} \text{ belongs to an honest party}) \text{ or } (\exists \mathcal{C}' \in E_r \text{ that belongs} \\ \text{to an honest party and either } \mathrm{diff}(\mathcal{C}) > \mathrm{diff}(\mathcal{C}') \text{ or} \\ \mathrm{diff}(\mathcal{C}) = \mathrm{diff}(\mathcal{C}') \text{ and } \mathrm{head}(\mathcal{C}) \text{ was computed} \\ \text{no later than } \mathrm{head}(\mathcal{C}')) \end{array} \right\},$$

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

Next, we define a series of useful predicates with respect to such set of chains.

Definition 11. *For a round r, let:*

- GoodChains$(r) \triangleq$ *"For all $u \leq r$, every chain in $\mathcal{S}_u$ is good."*
- GoodRounds$(r) \triangleq$ *"All rounds $u \leq r$ are good."*
- NoStaleChains$(r) \triangleq$ *"For all $u \leq r$, there are no stale chains in $\mathcal{S}_u$."*
- MedianTime$(r) \triangleq$ *"For all $u \leq r$ and $\mathcal{C} \in \mathcal{S}_u$, $\mathcal{C}$ has k_{med} consecutive honest blocks computed in the last $\lfloor \epsilon^2 m/f \rfloor$ rounds of any completed epochs."*
- Duration$(r) \triangleq$ *"For all $u \leq r$ and $\mathcal{C} \in \mathcal{S}_u$, the duration Λ of any epoch in $\mathcal{C}$ satisfies $\frac{1}{2(1+\delta)\gamma^2} \cdot \frac{m}{f} \leq \Lambda \leq 2(1+\delta)\gamma^2 \cdot \frac{m}{f}$."*
- CommonPrefix$(r) \triangleq$ *"For all $u \leq r$ and $\mathcal{C}, \mathcal{C}' \in \mathcal{S}_u$, $\mathrm{head}(\mathcal{C} \cap \mathcal{C}')$ was created after round $u-\ell-2\Delta$.*

Our goal is to show that, with high probability, an execution satisfies the blockchain properties defined in Section D.5. To fulfill this goal we will first focus on showing that the execution satisfies the predicates defined above. In particular, we will argue first that none of these predicates can fail, assuming proper initialization. We first define a number of additional random variables.

Random variables. In our analysis, we will be interested in estimating the difficulty acquired by honest parties during a sequence of rounds. Their number at a round r is denoted n_r and define the real random variable D_r equal to the sum of the difficulties of all blocks computed by honest parties at round r. Also, define Y_r to equal the maximum difficulty among all blocks computed by honest parties at round r, and Q_r to equal Y_r when $D_u = 0$ for all $r < u < r + \Delta$ and 0 otherwise. We call a round r such that $D_r > 0$ *successful* and one wherein $Q_r > 0$ *isolated successful*. Regarding the adversary, let t_r denote the number of

parties he controls at round r (equivalently, the number of random-oracle queries he can make at round r). Note that n_r and t_r are determined by the environment at the beginning of round r and should conform to the (γ, s)-respecting definition (Definition 1). We wish to upper bound the difficulty he can acquire during a set J of queries. Looking ahead, to obtain a good upper bound that holds with high probability, we will need some upper bound on the difficulty of a single block. However, the adversary may query the oracle for arbitrarily low targets and may obtain blocks of arbitrarily high difficulty. The following definition will allow us to work around these technical obstacles.

Consider a set of consecutive adversarial queries J and note that the execution up to the first query in J determines the target associated with it. We denote this target by $T(J)$ and say that $T(J)$ is *associated* with J. We define $A(J)$ and $B(J)$ to be equal to the sum of the difficulties of all blocks computed by the adversary during queries in J for target at least $T(J)/\tau$ and $T(J)$, respectively. That is, queries in J for targets less than $T(J)/\tau$ (resp. $T(J)$) do not contribute to $A(J)$ (resp. $B(J)$). While considering consecutive epochs of a particular chain, the target can either increase by at most τ (and $B(J)$ will be appropriate), or decrease by at most τ (and $A(J)$ will be useful).

For a set of rounds S or queries J we write $n(S) = \sum_{r \in S} n_r$ and similarly $t(S)$, $D(S)$, $Q(S)$. An interval of rounds (or queries) is a set of consecutive rounds and is denoted using bracket notation. For example, if u and v are two rounds such that $u \leq v$ we write $[u, v)$ for $\{r : u \leq r < v\}$.

Let $\mathcal{E}_{r-1}$ fix the execution just before round r. In particular, a value E_{r-1} of $\mathcal{E}_{r-1}$ determines the adversarial strategy and so determines the targets against which every party will query the oracle at round r and the number of parties n_r and t_r, but it does not determine D_r or Q_r. For an adversarial query j we will write $\mathcal{E}_{j-1}$ for the execution just before this query.

D.2 Chain Growth Lemma

This lemma appears already in [12] in a model with fixed difficulty and fixed number of parties.[18] Here we give a different proof that works in the dynamic bounded-delay model. The lemma provides a lower bound on the progress of the honest parties, which holds irrespective of any adversary.

Lemma 5 (Chain Growth). *Suppose that at round u of an execution E an honest party broadcasts a chain of difficulty d. Then, by round v, every honest party has received a chain of difficulty at least $d + Q(S)$, where $S = [u + \Delta, v - \Delta]$.*

Proof. If two blocks are obtained at rounds which are at distance at least Δ, then we are certain that the later block increased the accumulated difficulty. To be precise, assume $S^* \subseteq S$ is such that, for all $i, j \in S^*$, $|i - j| \geq \Delta$ and $Y_i > 0$. We argue that, by round v, every honest party has a chain of difficulty at least

$$d + \sum_{r \in S^*} Y_r \geq d + \sum_{r \in S} Q_r.$$

Observe first that every honest party will receive the chain of difficulty d by round $u + \Delta$ and so the first block obtained in S^* extends a chain of weight at least d. Next, note that if a block obtained in S^* is the head of a chain of weight at least d', then the next block in S^* extends a chain of weight at least d'.

D.3 Typical Executions

Next, we define our notion of *typical* executions. It follows the analogous definition in [13], but it is substantially simplified. 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

[18] The name "chain growth" appeared for the first time in [19], where the authors explicitly state a Chain Growth Property. In [13], the lemma is proved in a synchronous model allowing variable difficulty and varying number of parties.

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 10, 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 $H(\cdot)$ used to generate PoWs and is modeled as a random oracle. The bad events are defined as follows (recall that a block's creation time is the round where it has been successfully produced by a query to the random oracle either by the adversary or an honest party).

Definition 12. *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 later creation time.*

Given the above we are now ready to specify what is a typical execution.

