

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/337093498>

Prism: Deconstructing the Blockchain to Approach Physical Limits

Conference Paper · November 2019

DOI: 10.1145/3319535.3363213

CITATIONS

76

READS

442

5 authors, including:



Vivek Kumar Bagaria

Stanford University

20 PUBLICATIONS 204 CITATIONS

SEE PROFILE

Prism: Deconstructing the Blockchain to Approach Physical Limits

Vivek Bagaria
vbagaria@stanford.edu
Stanford University

Sreeram Kannan
ksreeram@uw.edu
University of Washington at Seattle

David Tse
dntse@stanford.edu
Stanford University

Giulia Fanti
gfanti@andrew.cmu.edu
Carnegie Mellon University

Pramod Viswanath
pramodv@illinois.edu
University of Illinois at
Urbana-Champaign

ABSTRACT

The concept of a blockchain was invented by Satoshi Nakamoto to maintain a distributed ledger. In addition to its security, important performance measures of a blockchain protocol are its transaction throughput and confirmation latency. In a decentralized setting, these measures are limited by two underlying physical network attributes: communication capacity and speed-of-light propagation delay. In this work we introduce Prism, a new proof-of-work blockchain protocol, which can achieve 1) security against up to 50% adversarial hashing power; 2) optimal throughput up to the capacity C of the network; 3) confirmation latency for honest transactions proportional to the propagation delay D , with confirmation error probability exponentially small in the bandwidth-delay product CD ; 4) eventual total ordering of all transactions. Our approach to the design of this protocol is based on *deconstructing* Nakamoto’s blockchain into its basic functionalities and systematically scaling up these functionalities to approach their physical limits.

1 INTRODUCTION

In 2008, Satoshi Nakamoto invented the concept of a blockchain, a mechanism to maintain a distributed ledger in a permissionless setting. Honest nodes mine blocks on top of each other by solving Proof-of-Work (PoW) cryptographic puzzles; by following a longest chain protocol, they can come to consensus on a transaction ledger that is difficult for an adversary to alter. Since then, many other blockchain protocols have been invented.

1.1 Performance measures

The fundamental performance measures of a PoW blockchain protocol are:

- (1) the fraction β of hashing power the adversary can control without compromising system security, assuming the rest of the nodes follow protocol;
- (2) the throughput λ , number of transactions confirmed per second;
- (3) the confirmation latency, τ , in seconds, for a given probability ϵ that a confirmed transaction will be removed from the ledger in the future.

For example, Bitcoin is secure against an adversary holding up to 50% of the total network hash power ($\beta = 0.5$), has throughput λ of a few transactions per seconds and confirmation latency of the order of tens of minutes to hours. There is a tradeoff between the

confirmation latency and the confirmation error probability: the smaller the desired confirmation error probability, the longer the needed latency is in Bitcoin. For example, Nakamoto’s calculations [16] show that for $\beta = 0.3$, while it takes a latency of 6 blocks (on the average, 60 minutes) to achieve an error probability of 0.15, it takes a latency of 30 blocks (on the average, 300 minutes) to achieve an error probability of 10^{-4} .

1.2 Physical limits

Bitcoin has strong security guarantees but its throughput and latency performance are poor. In the past decade, much effort has been expended to improve the performance in these metrics. But what are the fundamental bounds that limit the performance of *any* blockchain protocol?

Blockchains are protocols that run on a distributed set of nodes connected by a physical network. As such, their performance is limited by the attributes of the underlying network. The two most important attributes are C , the communication capacity of the network, and D , the speed-of-light propagation delay across the network. Propagation delay D is measured in seconds and the capacity C is measured in transactions per second. Nodes participating in a blockchain network need to communicate information with each other to reach consensus; the capacity C and the propagation delay D limit the *rate* and *speed* at which such information can be communicated. These parameters encapsulate the effects of both fundamental network properties (e.g., hardware, topology), as well as resources consumed by the network’s relaying mechanism, such as validity checking of transactions or blocks.¹ Assuming that each transaction needs to be communicated at least once across the network, it holds that λ , the number of transactions which can be confirmed per second, is at most C , i.e.

$$\lambda < C. \quad (1)$$

One obvious constraint on the confirmation latency τ is that

$$\tau > D. \quad (2)$$

Another less obvious constraint on the confirmation latency comes from the network capacity and the reliability requirement ϵ . Indeed, if the confirmation latency is τ and the block size is B_τ transactions, then at most $C/B_\tau \cdot \tau$ mined blocks can be communicated across

¹We define confirmation formally in Section 2, but informally, we say a node ϵ -confirms a transaction if, upon successfully evaluating a *confirmation rule* under parameter ϵ , the transaction has a probability of at most ϵ of being reverted by any adversary.

the network during the confirmation period for a given transaction. These mined blocks can be interpreted as confirmation *votes* for a particular transaction during this period; i.e. votes are communicated at rate C/B_v and $C\tau/B_v$ votes are accumulated over duration τ . (The parameter B_v can be interpreted as the minimum block size to convey a vote.) On average, a fraction $\beta < 0.5$ of these blocks are adversarial, but due to the randomness in the mining process, there is a probability, exponentially small in $C\tau/B_v$, that there are more adversarial blocks than honest blocks; if this happens, confirmation cannot be guaranteed. Hence, this probability is a lower bound on the achievable confirmation error probability, i.e. $\varepsilon = \exp(-O(C\tau/B_v))$. Turning this equation around, we have the following lower bound on the latency for a given confirmation probability ε :

$$\tau = \Omega\left(\frac{B_v}{C} \cdot \log \frac{1}{\varepsilon}\right). \quad (3)$$

Comparing the two constraints, we see that if

$$\frac{CD}{B_v} \gg \log \frac{1}{\varepsilon},$$

the latency is limited by the propagation delay; otherwise, it is limited by the confirmation reliability requirement. The quantity CD/B_v is analogous to the key notion of *bandwidth-delay product* in networking (see eg. [10]); it is the number of ‘‘in-flight’’ votes in the network.

To evaluate existing blockchain systems with respect to these limits, consider a global network with communication links of capacity 20 Mbits/second and speed-of-light propagation delay D of 1 second. If we take a vote block of size 100 bytes, then the bandwidth-delay product $CD/B_v = 25000$ is very large. Hence, the confirmation latency is limited by the propagation delay of 1 seconds, but not by the confirmation reliability requirement unless it is astronomically small. Real-world blockchains operate far from these physical network limits. Bitcoin for example, has λ of the order of 10 transactions per second, τ of the order of minutes to hours, and is limited by the confirmation reliability requirement rather than the propagation delay. Ethereum has $\lambda \approx 15$ transactions per second and $\tau \approx 3$ minutes to achieve an error probability of 0.04 for $\beta = 0.3$ [4].