Definition 13 (Typical execution). *An execution E is* typical *if the following hold.*

(a) For any set S of at least ℓ consecutive good rounds,

$$(1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta} pn(S) < Q(S) \leq D(S) < (1+\epsilon)pn(S).$$

(b) For any set J of consecutive adversarial queries and $\alpha(J) = 2(\frac{1}{\epsilon}+\frac{1}{3})\lambda/T(J)$,

$$A(J) < p|J| + \max\{\epsilon p|J|,\, \tau\alpha(J)\} \quad \textit{and} \quad B(J) < p|J| + \max\{\epsilon p|J|,\, \alpha(J)\}.$$

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

We will be interested in comparing the computational power of the adversary against that of the honest parties in a set of consecutive rounds S. However, in the bounded-delay model with delay Δ, the adversary can mute the honest parties for the final Δ rounds. The calculations summarized in the following lemma will be used repeatedly.

Lemma 6. *Consider a typical execution in a (γ, s)-respecting environment. Let $S = \{r : u \leq r \leq v\}$ be a set of at least ℓ consecutive good rounds and J the set of adversarial queries in $U = \{r : u - \Delta \leq r \leq v + \Delta\}$.*

(a) $(1+\epsilon)p|J| \leq Q(S) \leq D(U) < (1+4\epsilon)Q(S)$.

(b) $T(J)A(J) < \frac{(1-\epsilon)^3}{32(1+\delta)^2\gamma^9} \cdot \epsilon^2 m$ *or* $A(J) < (1+\epsilon)p|J|$*; similarly* $\tau T(J)B(J) < \frac{(1-\epsilon)^3}{32(1+\delta)^2\gamma^9} \cdot \epsilon^2 m$ *or* $B(J) < (1+\epsilon)p|J|$.

(c) If w is a good round such that $|w-r| \leq s$ for any $r \in S$, then $Q(S) > (1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta}|S|pn_w/\gamma$. If in addition $pn_w T(J) \geq f/2\gamma^2$, then $A(J) < (1-\delta+3\epsilon)Q(S)$.

(d) In at most $\ell + 2\Delta$ rounds can be produced at most $\frac{(1-\epsilon)^2}{8(1+\delta)\gamma^4} \cdot \epsilon^2 m$ blocks of a good chain.

The main result of this section is that almost all polynomially bounded in κ and λ executions are typical. We first prove a couple of auxiliary results.

Proposition 1. *Consider an execution E_{r-1} such that $n_r = n$, $T_r^{\max} = T^{\max}$, and $T_r^{\min} = T^{\min}$. Then,*

$$[1-f(T^{\max}, n)]pn \leq \mathbf{E}[Y_r|\mathcal{E}_{r-1} = E_{r-1}] \leq \mathbf{E}[D_r|\mathcal{E}_{r-1} = E_{r-1}] = pn,$$
$$\mathbf{E}[Y_r^2|\mathcal{E}_{r-1} = E_{r-1}] \leq pn/T^{\min}, \quad \mathbf{var}[D_r|\mathcal{E}_{r-1} = E_{r-1}] \leq pn/T^{\min}.$$

Proof. Suppose that the n honest parties at round r query for targets $T_1, \ldots, T_n$. Observe that all these variables are determined by E_{r-1}. We have

$$\begin{aligned}\mathbf{E}[Y_r|\mathcal{E}_{r-1} = E_{r-1}] &= \sum_{i\in[n]} \frac{1}{T_i} \cdot \frac{T_i}{2^\kappa} \prod_{i<j} \big[1 - f(T_j, 1)\big] \geq \sum_{i\in[n]} p \prod_{j\in[n]} \big[1 - f(T_j, 1)\big] \\ &\geq \sum_{i\in[n]} p \prod_{j\in[n]} \big[1 - f(T^{\max}, 1)\big] = \sum_{i\in[n]} p[1 - f(T^{\max}, n)] = pn[1 - f(T^{\max}, n)],\end{aligned}$$

where the third inequality holds because $f(T, n)$ is increasing in T. For the upper bound on variance,

$$\mathbf{var}[D_r|\mathcal{E}_{r-1} = E_{r-1}] \leq \sum_{i\in[n]} \frac{1}{T_i^2} \cdot \frac{T_i}{2^\kappa} = \sum_{i\in[n]} \frac{p}{T_i} \leq \frac{pn}{T^{\min}}$$

and $\mathbf{E}[Y_r^2|\mathcal{E}_{r-1} = E_{r-1}]$ is bounded alike.

The following proposition collects a few useful inequalities that hold in a (γ, s)-respecting environment.

Proposition 2. *Let U be a set of at most s consecutive rounds in a (γ, s)-respecting environment and $S \subseteq U$.*

(a) For any $n \in \{n_r : r \in U\}$, $\frac{n}{\gamma} \leq \frac{n(S)}{|S|} \leq \gamma n$.
(b) $n(U) \leq \big(1 + \frac{\gamma|U\setminus S|}{|S|}\big)n(S)$.
(c) $|S| \sum_{r\in S}(pn_r)^2 \leq \gamma(\sum_{r\in S} pn_r)^2$.

Proof. The first part is proved in [13] and is a direct consequence of the definition. For the second, note that $n(U) = n(S) + n(U \setminus S)$. By the first part we obtain that the greatest $n \in \{n_r : r \in U \setminus S\}$ is at most $\gamma n(S)/|S|$ and so $n(U \setminus S) \leq |U \setminus S|\gamma n(S)/|S|$. For the third, note that $pn/\gamma \leq pn_r \leq \gamma pn$ for any $r \in S$. The inequality follows from Theorem 12, since $A/G \leq (\gamma + 1/\gamma)/2 \leq \gamma$.

THEOREM 2. *Assuming the ITM system $(\mathcal{Z}, C)$ runs for L steps, the probability of the event "$\mathcal{E}$ is not typical" is bounded by $O(L^2)(e^{-\lambda} + 2^{-\kappa})$.*

Proof. Since the length L of the execution is fixed we will prove the stated bound for a fixed set of consecutive rounds S—or, with respect to the adversary, a fixed set of consecutive queries J—and then apply a union bound over all such sets in the length of the execution. Furthermore, we may assume $|S| \leq s$. This is because $\ell \leq s/2$ and we may partition S in parts such that each part has size between ℓ and s. We then sum over all parts to obtain the desired bound. Let us also fix an execution E_0 just before the beginning of S (or J). We will prove that the statements fail with exponentially small probability for an arbitrary E_0. Note that E_0 determines the number of parties n_0 and t_0 at the beginning of S (or J) and the target $T(J)$ associated with the first query in J.