1.3 Main contribution

The main contribution of this work is a new blockchain protocol, Prism, which, in the face of any powerful adversary² with power $\beta < 0.5$, can *simultaneously* achieve:

- (1) **Security:** (Theorem 4.3) a total ordering of the transactions, with consistency and liveness guarantees.
- (2) **Throughput:** (Theorem 4.4) a throughput

$$\lambda = 0.9(1 - \beta)C \quad \text{transactions per second.} \quad (4)$$

- (3) **Latency:** (Theorem 4.8) confirmation of all honest transactions (without public double spends) with an expected latency of

$$\mathbb{E}[\tau] < \max\left\{c_1(\beta)D, c_2(\beta)\frac{B_v}{C} \log \frac{1}{\varepsilon}\right\} \quad \text{seconds,} \quad (5)$$

with confirmation reliability at least $1 - \varepsilon$ (Figure 1). Here, $c_1(\beta)$ and $c_2(\beta)$ are constants depending only on β

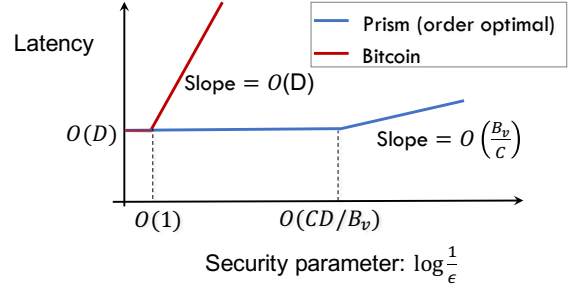


Figure 1: Confirmation latency vs. security parameter for Prism . The latency of Prism is independent of the security parameter value up to order CD/B_v and increases very slowly after that (with slope B_v/C). For Bitcoin , latency increases much more rapidly with the security parameter, with slope proportional to D . (Since $CD/B_v \gg 1$, this latter slope is much larger.)

Notice that the worst-case optimal throughput of any protocol with $1 - \beta$ fraction of hash power is $(1 - \beta)C$ transactions/second, assuming each transaction needs to be communicated across the network. Hence, Prism’s throughput is near-optimal. At the same time, Prism achieves a confirmation latency for honest transactions matching the two physical limits (2) and (3). In particular, if the desired security parameter $\log \frac{1}{\varepsilon} \ll CD/B_v$, the confirmation latency is of the order of the propagation delay and *independent* of $\log 1/\varepsilon$. Put another way, one can achieve latency close to the propagation delay with a confirmation error probability exponentially small in the bandwidth-delay product CD/B_v . Note that the latency is worst-case over all adversarial strategies but averaged over the randomness in the mining process.

To the best of our knowledge, no other existing PoW protocol has guaranteed performance which can match that of Prism. Two novel ideas which enable this performance are 1) a *total decoupling* of transaction proposing, validation and confirmation functionalities in the blockchain, allowing performance scaling; 2) the concept of confirming a *list* of possible ledgers rather than a unique ledger, enabling honest non-double-spend transactions to be confirmed quickly³.

1.4 Performance of existing PoW protocols

High forking protocols. Increasing the mining rate in Bitcoin can decrease latency and improve throughput, however, this comes at the expense of decreased security [25]. Thus, unlike Prism, the throughput and latency of Bitcoin is *security-limited* rather than *communication-limited*. To increase the mining rate while maintaining security, one line of work (GHOST [25], Inclusive [14], Spectre [23], Phantom [24], Conflux [15]) in the literature has used more complex fork choice rules and added reference links to convert the blocktree into a directed acyclic graph (DAG). This allows blocks to be voted on by blocks that are not necessarily its descendants.

²The powerful adversary will be precisely defined in the formal model.

³This idea was inspired by the concept of list decoding from information theory.

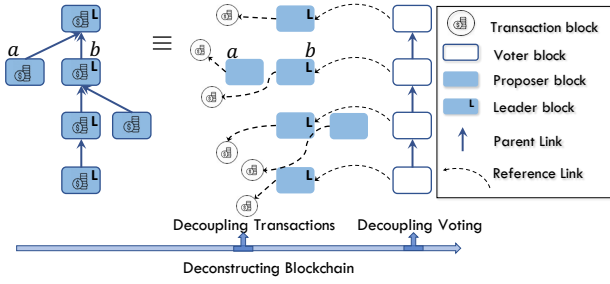


Figure 2: Deconstructing the blockchain into transaction blocks, partially ordered proposal blocks arranged by level, and voter blocks organized in a voter tree. The main chain is selected through voter blocks, which vote among the proposal blocks at each level to select a leader block. For example, at level 3, block b is elected the leader over block a .

While GHOST remains secure at low mining rates[11], there is a balancing attack by the adversary [12, 17], which severely limits the security at high mining rates. Thus, like Bitcoin, the throughput of GHOST is security-limited. The other protocols Inclusive and Conflux that rely on GHOST inherit this drawback. While Spectre and Phantom improve latency and throughput, Spectre cannot provide a total order on all transactions (required for smart contracts) and Phantom does not yet have a formal proof of security.

Decoupled consensus. Protocols such as BitcoinNG [7] decouple transaction proposal and leader election (which are coupled together in Bitcoin). BitcoinNG elects a single leader to propose many transaction blocks till the next leader is elected by PoW. While this achieves high throughput, the latency cannot be reduced using this approach. Furthermore, BitcoinNG is vulnerable to bribery or DDoS attacks, whereby an adversary can corrupt a leader after learning its identity (unlike Bitcoin). Subchains [22] and weak blocks [3, 26] both employ blocks with lower hash threshold (“weak blocks”) along with regular blocks in an attempt to scale throughput. However, since weak blocks are required to form a chain, it does not achieve the optimal throughput.

Hybrid blockchain-BFT consensus. Several protocols combine ideas from Byzantine fault tolerant (BFT) based consensus into a PoW setting [1, 13, 20, 21]. ByzCoin [13] and its predecessor DiscCoin [6] attempt to address the latency shortcoming of BitcoinNG but is proven in a later paper [20] to be insecure when the adversarial fraction $\beta > 0.25$. *Hybrid consensus* uses a combination of proof-of-work based committee selection with Byzantine fault tolerance (BFT) consensus [20]. However, this protocol is secure only till $\beta = 0.33$. While the protocol latency is responsive, i.e., it decreases with network delay linearly, for a known network delay, it has similar non-optimal dependence on ϵ as Bitcoin.

A closely-related protocol called Thunderella [21] achieves very low latency under optimistic conditions, i.e., when the leader is honest and $\beta < 0.25$. However even when β is very small, a dishonest leader can keep delaying transactions to the Bitcoin latency (since such delaying behavior is detected by a slow PoW blockchain).

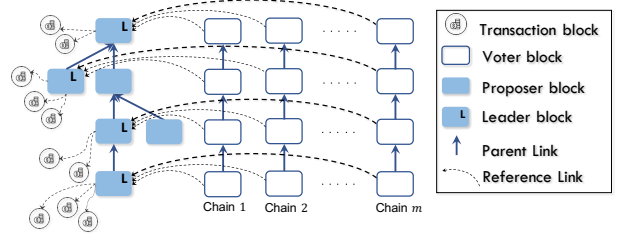


Figure 3: Prism. Throughput, latency and reliability are scaled to the physical limits by increasing the number of transaction blocks and the number of parallel voting chains.

1.5 Our Approach

Increasing the mining rate is critical to improving the throughput and latency of blockchain protocols. The challenges facing the DAG approaches arise from the fact that the DAG is *unstructured*, due to the excessive random forking when the mining rate is increased. In contrast, Prism is based on a *structured* DAG created by cryptographic sortition of the mined blocks into different types of different functionalities and scaling these functionalities separately.

Deconstruction. We start by deconstructing the basic blockchain structure into its atomic functionalities, illustrated in Figure 2. The selection of a main chain in a blockchain protocol (e.g., the longest chain in Bitcoin) can be viewed as electing a leader block among all the blocks at each level of the blocktree, where the level of a block is defined as its distance (in number of blocks) from the genesis block. Blocks in a blockchain then serve three purposes: they stand for election to be leaders, they add transactions to the main chain, and they vote for ancestor blocks through parent link relationships. We explicitly separate these three functionalities by representing the blocktree in a conceptually equivalent form (Figure 3). In this representation, blocks are divided into three types: proposer blocks, transaction blocks and voter blocks. The voter blocks vote for transactions indirectly by voting for proposer blocks, which in turn link to transaction blocks. Proposer blocks are grouped according to their level in the original blocktree, and each voter block votes among the proposer blocks at the same level to select a leader block among them. The elected leader blocks can then bring in the transactions to form the final ledger. The valid voter blocks are the ones in the longest chain of the voter tree, and this longest chain maintains the security of the whole system.

Scaling. This alternative representation of the traditional blockchain, although seemingly more complex than the original blockchain representation, provides a natural path for scaling performance to approach physical limits (Figure 3). To increase the transaction throughput, one can simply increase the number of transaction blocks that a proposer block points to without compromising the security of the blockchain. This number is limited only by the physical capacity of the underlying communication network. To provide fast confirmation, one can increase the number of parallel voting trees, voting on the proposal blocks in parallel to increase the voting rate, until reaching the physical limit of confirming with speed-of-light latency and extremely high reliability. Note that even though the overall block generation rate has increased tremendously, the

number of proposal blocks per level remains small and manageable, and the voting blocks are organized into many separate voting chains with low block mining rate per chain and hence little forking. The overall structure, comprising of the different types of blocks and the links between them, is a structured DAG.

Sortition. The sortition of blocks into the three types of blocks, and further into blocks of different voting trees, can be accomplished by using the random hash value when a block is successfully mined. This sortition splits the adversary power equally across the structures and does not allow it to focus its power to attack specific structures. This sortition is similar to the 2-for-1 PoW technique used in [9], which is also used in Fruitchains [19] for the purpose of providing fairness in rewards. In fact, the principle of *decoupling* functionalities of the blockchain, central to our approach, has already been applied in Fruitchains, as well as other works such as BitcoinNG. The focus of these works is only on decoupling the transactions-carrying functionality. In our work, we broaden this principle to decouple *all* functionalities. **Concurrent work.** We were made aware of two independent but related works [8, 27] which appeared after we posted this work online. [8] proposes two protocols, one achieves high throughput $O(C)$ but Bitcoin latency, and the other achieves low latency $O(1/\sqrt{C})$ but low throughput $O(1)$. In contrast, Prism achieves *simultaneously* high throughput $O(C)$ and even lower latency $O(1/C)$. Although [8] also uses the concept of multiple chains, the key difference with Prism is that there is *no* decoupling: the blocks in each chain both carry transactions and vote. Thus, either different transactions are put on the different chains to increase throughput, but the voting rate is low and hence the latency is poor, or the same transaction is repeated across all the chains to increase the voting rate, but the throughput is poor. In contrast, Prism decouples blocks into transaction blocks and voter blocks, tied together through proposer blocks, and allocate a fraction of the network capacity to each to deliver both low latency and high throughput. The protocol in [27] is similar to first one in [8], achieving high throughput but only Bitcoin latency.

1.6 Outline of paper

Section 2 presents our model. It is a combination of the synchronous model used in [9] and a network model that ties the blockchain parameters to physical parameters of the underlying network. In Section 3, we give a pseudocode description of Prism. The analysis of the security, throughput and latency of Prism is presented in Section 4, with details of the proofs in the appendices. Section 5 contains simulation results.

2 MODEL

We consider a synchronous, round-based network model similar to that of Garay *et al.* [9]. We define a blockchain protocol as a pair (Π, g) , where Π is an algorithm that maintains a blockchain data structure C consisting of a set of *blocks*. The function $g(\text{tx}, C)$ encodes a *ledger inclusion rule*; it takes in a transaction tx and a blockchain C , and outputs $g(\text{tx}, C) = 1$ if tx is contained in the ledger defined by blockchain C and 0 otherwise. For example, in Bitcoin, $g(\text{tx}, C) = 1$ iff tx appears in any block on the longest chain. If there are multiple longest chains, g can resolve ties deterministically, e.g., by taking the chain with the smallest hash value.

The blockchain protocol proceeds in rounds of Δ seconds each. Letting κ denote a security parameter, the *environment* $\mathcal{Z}(1^\kappa)$ captures all aspects external to the protocol itself, such as inputs to the protocol (i.e., new transactions) or interaction with outputs.

Let \mathcal{N} denote the set of participating nodes. The set of *honest* nodes $\mathcal{H} \subset \mathcal{N}$ strictly follow the blockchain protocol (Π, f) . *Corrupt* nodes $\mathcal{N} \setminus \mathcal{H}$ are collectively controlled by an adversarial party \mathcal{A} . Both honest and corrupt nodes interact with a random function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ through an oracle $H(x)$, which outputs $H(x)$. In each round, each node $n \in \mathcal{N}$ is allowed to query the oracle $H(\cdot)$ at most q times. The adversary’s corrupt nodes are collectively allowed up to $\beta q |\mathcal{N}|$ sequential queries to oracle $H(\cdot)$, where $\beta < 0.5$ denotes the fraction of adversarial hash power, i.e., $1 - \frac{|\mathcal{H}|}{|\mathcal{N}|} = \beta$.⁴ Like [9], the environment is not allowed to access the oracle. These restrictions model the limited hash rate in the system.

In an execution of the blockchain protocol, the environment \mathcal{Z} first initializes all nodes as either honest or corrupt; like [9], once the nodes are initialized, the environment can adaptively change the set \mathcal{H} between rounds, as long as the adversary’s total hash power remains bounded by β . Thereafter, the protocol proceeds in rounds. In each round, the environment first delivers inputs to the appropriate nodes (e.g., new transactions), and the adversary delivers any messages to be delivered in the current round. Here, delivery means that the message appears on the recipient node’s input tape. Nodes incorporate the inputs and any messages (e.g., new blocks) into their local blockchain data structure according to protocol Π . The nodes then access the random oracle $H(\cdot)$ as many times as their hash power allocation allows. Hence, in each round, users call the oracle $H(\cdot)$ with different nonces s in an attempt to find a valid proof of work. If an oracle call produces a proof of work, then the node can deliver a new block to the environment. Note that the computational constraints on calling oracle $H(\cdot)$ include block validation. Since each block only needs to be validated once, validation represents a small fraction of computational demands.

Since each node is allowed a finite number of calls to $H(x)$ in each round, the number of blocks mined per round is a Binomial random variable. To simplify the analysis, we consider a limit of our model as the number of nodes $|\mathcal{N}| \rightarrow \infty$. As $|\mathcal{N}|$ grows, the proof-of-work threshold adjusts such that the expected number of blocks mined per round remains constant. Hence, by the Poisson limit theorem, the number of voter blocks mined per round converges to a Poisson random variable.

All messages broadcast to the environment are delivered by the adversary. The adversary has various capabilities and restrictions. (1) Any message broadcast by an honest node in the previous round must be delivered by the adversary at the beginning of the current round to all remaining honest nodes. However, during delivery, the adversary can present these messages to each honest node in whatever order it chooses. (2) The adversary cannot forge or alter any message sent by an honest node. (3) The adversary can control the actions of corrupt nodes. For example, the adversary can choose how corrupt nodes allocate their hash power, decide block content, and release mined blocks. Notably, although honest blocks publish mined blocks immediately, the adversary may choose to keep blocks

⁴ β for bad. Like [9], we have assumed all nodes have the same hash power, but this model can easily be generalized to arbitrary hash power distributions.

they mined private and release in future round. (4) The adversary can deliver corrupt nodes' messages to some honest nodes in one round, and the remaining honest nodes in the next round. We consider a "rushing" adversary that observes the honest nodes' actions before taking its own action for a given round. Notice that we do not model rational users who are not necessarily adversarial but nevertheless may have incentives to deviate from protocol.