For each round $i \in S$, define a Boolean random variable F_i equal to 1 exactly when all n_i hash values that were returned to the queries of the honest parties were above $\min\{T : f(T, n_i) \geq (1 + \delta)\gamma^2 f\}$; define $Z_i = Y_i \cdot F_{i+1} \cdots F_{i+\Delta-1}$. Let G denote the event that the rounds in S are good. Given G, for any $i \in S$, $(F_i = 1) \implies (D_i = 0)$ and so $Q_i \geq Z_i$. Thus, for any d,

$$\mathbf{Pr}\Big[G \wedge \sum_{i\in[k]} Q_i \leq d\Big] \leq \mathbf{Pr}\Big[G \wedge \sum_{i\in[k]} Z_i \leq d\Big],$$

and we may focus on the term on the right-hand side. Identify S with $\{1, \ldots, |S|\}$ and partition it with sets of the form $S_j = \{j, j + \Delta, j + 2\Delta, \ldots\}$ for $j \in \{0, 1, \ldots, \Delta - 1\}$. We will show that, for each part S_j,

$$\mathbf{Pr}\Big[G \wedge \sum_{i\in S_j} Z_i \leq (1 - \epsilon)[1 - (1 + \delta)\gamma^2 f]^\Delta p \sum_{i\in S_j} n_i\Big] \leq e^{-\lambda}.$$

Let us fix such a set $S_j = \{s_1, s_2, \ldots, s_\nu\}$, with $\nu \geq \lfloor |S|/\Delta \rfloor$, and define the event G_t as the conjunction of the events G and $t = \epsilon(1-2\gamma^2 f)^\Delta pn(S_j)$. Note that $n(S_j) \leq L$ and so t ranges over a discrete set of size at most L and we can afford a union bound over it. Thus, it is sufficient to show that for any such t,

$$\mathbf{Pr}\Big[G_t \wedge \sum_{i \in S_j} Z_i \leq [1-(1+\delta)\gamma^2 f]^\Delta p \sum_{i \in S_j} n_i - t\Big] \leq e^{-\lambda}.$$

To that end, consider the sequence of random variables

$$X_0 = 0; \;\; X_k = \sum_{i \in [k]} Z_{s_i} - \sum_{i \in [k]} \mathbf{E}[Z_{s_i} | \mathcal{E}_{s_i - 1}], \;\; k \in [\nu].$$

This is a martingale with respect to the sequence $\mathcal{E}_{s_1-1}(\mathcal{E}_0 = E_0), \ldots, \mathcal{E}_{s_\nu - 1}, \mathcal{E}$, because (recalling basic properties of conditional expectation [22]),

$$\mathbf{E}[X_k | \mathcal{E}_{s_k - 1}] = \mathbf{E}\big[Z_{s_k} - \mathbf{E}[Z_{s_k} | \mathcal{E}_{s_k - 1}] \big| \mathcal{E}_{s_k - 1}\big] + \mathbf{E}[X_{k-1} | \mathcal{E}_{s_k - 1}] = X_{k-1}.$$

Specifically, the above follows from linearity of conditional expectation and the fact that X_{k-1} is a deterministic function of $\mathcal{E}_{s_{k-1}+\Delta-1} = \mathcal{E}_{s_k - 1}$. Furthermore, given an execution E satisfying G_t,

$$\epsilon \sum_{i \in S_j} \mathbf{E}[Z_i | \mathcal{E}_{s_k - 1} = E_{s_k - 1}] \geq \epsilon \sum_{i \in S_j} [1-(1+\delta)\gamma^2 f]^\Delta p n_i = t.$$

Thus, our goal is to show $\mathbf{Pr}[-X_\nu \geq t \wedge G_t] \leq e^{-\lambda}$.

We now provide the details relevant to Theorem 10. Consider an execution E satisfying G_t and let B denote the event $\mathcal{E}_{s_k - 1} = E_{s_k - 1}$. Note that $Z_{s_k}^2 = Y_{s_k}^2 \cdot F_{s_k+1} \cdots F_{s_k+\Delta-1}$ and all these random variables are independent given B. Since $X_k - X_{k-1} = Z_{s_k} - \mathbf{E}[Z_{s_k} | \mathcal{E}_{s_k - 1}]$ and

$$Z_{s_k} - \mathbf{E}[Z_{s_k} | B] \leq \frac{1}{T_{s_k}^{\min}} = \frac{pn_{s_k}}{pn_{s_k} T_{s_k}^{\min}} \leq \frac{\gamma pn(S_j)}{pn_{s_k} T_{s_k}^{\min} |S_j|} \leq \frac{\gamma pn(S_j)}{\nu f/(2\gamma^2)} \leq \frac{2\gamma^3 t}{\epsilon(1-2\gamma^2)^\Delta f \nu} \stackrel{\text{def}}{=} b, \tag{8}$$

we see that the event G implies $X_k - X_{k-1} \leq b$. With respect to $V = \sum_k \mathbf{var}[X_k - X_{k-1} | \mathcal{E}_{s_k - 1}] \leq \sum_k \mathbf{E}[Z_{s_k}^2 | \mathcal{E}_{s_k - 1}]$, using the independence of the random variables and Proposition 1,

$$\sum_{k \in [\nu]} \mathbf{E}[Z_{s_k}^2 | B] \leq [1-(1+\delta)\gamma^2 f]^{\Delta - 1} \sum_{k \in [\nu]} \frac{(pn_{s_k})^2}{pn_{s_k} T_{s_k}^{\min}} \leq \frac{[1-(1+\delta)\gamma^2 f]^{\Delta-1}}{f/(2\gamma^2)} \cdot \sum_{k \in [\nu]} (pn_{s_k})^2.$$

Applying Proposition 2 on this bound, we see that event G_t implies

$$V \leq \frac{2\gamma^3 [1-(1+\delta)\gamma^2 f]^{\Delta-1}}{f|S_j|} \cdot \Big(\sum_{k \in [\nu]} pn_{s_k}\Big)^2 \leq \frac{2\gamma^3 t^2}{\epsilon^2 f (1-2\gamma^2 f)^{\Delta+1} \nu} \stackrel{\text{def}}{=} v. \tag{9}$$

In view of these bounds (note that $bt < \epsilon v$), by Theorem 10,

$$\mathbf{Pr}[-X_\nu \geq t \wedge G_t] \leq \exp\Big\{-\frac{t^2}{2v(1+\frac{\epsilon}{3})}\Big\} \leq \exp\Big\{-\frac{\epsilon^2 f [1-(1+\delta)\gamma^2 f]^{\Delta+1} \nu}{4\gamma^3 (1+\frac{\epsilon}{3})}\Big\} \leq e^{-\lambda},$$

where the last inequality follows from the value of ℓ (recall that $\nu \geq \ell/\Delta$ and equation (1)).