Physical Network Constraints. To connect to the physical parameters of the network, we assume a simple network model. Let B be the size of a block, in units of number of transactions. The network delay Δ (in seconds) is given by:

$$\Delta = \frac{B}{C} + D \quad (6)$$

i.e. there is a processing delay of B/C followed by a propagation delay of D seconds. This is the same model used in [25], based on empirical data in [5], as well in [22]. Notice that the network delay Δ is by definition equal to the duration of a single round.

In practice, networks cannot transport an infinite number of messages at once. We model this by allowing the environment to transport only a finite volume of messages per round. This volume is parametrized by the *network capacity* C , measured in units of transactions per second. Hence, during each round, the environment can process a message volume equivalent to at most ΔC transactions. This puts a constraint on the number of blocks mined per unit time in any protocol. This *stability constraint* differentiates our model from prior work, which has traditionally assumed infinite network capacity; in particular, this gives us a foothold for quantifying physical limits on throughput and latency.

For simplicity, we assume that the dissemination of new transactions consumes no bandwidth. Instead, the cost of communicating transaction messages is captured when the environment transmits blocks carrying transactions. In other words, we assume that the cost of transmitting transactions is counted only once.

Metrics. We let random variable $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$ denote the joint view of all parties over all rounds; here we have suppressed the dependency on security parameter κ . The randomness is defined over the choice of function $H(\cdot)$, as well as any randomness in the adversary \mathcal{A} or environment \mathcal{Z} . Our goal is to reason about the joint view for all possible adversaries \mathcal{A} and environments \mathcal{Z} . In particular, we want to study the evolution of C_i^r , or the blockchain of each honest node $i \in \mathcal{H}$ during round r . Following the Bitcoin backbone protocol model [9], we consider protocols that execute for a finite execution horizon r_{\max} , polynomial in κ . Our primary concern will be the efficiency of *confirming* transactions.

DEFINITION 2.1. We say a transaction tx is $(\varepsilon, \mathcal{A}, \mathcal{Z}, r_0, \kappa)$ -cleared iff under an adversary \mathcal{A} , environment \mathcal{Z} , and security parameter κ ,

$$\mathbb{P}_{\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}} \left(\bigcap_{\substack{r \in \{r_0, \dots, r_{\max}\} \\ i \in \mathcal{H}}} \{g(\text{tx}, C_i^r) = b\} \right) \geq 1 - \varepsilon - \text{negl}(\kappa),$$

where $b \in \{0, 1\}$; $b = 1$ corresponds to confirming the transactions and $b = 0$ corresponds to rejecting the transaction.

That is, a transaction is considered confirmed (resp. rejected) if all honest party will include (resp. exclude) it from the ledger with

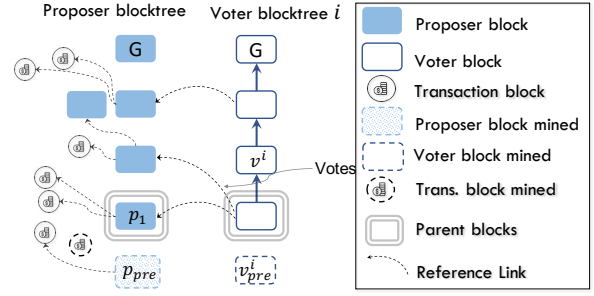


Figure 4: Snapshot of a miner's blocktree: The previously mined blocks have solid boundary whereas blocks which are being mined have dotted-boundary. A miner simultaneously mines on p_1 , parent on proposer blocktree, v^i , parent on voter block blocktree $i (v_i \in [m])$.

probability more than ε plus a term negligible in κ resulting from hash collisions, which we ignore in our analysis. We suppress the notation κ from here on.

Our objective is to optimize two properties of a blockchain protocol: the throughput and latency of confirming transactions. We let $|S|$ denote the number of elements in set S . We let \mathcal{T} denote the set of all transactions generated during the execution horizon, and \mathcal{T}^r denote all transactions delivered up to and including round r .

DEFINITION 2.2 (THROUGHPUT). We say a blockchain protocol Π supports a throughput of λ transactions per round if there exists U_ε , linear in $\log(1/\varepsilon)$, such that for all environments \mathcal{Z} that produce at most λ transactions per round, and for $\forall r \in [1, r_{\max}]$,

$$\max_{\mathcal{A}} |\{\text{tx} \in \mathcal{T}^r : \text{tx is not } (\varepsilon, \mathcal{A}, \mathcal{Z}, r)\text{-cleared}\}| < \lambda U_\varepsilon. \quad (7)$$

The system throughput is the largest throughput that a blockchain protocol can support.

Notice that although $|\mathcal{T}^r|$ grows with r , the right-hand side of (7) is constant in r ; this implies that the system throughput λ is the expected *rate* at which we can clear transactions maintaining a bounded transaction queue, taken worst-case over adversary \mathcal{A} and environments \mathcal{Z} producing at most $\lambda\Delta$ transactions per round.

DEFINITION 2.3 (LATENCY). For a transaction tx , let $r(\text{tx})$ denote the round in which the transaction was first introduced by the environment, and let random variable $R_\varepsilon(\text{tx})$ denote the smallest round r for which tx is $(\varepsilon, \mathcal{A}, \mathcal{Z}, r)$ -cleared. The expected ε -latency of transaction tx is defined as:

$$\tau_\varepsilon(\text{tx}) \triangleq \max_{\mathcal{Z}, \mathcal{A}} E_{\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}} [R_\varepsilon(\text{tx}) - r(\text{tx})] \quad (8)$$

Note that if all transactions have finite ε -latency, it implies that the blockchain has both consistency and liveness properties.

3 PROTOCOL DESCRIPTION

We first describe the content and roles of three types of blocks in the Prism(Π, g) blockchain. We then present Algorithm 1, which defines the protocol Π and the blockchain data structure C . We then define the *ledger inclusion rule*, g , in Algorithm 2. Due to space constraints, all pseudocode for these algorithms can be found in

Appendix A. Prism’s blockchain data structure, C , has one proposer blocktree and m voter blocktrees, as shown in Figure 3. We use these different blocktrees to maintain three distinct types of blocks:

Proposer blocks: Proposer blocks represent the skeleton of the Prism blockchain and are mined on the proposer blocktree according to the longest-chain rule. The *level* of a proposer block is defined as its distance from the proposer genesis block. The blocktree structure is only utilized in our protocol as a proof of level of a given proposal block. To construct the ledger, our protocol selects proposal block sequences where one block is chosen at each level. Proposer blocks can *refer* to transaction blocks and other proposer blocks by including pointers to referred blocks in their payload. For example, in Fig 4, the proposer blocktree has two proposer blocks mined at level 1, and one proposer block mined at levels 2 and 3, and they point to five transaction blocks in total.

Voter blocks: Voter blocks are mined on m separate voter blocktrees, each with its own genesis block, according to the longest chain rule. We say a voter block *votes* on a proposer block B if it includes a pointer to B in its payload. Note that unlike many BFT consensus protocols, a malicious miner in Prism cannot equivocate when voting because voter blocks are sealed by proof of work. Even if a miner mines conflicting voter blocks and tries to send them to disjoint sets of honest users, all users will receive both blocks within one round. Each longest chain from each voter blocktree can cast at most one vote for each level in the proposer blocktree. More precisely, a voter block votes on all levels in the proposer tree that are unvoted by the voter block’s ancestors. Therefore, the voter trees collectively cast at most m votes on a given level of the proposer blocktree. Fig. 3 shows voter blocktree i and its votes (dotted arrows) on each level of the proposer blocktree. For each level ℓ on the proposer blocktree, the block with the highest number of votes is defined as the *leader* block of level ℓ .

Transaction blocks: Transaction blocks contain transactions and are mined on the proposer blocktree as in Fig. 3. Although transaction blocks are not considered part of the proposer blocktree, each transaction block has a proposer block as its parent.

The process by which a transaction is included in the ledger is as follows: (1) the transaction is included in a transaction block B_T . (2) B_T is referred by a proposer block B_P . (3) Proposer block B_P is confirmed, either directly (by becoming a leader) or indirectly (e.g., by being referred by a leader).

3.1 Protocol II

Algorithm 1 presents Prism’s protocol II. The protocol begins with a trusted setup, in which the environment generates genesis blocks for the proposer blocktree and each of the m voter blocktrees. Once the trusted setup completes, the protocol enters the mining loop.

Whereas Bitcoin miners mine on a single blocktree, Prism miners simultaneously mine one proposer block, one transaction block, and m voter blocks, each with its own parent and content. This simultaneous mining happens via cryptographic sortition. Roughly, a miner first generates a “superblock” that contains enough information for all $m + 2$ blocks simultaneously. It then tries different nonce values; upon mining a block, the output of the hash is deterministically mapped to either a voter block (in one of the m trees), a transaction block, or a proposer block (lines 41-47 in Algorithm

1). After sortition, the miner discards unnecessary information and releases the block to the environment.

More precisely, while mining, each miner maintains outstanding content for each of the $m + 2$ possible mined blocks. In Bitcoin, this content would be the transaction memory pool, but since Prism has multiple types of blocks, each miner stores different content for each block type. For transaction blocks, the content consists of all transactions that have not yet been included in a transaction block. For proposer blocks, the content is a list of transaction blocks and proposer blocks that have not been referred by any other proposer block. For voter blocks in the i th voter tree, the content is a list of proposer blocks at each level in the proposer blocktree that has not yet received a vote in the longest chain of the i th voter tree. If a miner observes multiple proposer blocks at the same level, it always votes on the first one it received. For example, in Figure 4, voter block v_{new}^i votes on one proposer block on levels 3 and 4 because its ancestors have voted on level 1 and 2.

Upon collecting this content, the miner generates a block. Instead of naively including all the $m + 2$ parents⁵ and content hashes in the block, Prism’s header contains a) the Merkle root of a tree with $m + 2$ parent blocks, b) the Merkle root of a tree with $m + 2$ contents, and c) a nonce. Once a valid nonce is found, the block is sortitioned into a proposer block, a transaction block, or a voter block on one of the m voter trees. The mined, sortitioned block consists of the header, the appropriate parent and content, and their respective Merkle proofs. For instance, if the mined block is a proposer block, it would contain only the proposer parent reference, proposer content, and appropriate Merkle proofs; it would *not* store transactions or votes.

While mining, nodes may receive blocks from the network, which are processed in much the same way as Bitcoin. Upon receiving a new block, the miner first checks validity. A block B is valid if it satisfies the PoW inequality and the miner has all the blocks (directly or indirectly) referred by B . If the miner lacks some referred blocks, it requests them from the network. Upon receiving a valid transaction block B , the miner removes the transactions in B from its transaction pool and adds B to the unreferred transaction block pool. Upon receiving a valid voter block, the miner updates the longest chain if needed, and updates the vote counts accordingly. Upon receiving a valid proposer block B at a level ℓ higher than the previous highest level, the miner makes B the new parent proposer block, and updates all m voter trees to vote on B at level ℓ .

3.2 Ledger confirmation rule g

As defined before, the proposer block with the most votes on level ℓ is defined as the *leader block* of level ℓ . The leader block for a fixed level ℓ can initially fluctuate when the voter blocktrees start voting on level ℓ . However, as the voter blocktrees grow, these votes on level ℓ are cemented deeper into their respective voter blocktrees and the leader fluctuation ceases and thus we can confirm the leader block at level w.h.p. The sequence of leader blocks for each level of the proposer blocktree is defined as the *leader sequence*.

Confirmation and Ordering: A set of transactions can often be individually confirmed before being ordered among themselves. For this reason, confirming transactions is easier than ordering

⁵Proposer and tx block share the same parent and are included twice for simplicity.

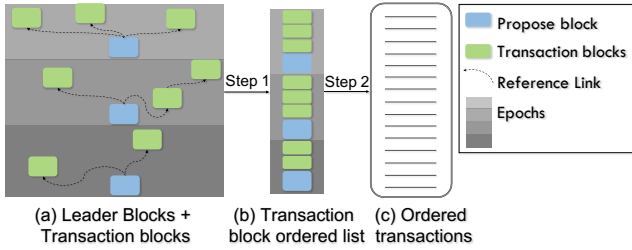


Figure 5: BUILDLEDGER(): The proposer blocks for a given proposer block sequence are blue, and the referenced transaction blocks are green. Each shade of gray region is all the tx blocks referred by the proposer block.

the transactions. For example, consider the following two transactions a) Alice pays Bob 10\$, and b) Carol pays Drake 10\$. Both these transactions can be individually confirmed without deciding which transaction occurred first. In Bitcoin, transactions are simultaneously confirmed and ordered; however, in Prism, transactions can be confirmed before being ordered. The procedure `IS_TX_CONFIRMED()` in Algorithm 2 defines the transaction confirmation rule g and the procedure `GET_ORDERED_CONFIRMED_TXS()` defines the rule for ordering the confirmed transactions. Both these procedures use `BUILDLEDGER()` which is described next.

`BUILDLEDGER()`: Given a proposer block sequence from levels 1 to ℓ , $\{p_1, \dots, p_\ell\}$, represented by blue blocks in Fig. 5(a). Let L_{p_i} , represented by green blocks in the grey shaded area in Fig. 5(a), be an ordered list of all the transaction blocks directly or indirectly referred by block p_i . Note that a transaction block t is indirectly referred by proposer block p_i if p_i includes a reference link to another proposer block p' that directly refers t . Since honest proposer blocks link to any unreferenced transaction blocks and proposer blocks, this ensures that the transaction blocks not referred by the proposer leader sequence are also included in the ledger. Let $\{L_{p_1}, \dots, L_{p_\ell}\}$ be the *transaction block list* of sequence $\{p_1, \dots, p_\ell\}$ as shown in Fig. 5(b). The procedure then expands this transaction-block list and remove all the duplicate and double-spent transactions to output *ordered-transaction list* as shown in Fig. 5(c).