For the bound on $D(S)$ it will be convenient to work per query. Let J denote the queries in S, $\nu = |J|$, and Z_i the difficulty of any block obtained from query $i \in J$. Define the martingale sequence

$$X_0 = 0; \;\; X_k = \sum_{i \in [k]} Z_i + \sum_{i \in [k]} \mathbf{E}[Z_i | \mathcal{E}_{i-1}], \;\; k \in [\nu].$$

With similar calculations as above we obtain that G_t (with $t = \epsilon p \nu$) implies

$$X_k - X_{k-1} \le \frac{2\gamma^3 t}{\epsilon f |S|} \stackrel{\text{def}}{=} b \quad \text{and} \quad V \le \frac{2\gamma^3 t^2}{\epsilon^2 f |S|} \stackrel{\text{def}}{=} v.$$

Applying Theorem 10 we obtain

$$\mathbf{Pr}\big[X_\nu \ge t \wedge G_t\big] \le \exp\left\{-\frac{\epsilon t}{2b(1+\frac{\epsilon}{3})}\right\} \le e^{-\lambda}.$$

We next focus on part (b). For each $j \in J$, let A_j be equal to the difficulty of the block obtained with the j-th query as long as the target was at least $T(J)/\tau$; thus, $A(J) = \sum_{j\in J} A_j$. If $|J| = \nu$, identify J with $[\nu]$ and define the martingale

$$X_0 = 0; \;\; X_k = \sum_{j\in[k]} A_j - \sum_{j\in[k]} \mathbf{E}[A_j|\mathcal{E}_{j-1}], \;\; k \in [\nu].$$

For all $k \in [\nu]$ we have $X_k - X_{k-1} \le \tau/T(J)$, $\mathbf{var}[X_k - X_{k-1}|\mathcal{E}_{k-1}] \le p\tau/T(J)$, and $\mathbf{E}[A_j|\mathcal{E}_{j-1}] \le p$. We may apply Theorem 10 with $b = \tau/T(J)$, $v = bp\nu \le bt/\epsilon$, and $t = \max\{\epsilon p\nu,\, 2(\frac{1}{\epsilon} + \frac{1}{3})b\lambda\}$. We obtain

$$\mathbf{Pr}\left[\sum_{j\in J} A_j \ge p\nu + t\right] \le \exp\left\{-\frac{t}{2b(\frac{1}{3}+\frac{1}{\epsilon})}\right\} \le e^{-\lambda}.$$

For part (c), as in [13], it can be shown that an insertion or a copy imply a collision, which can be shown to occur with probability at most $\binom{L}{2}2^{-\kappa}$. Also, since there can be at most L predicted blocks, the probability a prediction occurs is at most $L^2 2^{-\kappa}$.

Proof (Proof of Lemma 6). (a) The middle inequality follows directly from the definition of the random variables. For the other two, let us assume first $|U| \le s$. With respect to the lower bound on $Q(S)$ we have

$$|J| < (1-\delta)n(U) \le (1-\delta)\Big(1 + \frac{2\gamma\Delta}{\ell}\Big)n(S) < (1-\delta)\Big(1+\frac{\epsilon^2}{2}\Big)n(S).$$

The second inequality follows from $|U| \le s$ and Proposition 2; the third from $\ell > 4\Delta\gamma^3/\epsilon^2 f$, obtained from Equation (1). On the other hand, since $|S| \ge \ell$ we may use Definition 13(a) to obtain $Q(S) > (1-2\epsilon)pn(S)$, which suffices for $\epsilon \le \delta/16$.

With respect to the upper bound on $D(U)$ we obtain similarly that

$$D(U) < (1+\epsilon)pn(U) < (1+\epsilon)\Big(1+\frac{\epsilon^2}{2}\Big)pn(S) < (1+4\epsilon)Q(S).$$

For the case $|U| > s$, we partition the sets S and U into $S_1, \ldots, S_m$ and $U_1, \ldots, U_m$, respectively, as follows. We consider any partition such that each part is at most s and at least ℓ and $S_i = U_i$ for $i = 2, 3, \ldots, m$ (this is always possible because $s = \tau m/f > \ell/2$, by Condition C1). The above derivations hold for each part and summing over all of them gives the desired inequalities.

(b) Either $\epsilon p|J| \ge \tau\alpha(J)$ and Definition 13 applies directly, or $p|J| < \tau\alpha(J)/\epsilon$ and by Equation (1) and Condition C1,

$$T(J)\cdot A(J) < \frac{2}{\epsilon^2}\cdot(1+\epsilon)\Big(1+\frac{\epsilon}{3}\Big)\tau\lambda < \frac{f\ell}{2\gamma^3} \le \frac{(1-\epsilon)^3}{32(1+\delta)^2\gamma^9}\cdot\epsilon^2 m.$$

(c) In a (γ, s)-respecting environment, $\gamma n(S) \ge n_w|S|$. Incorporating this in Definition 13 we obtain the first bound. For the second one, Using $\epsilon < 1/6$,

$$\epsilon(1-2\epsilon)pn(S) \ge \epsilon(1-2\epsilon)\frac{pn_w|S|}{\gamma} \ge \frac{\epsilon(1-2\epsilon)f\ell}{2\gamma^3 T(J)} > \tau\alpha(J).$$

As in (a), $p|J| \leq (1-\delta+\epsilon^2/2)pn(S)$. We obtain $A(J) \leq (1-\delta+\epsilon)pn(S)$ and use Condition C2.

(d) Suppose the parties query the oracle for target T during a set of rounds S of size $\ell + 2\Delta$ and so we may bound its size by the right-hand side of Condition C1. Furthermore, since the blocks belong to a good chain, $pn_r T \leq (1+\delta)\gamma^2 f$ for each $r \in S$. Putting these together, the number of such blocks that the honest parties computed are less than

$$T \cdot D(S) < (1+\epsilon)\sum_{r\in S} pn_r T \leq (1+\epsilon)\cdot \frac{(1-\epsilon)^3}{16(1+\delta)\gamma^4}\cdot \epsilon^2 m \leq \frac{(1-\epsilon)^2}{16(1+\delta)\gamma^4}\cdot \epsilon^2 m.$$

Adding the contribution of the adversary and using the bound in part (b), we obtain the desired bound.

D.4 Properties of Typical *and* Hot-Hand Executions

In this subsection we study in detail the validity of the predicates of Definition 11 over the space of typical and hot-hand executions in a (γ, s)-respecting environment. All statements in this subsection assume a (γ, s)-respecting environment for $s \geq 2(1+\delta)\gamma^2 m/f$. Furthermore, Conditions (C1-3) (Section 2) are assumed to hold for the initialization parameters n_0 and T_0.