`IS_TX_CONFIRMED()`: While confirming a leader block can take some time⁶, we quickly narrow down a set of proposer blocks, defined as *proposer-set*, which is guaranteed to contain the leader block for that level. The proposer-set is realized using Def. (4.5). This procedure first gets all the votes from the voter trees and then gets the proposer-set for each level from the genesis to the last level for which the proposer-set can be realized (lines:5-9). It then takes the outer product of these proposer-sets and enumerates many proposer block sequences (line:11). Note that by design, one of these sequences will be the leader block sequence in the future. It then builds a ledger for each proposer block sequence and confirms the transaction if it is present in *all* of the ledgers (lines:12-14).

`GET_ORDERED_CONFIRMED_TXS()`: First obtain a leader block for each level on proposer blocktree from genesis up until the level which has a confirmed leader block (line:42). Then return the ledger built from these leader blocks.

⁶In absence of an active attack, it will be fast, as described in Section 4.

4 ANALYSIS

In this section, we analyze three aspects of Prism: security, throughput, and latency. Before listing the formal guarantees satisfied by Prism, we first describe at an intuitive level why Prism is able to achieve good latency without sacrificing security.

4.1 Intuition and Sketch of Proofs

In the longest-chain protocol, for a fixed block size and network, the maximum tolerable adversarial hash power β is governed by the block production rate; the faster one produces blocks, the smaller the tolerable β [9, 18]. In Prism, we need to be able to tolerate β adversarial hash power in each of the voter trees and in the proposer tree. Hence, following the observations of [9, 18] each of these trees individually must operate at the same rate as a single longest-chain blocktree in Bitcoin in order to be secure.

The security of Prism is provided by the voter trees; a proposer block is confirmed by votes which are on the longest chains of these voter trees. Consider a conservative confirmation policy for Prism, where we wait for each vote on each voter tree to reach a confirmation reliability $1 - \epsilon$ before counting it. This would require us to wait for each vote to reach a depth of $k(\epsilon)$ in its respective tree, where $k(\epsilon)$ denotes the confirmation depth for reliability $1 - \epsilon$. This conservative confirmation rule immediately implies that Prism has the same security guarantee as that of each of the voter tree, i.e. that of Bitcoin. However, this rule has as poor a latency as Bitcoin's. For example, for $\epsilon = 10^{-3}$ and the tolerable adversary power $\beta = 0.3$, the vote has to be 24 blocks deep [16]. With a more intelligent transaction confirmation rule, we can do far better. The key insight is that even though each vote individually stabilizes at the same rate at Bitcoin (i.e., slowly), the *aggregate* opinion can converge much faster because there are many voter trees.

4.1.1 Case 1: Isolated Proposer Block

Consider first the situation when a transaction block TB is referred to by a honest proposer block H which is currently isolated at its level, i.e. no other public proposal block exists at the same level for a certain fixed number of rounds. See Figure 6(a). This case is quite common since the mining rate of the proposer blocks is chosen such that there is little forking in the proposer tree. Block H will start collecting votes, each of which is on the longest chain of its respective voter tree. Over time, each of these votes will become deeper in its voter chain. An attack by the adversary is to mine a private proposal block A at the same level, and on each of the voter trees fork off and mine a private alternate chain and send its vote to the block A . After leader block H is confirmed, the adversary continues to mine on each of the voter alternate chains to attempt to overtake the public longest chain and shift the vote from H to A . If the adversary can thereby get more votes on A than on H , then its attack is successful.

This can be viewed as the m -chain analog to Nakamoto's private attack on a single chain [16], where instead of having one race between the honest chain and the attack chain we have m such races. In fact, Nakamoto's calculations on the success probability of an attack on a single chain can help us determine how deep we need to wait for the votes to become to confirm the proposer block H . At tolerable adversary power $\beta = 0.3$, the reversal probability in a

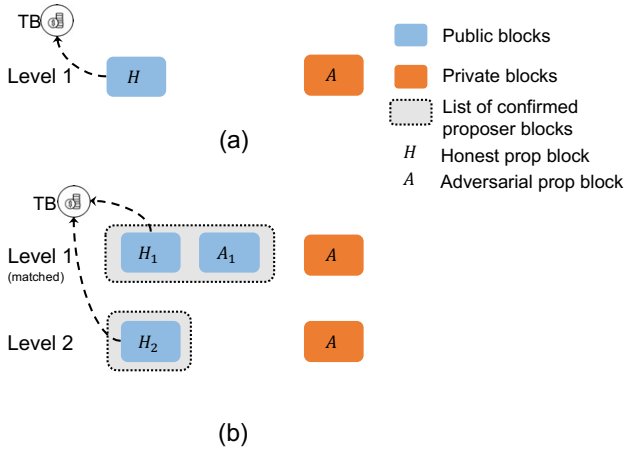


Figure 6: (a) Transaction block is referred to by an isolated honest proposer block. (b) Transaction block is referred to by a non-isolated proposer block but on the next level there is an isolated proposer block. Note that the link from H_2 to TB is implicit; since H_2 is honest, it refers to all unreferenced transaction and proposer blocks, i.e., H_1 and A_1 . Since H_1 refers TB, H_2 implicitly does too (Section 3.2)

single chain is 0.45 when a block is 2-deep [16]. With $m = 1000$ voter chains and each vote being 2-deep, the expected number of chains that can be reversed by the adversary is 450. The probability that the adversary got lucky and can reverse more than half the votes, i.e. 500, is about 10^{-3} . Hence to achieve $\varepsilon = 10^{-3}$, we only need to wait for 1000 votes each 2-deep. This incurs much shorter latency than the 24 block depth needed for *each* vote to be reversed with probability 10^{-3} . This reduction in latency is conceptually similar to averaging many unreliable classifiers to form a strong aggregate classifier: the more voter chains there are, the less certainty of permanence each individual vote needs to be, thereby reducing confirmation time. This gain comes without sacrificing security: each voter chain is operating slowly enough to tolerate β adversarial hash power.

Just like Nakamoto’s private attack, the attack considered here is a particular attack. Our formal security analysis, sketched in Section 4.1.3, consider all possible attacks in the model. In particular, the attacker can correlate its actions on the different voter chains. However, the confirmation latency behaves similarly to the latency under this attack.

4.1.2 Case 2: Non-isolated Proposer Block

Consider now the case when the the transaction block TB is referred to by a honest proposal block H_1 which is not isolated at its level, i.e. H_1 is matched by an adversarial public proposer block A_1 (the competing proposer block could also be honest). This matching could persist for L levels until reaching a level when there is an isolated honest proposal block. See Figure 6(b) for the special case of $L = 1$. Let us separately consider the life cycle of an honest transaction vs. a double-spent one.

Honest Transaction: A naive approach for confirming TB would be to wait until we can definitively confirm H_1 or A_1 . However, this may be slow because of adversarial attacks that try to balance votes. A key insight is that for honest (non-double-spent) transactions, we do not need to know *which* of H_1 and A_1 is confirmed—only that one of them will be confirmed. This weaker form of *list confirmation* works because if A_1 eventually gets confirmed, a later honest proposer block can still refer to H_1 and include TB (Section 3.2). To confirm an honest transaction at level i , we need two events: (1) list confirmation of all levels up to i ; (2) an isolated honest proposer at level i . Once we have list-confirmed a set of proposer blocks at level i referring TB (e.g., either H_1 or A_1 will be the leader), we know that no other block can be the leader at that level. However, list confirmation alone is not enough for honest transaction confirmation if the transaction is not present in all ledgers. In that case, we also need to wait for an isolated honest proposer level, where the proposer block will implicitly or explicitly include TB in the ledger. Once this isolated honest proposer level is confirmed *and* all the preceding levels are list-confirmed, we can be sure that TB will appear in the final ledger. The confirmation latency is thus the maximum of two parts:

(1) *List confirmation.* We fast confirm that the adversary cannot produce a private block A with more votes than the votes of public blocks H_1 and A_1 . The logic is similar to the case of isolated honest proposer block discussed above, viewing the situation as a race between honest nodes voting for the public blocks H_1 or A_1 and adversary voting for A . Adversarial actions (e.g., presenting first H_1 to half the honest nodes and A_1 to the other half) can cause the number of votes to be evenly split between H_1 and A_1 , which can slow down list confirmation, albeit not significantly.

(2) *Isolated honest proposer level.* In Figure 6(a), if we wait until level 2, we see an isolated public proposer block H_2 which can be fast confirmed (Section 4.1.1). At this point, we know that the final leader sequence at levels 1, 2 is either H_1, H_2 or A_1, H_2 , both of which contain our honest transaction since H_2 refers to all previous unreferenced proposer blocks. Since isolated honest proposer blocks happen frequently (Section 4.1.3), this step is fast.

Double-Spent Transaction: To confirm double-spent transactions, we need stronger conditions than those listed above: namely, instead of list confirmation, we need *unique block confirmation*, confirming which block at a proposer level will be the ultimate leader. This is achieved once list confirmation occurs *and* one of the list-confirmed blocks can be reliably declared the winner. If one of the public proposer blocks H_1 or A_1 gathers many more votes than the other block, then we can fast confirm a unique leader, even for double-spent transactions; this happens both in the absence of active attacks and under some classes of attacks (Section 5). However, other adversarial attacks (such as balancing the votes on H_1 and A_1) can cause the number of votes to be evenly split between H_1 and A_1 , so we cannot fast confirm a leader block. In this case, we must wait until every vote on H_1 and A_1 stabilizes, in which case either H_1 or A_1 is confirmed and only one of the double-spent transactions is accepted. A content-dependent tie breaking rule can be used to break ties after votes are stabilized.

4.1.3 Sketch of Security and Latency Proofs

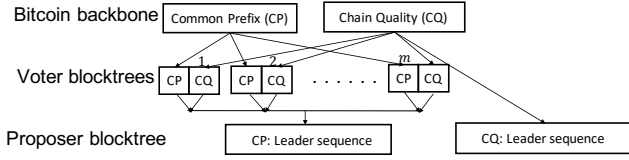


Figure 7: Common-prefix and chain-quality properties of voter chains imply common-prefix and chain-quality properties of the proposer leader sequence.

To translate the above intuitive arguments into formal security and latency proofs, we borrow key ideas from [9], but also require several new insights. [9] proves the consistency and liveness of the Bitcoin backbone protocol by first proving two key properties: common-prefix and chain-quality. Similarly, to show that Prism achieves consistency and liveness, we need to show that the proposer leader sequence satisfies these properties. The results of [9] do not directly apply because the proposer leader sequence is not determined by a single longest-chain protocol; rather, it is determined by a combination of the proposer tree and the aggregate voter tree votes. As shown in Figure 7, we prove the two properties for the proposer and the voter trees and use them to prove corresponding properties for the leader sequence. Specifically:

(1) Each voter tree is constructed according to the backbone protocol, and hence satisfies the chain-quality and common-prefix property. Chain-quality of the voter trees implies that honest nodes will continually be able to vote for proposer blocks from every voter tree and at every proposer level. Common-prefix implies that all these votes will eventually stabilize. This further implies that the leader sequence satisfies the common-prefix property (Theorem 4.1), since the leader block at each level will eventually stabilize. Hence, the resulting ledger is consistent. The leader-sequence also can be shown to have a certain chain quality (Lemma D.6) and this ensures liveness of the ledger (Theorem 4.2).

(2) To show fast confirmation of all honest transactions, we follow the intuitive arguments above. We first show that an isolated proposer block, or an honest proposer block that does not have a competing adversarial proposer block for a certain duration of time, appears in constant expected time (independent of ϵ). Specifically, the honest users are mining proposer blocks at the rate $(1 - \beta)\bar{f}_p$ whereas the adversary is mining at rate $\beta\bar{f}_p$. Since $\beta < 0.5$, the adversary is mining slower than the honest users, and within the next $\frac{1}{1-2\beta}$ levels in expectation, there is a level on which the adversary cannot *immediately* create a competing block with the honest block⁷. Similarly, an isolated level on which the adversary cannot match the honest block for next R rounds after the honest block is mined happens within $\frac{1+2R\bar{f}_v}{1-2\beta}$ levels in expectation.

(3) We next show that we can fast confirm an isolated public honest proposer block. The argument has two parts: i) the isolated honest block wins enough votes; 2) the leader block persists, i.e., wins the vote race against a private adversarial proposer block for

all time. The first part follows from the chain-quality of the voter chains, which ensures that there is a steady stream of honest votes for the public proposer block until it gathers a sufficiently large fraction of total votes (Lemma E.7). The second part follows from common-prefix of the voter trees, which ensures that a large fraction of the votes cannot be reversed by the adversary (Lemma E.9). (4) Fast *list* confirmation of proposer blocks at all previous levels can be proved similarly (see Lemma E.10 and Theorem 4.6). Now, Prism ensures that at each proposer level, one of the list-confirmed blocks will remain in the ledger. This, combined with the assurance that every transaction will be either directly or indirectly referred by the isolated proposal block, ensures that all honest transactions are entered into the ledger. This lets Prism confirm honest transactions within a short time (see Theorem 4.7).

Note that [9] proves the k -common-prefix property is satisfied with high probability only for large k . Similarly, chain-quality is shown to be satisfied with high probability only over a large number of blocks. While this is sufficient to prove (1) and (2) above for the consistency and liveness of the eventual ledger, it is not sufficient to prove (4) and (5) for fast confirmation latency, since we need these two properties over short latencies, i.e. windows of few blocks. In these small time windows, these properties do not hold with high probability *microscopically*, for every *individual* voter tree. However, since the proposer leader block depends only on the *macroscopic* vote counts, we only need to show that these properties hold with high probability *macroscopically*, for a good fraction of voter trees.

4.2 Parameter Selection

We first specify the parameters of Prism in terms of the parameters of the physical network. First, recall that the network delay of a block containing B transactions is given by $\Delta = \frac{B}{C} + D$. Let B_t , B_v , and B_p be the size of transaction, voter, and proposer blocks respectively, in units of number of transactions. The network delays Δ_t , Δ_v , and Δ_p for each type of block are thus given by:

$$\Delta_t = \frac{B_t}{C} + D, \quad \Delta_v = \frac{B_v}{C} + D, \quad \Delta_p = \frac{B_p}{C} + D. \quad (9)$$

Given that different block types have different sizes and network delays, what is a reasonable choice for Δ , the duration of a round? Since the synchronous model is used for security analysis, and the security of Prism depends only on the network delay of the proposer and voter blocks but not of the transaction blocks, we choose: $\Delta = \max\{\Delta_p, \Delta_v\}$. Moreover, the voter blocks and the proposer blocks contain only reference links and no transactions, so their sizes are expected to be small. Assuming the bandwidth-delay product $CD/\max\{B_v, B_p\} \gg 1$, we have that the network delay $\Delta = \max\{\frac{B_v}{C}, \frac{B_p}{C}\} + D \approx D$, the smallest possible.