We first handle properties that require only an execution being typical and subsequently properties that require the execution to be in addition hot hand. The first part follows [13], but the proofs use our simplified definition of a typical execution.

Our first lemma says that the adversary cannot maintain a chain by himself for too long. The reason is that the honest parties will make progress faster and his blocks will be orphaned.

Lemma 7. *In a typical execution and a (γ, s)-respecting environment*

$$\textsc{GoodRounds}(r-1) \implies \textsc{NoStaleChains}(r).$$

Proof. Suppose—towards a contradiction—$\mathcal{C} \in \mathcal{S}_r$ and has not been extended by an honest party for at least $\ell + 2\Delta$ rounds and r is the least round with this property. Let B be the last block of $\mathcal{C}$ computed by honest parties at a round w (possibly $w = 0$ and B the genesis). Set $S = [w+\Delta, r-\Delta]$ and $U = [w, r]$. Note that by our assumption $|S| \geq \ell$. Suppose that the blocks of $\mathcal{C}$ after B 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 $M = m_1 + \cdots + m_k$ and $d = m_1/T_1 + \cdots + m_k/T_k$. Our plan is to contradict the assumption that $\mathcal{C} \in \mathcal{S}_r$ by showing that all chains in $\mathcal{S}_r$ have more difficulty than $\mathcal{C}$. By Chain-Growth (Lemma 5), all the honest parties have advanced (in difficulty) during the rounds in U by $Q(S)$. Therefore, to reach a contradiction it suffices to show that $d < Q(S)$.

When $k > 2$ we may partition these M blocks into $k-1$ parts so that each part has the following properties: (1) it contains at most one target-recalculation point, and (2) it contains at least $m/2$ blocks. For each $i \in [1, k)$, let $j_i \in J$ be the index of the query during which the first block of the i-th part was computed and set $J_i = [j_i, j_{i+1})$ (Definition 13(c) assures $j_i < j_{i+1}$). We claim

$$d = \sum_{i=1}^{k} \frac{m_i}{T_i} < \sum_{i=1}^{k-1} (1+\epsilon)p|J_i| \leq (1+\epsilon)p|J| \leq Q(S).$$

For the first inequality, consider part i. We have $T_i = T(J_i)$ and—because of the first property of the partition—two possible cases for T_{i+1}: either $T_i \leq T_{i+1} \leq \tau T_i$ or $T_i/\tau \leq T_{i+1} \leq T_i$. In the first case, the difficulty of the blocks acquired in J_i is at most $B(J_i)$ and their number at most $\tau T_i B(J_i)$. In the second case, the difficulty of the blocks acquired in J_i is at most $A(J_i)$ and their number at most $T_i A(J_i)$. In either case, since the adversary acquired at least $m/2$ blocks in J_i, the desired bound follows from Lemma 6(b). The final inequality is Lemma 6(a).

If $k \leq 2$, let J denote the queries in U starting from the first adversarial query attempting to extend B. Then, $T_1 = T(J)$ and $T_2 \geq T(J)/\tau$; thus, $d \leq A(J)$. If $A(J) < (1+\epsilon)p|J|$, then $d < Q(S)$ is obtained by Lemma 6(a). Otherwise,

$$A(J) < p|J| + \tau\alpha(J) < \Big(\frac{1}{\epsilon}+1\Big)\tau\alpha(J) = 2\Big(\frac{1}{\epsilon}+1\Big)\Big(\frac{1}{\epsilon}+\frac{\epsilon}{3}\Big)\tau\lambda/T(J),$$

where we used Definition 13 and the assumption $A(J) \geq (1+\epsilon)p|J|$. Consider only the first ℓ rounds in S. In a (γ, s)-respecting environment, $pn(S) \geq pn_w\ell/\gamma$. Furthermore, since $w < r$, w is a good round and so $pn_wT_1 \geq f/(2\gamma^2)$. Putting these together, $pn(S) \geq \ell f/(2\gamma^3 T_1)$. By Definition 13 and the value of ℓ,

$$Q(S) > \frac{(1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta} f\ell}{2\gamma^3 T(J)} \geq \frac{2(1-\epsilon)(1+3\epsilon)\tau\lambda}{\epsilon^2 T(J)}.$$

Using the inequality for $A(J)$ above and $\epsilon \leq 1/8$, we arrive at our desired contradiction $d < Q(S)$.

The following lemma says that two "longest" chains cannot diverge for too long. We say below that $d \in \mathbb{R}$ is *contained* in a block B and write $d \in B$, when B extends a chain $\mathcal{C}$ and $\mathrm{diff}(\mathcal{C}) < d \leq \mathrm{diff}(\mathcal{C}B)$.

Lemma 8. *In a typical execution and a (γ, s)-respecting environment*

$$\textsc{GoodRounds}(r-1) \implies \textsc{CommonPrefix}(r).$$

Proof. Suppose $\mathrm{head}(\mathcal{C} \cap \mathcal{C}')$ was created in round v and let $u \leq v$ be the greatest round in which an honest party computed a block on $\mathcal{C} \cap \mathcal{C}'$. Let $U = (u, r]$, $S = [u+\Delta, r-\Delta]$, and let J denote the adversarial queries that correspond to the rounds in U. We claim that, if $r - v \geq \ell + 2\Delta$, then

$$2Q(S) \leq D(U) + A(J).$$

Let us first verify that this contradicts Lemma 6. First, if $|S| \geq \ell$, then by Lemma 6(a) it holds $D(U) < (1+4\epsilon)Q(S)$. Next, Lemma 7 implies that neither $\mathcal{C}$ nor $\mathcal{C}'$ is stale. This allows us to apply Lemma 6(c) and obtain $A(J) < (1-\delta+4\epsilon)Q(S)$. Putting these together with Condition C2 we obtain $D(U) + A(J) < (2-\delta+8\epsilon)Q(S) < 2Q(S)$.

Towards proving the claim, associate with each $w \in S$ such that $Q_w > 0$ an arbitrary honest block that is computed at round w for difficulty Q_w. Let $\mathcal{B}$ be the set of these blocks and note that their difficulties sum to $Q(S)$. We argue the existence of a set of blocks $\mathcal{B}'$ computed in U such that $\mathcal{B} \cap \mathcal{B}' = \emptyset$ and $\{d \in B : B \in \mathcal{B}\} \subseteq \{d \in B : B \in \mathcal{B}'\}$. This suffices, because each block in $\mathcal{B}'$ contributes either to $D(U) - Q(S)$ or to $A(J)$ and so $Q(S) \leq D(U) - Q(S) + A(J)$.

Consider a block $B \in \mathcal{B}$ extending a chain $\mathcal{C}^*$ and let $d = \mathrm{diff}(\mathcal{C}^*B)$. If $d \leq \mathrm{diff}(\mathcal{C} \cap \mathcal{C}')$ (note that $u < v$ in this case and $\mathrm{head}(\mathcal{C} \cap \mathcal{C}')$ is adversarial), let B' be the block of $\mathcal{C} \cap \mathcal{C}'$ containing d. Such a block clearly exists and was computed after round u. Furthermore, $B' \notin \mathcal{B}$, since B' was computed by the adversary. If $d > \mathrm{diff}(\mathcal{C} \cap \mathcal{C}')$, note that there is a unique $B \in \mathcal{B}$ such that $d \in B$ (recall the argument in Chain Growth Lemma 5). Since B cannot simultaneously be on $\mathcal{C}$ and $\mathcal{C}'$, there is a $B' \notin \mathcal{B}$ either on $\mathcal{C}$ or on $\mathcal{C}'$ that contains d.

Lemma 9. *In a typical execution and a (γ, s)-respecting environment*

$$\textsc{GoodRounds}(r-1) \wedge \textsc{GoodChains}(r-1) \implies \textsc{Duration}(r).$$

Proof. Assume—towards a contradiction—that $\textsc{Duration}(r)$ is false. Then, there exists a $w \leq r$ and a chain $\mathcal{C} \in \mathcal{S}_w$ with an epoch of target T and duration Λ that does not satisfy

$$\frac{1}{2(1+\delta)\gamma^2} \cdot \frac{m}{f} \leq \Lambda \leq 2(1+\delta)\gamma^2 \cdot \frac{m}{f}.$$

We consider the earliest epoch for which one of these bounds on Λ fails.

For the upper bound, Lemma 7 implies the existence of two honest blocks in this epoch computed at least $\Lambda - 2\ell - 4\Delta$ rounds apart. Let u and v be these rounds and define $S = [u, v)$. Assuming $\Lambda > 2(1+\delta)\gamma^2 m/f$, Condition C1 implies $|S| \geq 2(1+\delta)(1-\epsilon)\gamma^2 m/f$. Using this bound and our hypothesis that the rounds in S are good in Definition 13,

$$Q(S) > (1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta} \cdot \frac{f|S|}{2\gamma^2 T} \geq (1+\delta)(1-\epsilon)^3 \cdot \frac{m}{T} > \frac{m}{T},$$

where we used Condition C2 for the last inequality. This contradicts Chain Growth, since the honest parties at round v already have more than m/T difficulty on top of u.

To prove the lower bound we are going to argue that even if the honest parties and the adversary join forces they still cannot obtain m blocks. Let u and v be the target-recalculation points of the epoch. Define $S = [u, v]$ and J the set of queries available to the adversary during the rounds in S starting with the first query for target T (so that $T(J) = T$). Without loss of generality, assume S has size exactly $\lfloor \frac{1}{2(1+\delta)\gamma^2} \cdot \frac{m}{f} \rfloor > \ell$. We have

$$D(S) < (1+\epsilon)pn(S) \leq (1+\epsilon)(1+\delta) \cdot \frac{\gamma^2 f|S|}{T} \leq (1+\epsilon) \cdot \frac{m}{2T}.$$

Since the epoch is assumed to be good, $pn_u T \leq (1+\delta)\gamma f$; also, the environment is (γ, s)-respecting, thus $n(S) \leq \gamma n_u |S|$. Putting these together verifies the second inequality,while the third follows from the bound on $|S|$.

With respect to the adversary, if $\tau T B(J) < \epsilon m/4$, then the total number of blocks is less than m and we are done. Otherwise, by Lemma 6(b),

$$B(J) < (1+\epsilon)p|J| \leq (1+\epsilon)(1-\delta)pn(S) \leq (1-\delta)(1+\epsilon) \cdot \frac{m}{2T},$$

and the total count of blocks is again less than m by Condition C2.

To prove the remaining properties we need timestamps to be approximately accurate. This needs the results of the previous subection 3.1 and the following lemma gives a simple criterion that allows us to apply them.

Lemma 10. *Let $\mathcal{C} \in \mathcal{S}_r$ for a round r with consecutive recalculation points u and v. Assuming* CommonPrefix$(r-1)$ *and* NoStaleChains(r)*, there is T such that any honest query in the set $[u + \ell + 2\Delta, v - \ell - 2\Delta]$ was for target T.*

Proof. Towards a contradiction, assume an honest query in $w \in [u + \ell + 2\Delta, v - \ell - 2\Delta]$ for a target $T' \neq T$ to extend a chain $\mathcal{C}'$. Note first that by CommonPrefix(w), head$(\mathcal{C} \cap \mathcal{C}')$ contains recalculation point u. It follows that there is another recalculation point $v' \leq w$ on $\mathcal{C}'$, which implies $T' < T$. But then no honest party would adopt $\mathcal{C}$ until round v, contradicting either NoStaleChains(r) or $\mathcal{C} \in \mathcal{S}_r$.

Lemma 11. *In a typical and hot-hand execution and a (γ, s)-respecting environment,* GoodRounds$(r-1) \wedge$ GoodChains$(r-1) \implies$ MedianTime(r).

Proof. Consider a chain $\mathcal{C} \in \mathcal{S}_r$ with a target-recalculation point $v \leq r$. We will show that there are k_{med} consecutive honest blocks on $\mathcal{C}$ that were computed in $S = [v - \lfloor \epsilon^2 m/f \rfloor, v - \ell - 2\Delta)$. Not first that we may focus on the single target T of the epoch ending with recalculation point v. Indeed, using Duration(r) (Lemma 9) and Condition C1, it is not hard to see that the assumption of the previous lemma are satisfied.

By Condition C1, the size of S is at least $4^{k_{\text{med}}+4}\lambda^3/\delta^3 f$. Also, by Lemma 9 and GoodChains$(r-1)$, the target-recalculation point of the epoch was good and so T is good for S. Since the execution is hot hand, there is a winning streak $S^* \subseteq S$. By Chain Growth the chain of each honest party increased by k_{med} blocks during S^*. Since S^* is a streak, no adversarial block lies between two of these blocks. We need to argue that all these blocks will belong to the chain of every honest party. If this is not the case, then it must be that either the first block of the streak or the last one belongs to a fork. In either case, suppose this block was computed in round w. By Lemma 4 there is an interval S such that $w \in S$ and $V(S) \leq 0$. Let $S' = S^* \cap S$ and $S'' = S \setminus S'$. Note that S' is nonempty since $w \in S'$. It follows from the definition of a winning streak that $V(S') > 0$ and $V(S'') \geq 0$. Adding these inequalities we obtain a contradiction.

Lemma 12. *In a typical and hot-hand execution and a (γ, s)-respecting environment* $\textsc{GoodRounds}(r-1) \wedge \textsc{GoodChains}(r-1) \implies \textsc{GoodChains}(r)$.

Proof. Note that it is our assumption that the first round (the genesis) is a good target-recalculation point. Therefore, it suffices to show that if a recalculation point u in a chain $\mathcal{C} \in \mathcal{S}_r$ is good, then the next one at $v = u + \Lambda \leq r$ is also good. Let T be the target of the epoch starting at u and T' the target of the next one. We wish to show that $f/2\gamma \leq pn_vT' \leq (1+\delta)\gamma f$. To that end, let u' and v' be the timestamps of the two target-recalculation points and set $\Lambda' = v' - u'$.

We prove first the lower bound. If $\Lambda' \geq \gamma m/f$, then $T' \geq \gamma T$ (using $\gamma \leq \tau$) and so $pn_vT' \geq pn_uT'/\gamma \geq pn_uT \geq f/2\gamma$, because u is assumed to be a good target-recalculation point. We assume now $\Lambda' < \gamma m/f$, which implies $\Lambda' \leq (T'/T)(m/f)$. Define $S = [u, v]$, $S' = [u - \ell - 2\Delta, v + \ell + 2\Delta]$, and J the set of queries available to the adversary in S'. By Condition C1,

$$|S'| = \Lambda + 2\ell + 4\Delta \leq \Lambda + \frac{\epsilon^2 m}{16(1+\delta)\gamma^2 f} \leq (1 + \epsilon^2/8)\Lambda,$$

where the last inequality follows from the lower bound on Λ implied by Lemma 9. Lemma 11 implies $u' \geq u - \epsilon^2 m/f - \Phi$, since u' will be forced by the honest median to be at most $\epsilon^2 m/f$ rounds away from the timestamp of the median which is in turn at most Φ rounds away from u. Also, $v' \leq v + \Delta_{\mathsf{fwd}}$. Putting these together and using again the lower bound on Λ and Condition C1 as above

$$\frac{|\Lambda - \Lambda'|}{\Lambda'} \leq \frac{\epsilon^2 m/f + \Phi + \Delta_{\mathsf{fwd}}}{m/2(1+\delta)\gamma^2 f - \epsilon^2 m/f - \Phi - \Delta_{\mathsf{fwd}}} \leq \frac{9\epsilon^2/8}{1/2(1+\delta)\gamma^2 - 9\epsilon^2/8} < 6\epsilon^2,$$

where the last inequality uses Condition C2. By the last two displayed inequalities

$$|S'| \leq (1 + \epsilon^2/8)(1 + 6\epsilon^2)\Lambda' < (1+\epsilon)\Lambda'.$$

Clearly, all blocks were computed during honest queries in S or adversarial ones in J. We now bound the contribution of each.

$$B(J) < (1-\delta)(1+\epsilon)pn(S') \leq (1-\delta)(1+\epsilon)p\gamma n_v|S'| \leq (1-\delta)(1+\epsilon)^2 p\gamma n_v \Lambda'.$$

Similarly, $D(S) < (1+\epsilon)pn(S) \leq (1+\epsilon)p\gamma n_v\Lambda'$. Assuming $pn_vT' < f/2\gamma$ we obtain the contradiction

$$2\gamma pn_v\Lambda' \leq 2\gamma pn_v \cdot \frac{T'}{T} \cdot \frac{m}{f} < \frac{m}{T} \leq D(S) + B(J) < (2 + 2\epsilon - \delta)p\gamma n_v\Lambda'.$$

For the upper bound, let $S = [u + \ell + 3\Delta, v - \ell + 3\Delta]$. Note first that if $\Lambda' \leq m/\gamma f$, then $T' \leq T/\gamma$ and so $pn_vT' \leq p\gamma n_uT' \leq pn_uT \leq (1+\delta)\gamma f$, where we used that u is a good target-recalculation point. Thus, we may assume $\Lambda' > m/\gamma f$, which implies $\Lambda' \geq (T'/T)(m/f)$. Similarly to what we did above for the lower bound we may show $|S| = |\Lambda - 2\ell - 6\Delta| \geq (1-\epsilon)\Lambda'$. Assuming $pn_vT' > (1+\delta)\gamma f$, we obtain the contradiction

$$\frac{pn_v\Lambda}{(1+\delta)\gamma} \geq \frac{pn_v}{(1+\delta)\gamma} \cdot \frac{T'}{T} \cdot \frac{m}{f} > \frac{m}{T} \geq Q(S) > (1-\epsilon)[1 - (1+\delta)\gamma^2 f]^{\Delta} \cdot \frac{pn_v|S|}{\gamma}.$$

The first two inequalities have been discussed above. For the third one, note that since $\mathcal{C} \in \mathcal{S}_r$, by Lemma 7 there is a block computed by an honest party among the first and the last $\ell + 2\Delta$ rounds of the epoch; the inequality follows by Chain Growth. The next one follows from Definition 13 ($|S| \leq s$ due to Lemma 9) and so $n(S) \geq n_v|S|/\gamma$. The contradiction is a consequence of Condition C2 and the bound on $|S|$.

Lemma 13. *In a typical and hot-hand execution and a (γ, s)-respecting environment* $\textsc{GoodRounds}(r-1) \wedge \textsc{GoodChains}(r-1) \implies \textsc{GoodRounds}(r)$.

Proof. Consider any $\mathcal{C} \in \mathcal{S}_r$ and let u be its last recalculation point before r and T the associated target. Note that if r is a recalculation point, it follows directly by Lemma 12 that it is good. Otherwise, we need to show that $f/2\gamma^2 \leq pn_rT \leq (1+\delta)\gamma^2 f$. By Lemma 12, $f/2\gamma \leq pn_uT \leq (1+\delta)\gamma f$. By Lemma 9, $n_u/\gamma \leq n_r \leq \gamma n_u$. Combining these two bounds we obtain the desired inequality.

Theorem 3. *Consider a typical and hot-hand execution in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment. If the Conditions C1-3 (Section 2) are satisfied, then all predicates of Definition 11 hold.*

Proof. We only need to verify that the predicates hold for the first $\ell + 2\Delta$ rounds, assuming they hold at the first round. Note that if no epoch has been completed, all honest parties query for target T_0 and are at most γn_0. Thus, we only need to verify $\textsc{Duration}(\ell + 2\Delta)$. The lower bound of Lemma 9 holds unless $\textsc{GoodRounds}(r)$ fails for some $r < \ell + 2\Delta$, which does not happen by Lemma 13.

D.5 Blockchain Properties

For parameters that satisfy Conditions C1-3 (Section 2) we can now show that a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment enjoys common prefix, chain quality, and the new concentrated chain quality property.

Theorem 4 (Common Prefix). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the common-prefix property holds for parameter $\epsilon^2 m$.*

Proof. Suppose common prefix fails for two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ at rounds $r_1 \leq r_2$. It is not hard to see that in such a case there was a round $r \leq r_2$ and two chains $\mathcal{C}$ and $\mathcal{C}'$ in $\mathcal{S}_r$, such that each had at least k blocks after $\text{head}(\mathcal{C} \cap \mathcal{C}')$. By Lemmas 9 and 8, at least $\epsilon^2 m/2$ belong to one epoch. In view of Lemma 8, it suffices to show that that these were computed in at least $\ell + 2\Delta$ rounds. Let T be the target of these blocks and suppose the honest parties query the oracle for target T during a set of rounds S of size $\ell + 2\Delta$. By Condition C1, $|S| \leq \epsilon^2 m/16(1+\delta)\gamma^2 f$. Furthermore, by Theorem 3, $pn_rT \leq (1+\delta)\gamma^2 f$ holds for each $r \in S$. Putting these together, the number of such blocks that the honest parties computed are less than

$$T \cdot D(S) < (1+\epsilon) \sum_{r \in S} pn_rT \leq (1+\epsilon)\epsilon^2 m/16.$$

By Lemma 6 the adversary contributed less than $(1+\epsilon)\epsilon^2 m/16(1+\delta)$ blocks, for a total of less than $\epsilon^2 m/2$.

Theoem 5 (Chain Quality). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the chain-quality property holds with parameters $\ell + 2\Delta$ and $\mu = \delta - 3\epsilon$.*

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 K consecutive blocks $B_u, \dots, B_v$. Define K' as the least number of consecutive blocks $B_{u'}, \dots, B_{v'}$ that include the K 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 at some round r or is B_1 in case such block does not exist ($r = 0$), and (2) that there exists a round r' such that $B_1 \dots B_{v'} \in \mathcal{S}_{r'}$. Denote by d' the total difficulty of these K' blocks. Define $U = [r..r']$, $S = [r + \Delta..r' - \Delta]$, and J the adversarial queries in U starting with the first to obtain one of the K' blocks. Let x denote the total difficulty of all the blocks from honest parties that are included in the K blocks and—towards a contradiction—assume $x < \mu d'$. In a typical execution, all the K' blocks $\{B_j : u' \leq j \leq v'\}$ have been computed in U. But then we have the following contradiction to Lemma 6(c).

$$A(J) \geq d' - x > (1-\mu)d' \geq (1-\mu)Q(S) = (1-\delta+3\epsilon)Q(S).$$

The first two inequalities follow from the definitions of x and d' and the assumed relation between them. It is not hard to see that the last inequality follows from Chain-Growth Lemma. Finally, to verify that this is indeed a contradiction, note that if $U > (1+\delta)\gamma^2 m/f$ we may use Lemma 7 to partition U appropriately (using blocks computed by honest parties as pivot points) and apply Lemma 6(c) to each part. This is valid, since a block computed by an honest party provides both properties (1) and (2) required for K'.

Theorem 6 (Concentrated Chain Quality). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the concentrated chain quality property holds for parameters $k \leq 2k_{\text{med}}$ and $K = 4^{k_{\text{med}}+4} \cdot \frac{\lambda^3}{\delta^3 f} + 2\ell + 4\Delta$.*

Proof. In view of Lemma 9 and Condition C1, in these K rounds there is an interval S of at least $4^{k_{\text{med}}+4}\lambda^3/\delta^3 f$ rounds satisfies the criterion of Lemma 10. Since the execution is typical, the target T this lemma provides is good for S. Since the execution is hot hand, there is a wining streak of k_{med} blocks computed in these rounds. It follows exactly as in Lemma 11 that there are k_{med} consecutive honest blocks in the chain; the statement follows.

D.6 Ledger Properties

For parameters that satisfy Conditions C1-3 (Section 2) we can show that a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment enjoys consistency, liveness, and the new timekeeping property. Consistency follows directly from the common prefix property, that we showed to hold in the above circumstances.

Theorem 7 (Consistency). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than ϵm blocks deep.*

Liveness follows easily from Lemma 5 and Lemma 7.

Theorem 8 (Liveness). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, Liveness is satisfied for depth $\epsilon^2 m$ with wait-time $(4\gamma^2+1)\epsilon^2 m/f$.*

Proof. We claim that the chain $\mathcal{C}$ of any honest party has at least $\epsilon^2 m$ blocks that where computed in the last $4\epsilon^2\gamma^2 m/(1-2\epsilon)f + 4\Delta$ rounds. Indeed, $\mathcal{C}$ must have a segment that lies in a single epoch—say of target T—and was computed in a set U of at least $2\epsilon^2\gamma^2 m/(1-2\epsilon)f + 2\Delta$ consecutive rounds. If S is its subset without the first and last Δ rounds, by Chain-Growth Lemma 5, the length of this segment is at least

$$T \cdot Q(S) > (1-\epsilon)[1-(1+\delta)f]^{\Delta} \sum_{r \in S} p n_r T \geq \frac{(1-2\epsilon)f|S|}{2\gamma^2} \geq \epsilon^2 m.$$

Furthermore, if a transaction tx is included in any block computed by an honest party for the first $\ell + 2\Delta$ rounds, by Lemma 7, the chain $\mathcal{C}$ of any honest party contains tx in a block B. The total wait-time amounts to

$$\ell + 6\Delta + \frac{4\epsilon^2\gamma^2 m}{(1-2\epsilon)f} \leq \frac{\epsilon^2 m}{2(1+\delta)f} + \frac{4\epsilon^2\gamma^2 m}{(1-2\epsilon)f} \leq (4\gamma^2+1) \cdot \frac{\epsilon^2 m}{f}.$$

□

Finally, using the new concentrated chain quality property, we show that the timestamps on the blockchain are approximately accurate.

Theorem 9 (Timekeeping). *For a typical and hot-hand execution in a $(\gamma, (1+\delta)\gamma^2 m/f)$-respecting environment, the Timekeeping property holds with $\Phi_{\text{drift}} = \max\{K + \Phi, \Delta_{\mathsf{fwd}} + \ell + 2\Delta\}$.*

Proof. Consider a block B in a chain $\mathcal{C}$ computed in round r and with timestamp t. Suppose $t > r + \Delta_{\mathsf{fwd}} + \ell + 2\Delta$. Then no honest party would adopt $\mathcal{C}$ for more than $\ell + 2\Delta$ rounds and it would become stale.

We now argue $t \geq r - K - \Phi$ and consider the K rounds preceding r. By concentrated chain quality there are $2k_{\text{med}} - 1$ blocks such that the median of the timestamps of these blocks is preceded by the timestamp of an honest block. This timestamp can be at most Φ rounds away from the round it was computed in and the timestamp of B cannot be smaller than this.