To provide security, we set the mining rate $\bar{f}_v := f_v D$ on each voter tree such that each voter tree is secure under the longest chain rule. According to [9] it should satisfy

$$\bar{f}_v < \frac{1}{1-\beta} \log \frac{1-\beta}{\beta}. \quad (10)$$

We also set the proposer and voter mining rates to be the same, i.e. $\bar{f}_p = \bar{f}_v$. This is not necessary but simplifies the notation.

Third, to utilize 90% of the communication bandwidth for carrying transaction blocks, we set $f_t B_t = 0.9C$. The individual choices

⁷Random walk analysis

of f_t and B_t are not very important, but choosing large B_t and small f_t is preferable to ensure that the number of reference links to transaction blocks per proposer block is small, thus giving a small proposer block size B_p .

Finally, speed up voting, we maximize the number of voter chains subject to the *stability constraint* of Sec. 2: $f_p B_p + m f_v B_v + f_t B_t < C$. Substituting the values of f_v , f_p and $f_t B_t$, we get

$$m = \frac{0.1CD}{\bar{f}_v B_v} - \frac{B_p}{B_v} \geq \frac{(1-\beta)}{\log(\frac{1-\beta}{\beta})} \cdot \frac{CD}{B_v} - \frac{B_p}{B_v}$$

i.e. the number of voter trees is at least proportional to CD/B_v , the bandwidth-delay product in unit of voting blocks. This number is expected to be large, which is a key advantage. The only degree of freedom left is the choice of \bar{f}_v , subject to (10). We will return to this issue in Section 4.5 when we discuss fast confirmation latency.

4.3 Total Ordering

In this subsection, we show that Prism can achieve total transaction ordering for any $\beta < 0.5$ using the procedure `GETORDEREDCONFIRMEDTXS()` in Algorithm 2. That is, as long as the adversary's hash power is less than 50%, transactions can be ordered with consistency and liveness guarantees. Following [9], we do so by first establishing two backbone properties: common-prefix and chain quality of the *proposer leader sequence*. Let $\mathcal{P}(r)$ denote the set of proposer blocks mined by round r . Let $\mathcal{P}_\ell(r) \subseteq \mathcal{P}(r)$ denote the set of proposer blocks mined on level ℓ by round r . Let the first proposer block on level ℓ be mined in round R_ℓ . Let $V_p(r)$ denote the number of votes on proposer block $p \in \mathcal{P}(r)$ at round r . Recall that only votes from the main chains of the voter trees are counted. The *leader block* on level ℓ at round r , denoted by $p_\ell^*(r)$, is the proposer block with maximum number of votes in the set $\mathcal{P}_\ell(r)$ i.e. $p_\ell^*(r) := \operatorname{argmax}_{p \in \mathcal{P}_\ell(r)} V_p(r)$, where tie-breaking is done in a

hash-dependent way. The *leader sequence* up to level ℓ at round r , denoted by $\text{LedSeq}_\ell(r)$ is:

$$\text{LedSeq}_\ell(r) := [p_1^*(r), p_2^*(r), \dots, p_\ell^*(r)]. \quad (11)$$

The leader sequence at the end of round r_{\max} , the end of the horizon, is the *final leader sequence*, $\text{LedSeq}_\ell(r_{\max})$.

THEOREM 4.1 (LEADER SEQUENCE COMMON-PREFIX PROPERTY). *Suppose $\beta < 0.5$. For a fixed level ℓ , we have*

$$\text{LedSeq}_\ell(r) = \text{LedSeq}_\ell(r_{\max}) \quad \forall r \geq R_\ell + r(\varepsilon) \quad (12)$$

with probability $1 - \varepsilon$, where $r(\varepsilon) = \frac{1024}{\bar{f}_v(1-2\beta)^3} \log \frac{8mr_{\max}}{\varepsilon}$, and R_ℓ is the round in which the first proposer block on level ℓ was mined.

PROOF. See Appendix D. □

THEOREM 4.2 (LIVENESS). *Assume $\beta < 0.5$. Once a transaction enters into a transaction block, w.p $1 - \varepsilon$ it will eventually be pointed to by a permanent leader sequence block after*

$$O\left(\frac{\log(1/\varepsilon)}{(1-2\beta)^4}\right) \text{ rounds.}$$

PROOF. See Appendix D. □

Theorems 4.1 and 4.2 yield the following:

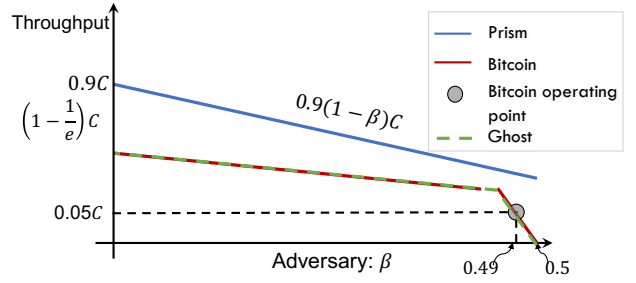


Figure 8: Throughput versus β tradeoffs of Prism, Bitcoin and GHOST. The tradeoffs for the baseline protocols are upper bounds, while that for Prism is exact.

THEOREM 4.3. *The ledger has consistency and liveness with the expected ε -latency for all transactions (Def. 8) to be at most $O(\log(1/\varepsilon))$ for $\beta < 0.5$.*

4.4 Throughput

We now analyze the transaction throughput of Prism. The leader sequence blocks of Prism orders all the transactions in the transaction blocks they refer to. Due to liveness, all transaction blocks are referred to by some proposer blocks. Since the transaction block generation rate $f_t B_t$ is chosen to be $0.9C$ transactions per second, assuming a worst case that only honest blocks carry transactions yield a throughput of $0.9(1-\beta)C$ transactions per seconds.

This seems to give the advertised goal, but there is a catch: blocks mined in the same round may contain the same transactions, since transactions are broadcast to the entire network. To achieve the full throughput, one can minimize the transaction redundancy in the blocks by **scheduling** different transactions to different blocks. Concretely, we split the transactions randomly into q queues, and each honest block is created from transactions drawn from one randomly chosen queue. Thinking of each transaction queue as a color, we have transaction blocks of q different colors.

We will only have honest blocks with redundant transactions if two or more blocks of the same color are mined in the same round. The number of honest blocks of the same color mined at the same round is distributed as Poisson with mean $(1-\beta)f_t \Delta/q$, and so the throughput of non-redundant blocks of a given color is the probability that at least one such block is mined in a round, i.e. $1 - e^{-(1-\beta)f_t \Delta/q}$ blocks per round. The total throughput of non-redundant honest blocks of all colors is

$$q \left[1 - e^{-(1-\beta)f_t \Delta/q} \right] \text{ blocks per round.} \quad (13)$$

For large q , this approaches $(1-\beta)f_t \Delta$ blocks per round, which equals $0.9(1-\beta)C$ transactions per second when we set $f_t = 0.9C/B_t$. Thus, we achieve the claimed result (4). Note that the throughput as a fraction of capacity does not vanish as $\beta \rightarrow 0.5$, unlike Bitcoin and GHOST (App. H and I).

THEOREM 4.4 (THROUGHPUT). *Theorem 4.8 guarantees that all the transactions proposed before round $r - O(\log(1/\varepsilon))$ are confirmed by round r . Therefore Prism can support $\lambda = 0.9(1-\beta)C$ throughput (Def. 7), where $U_\varepsilon = O(\log(1/\varepsilon))$.*

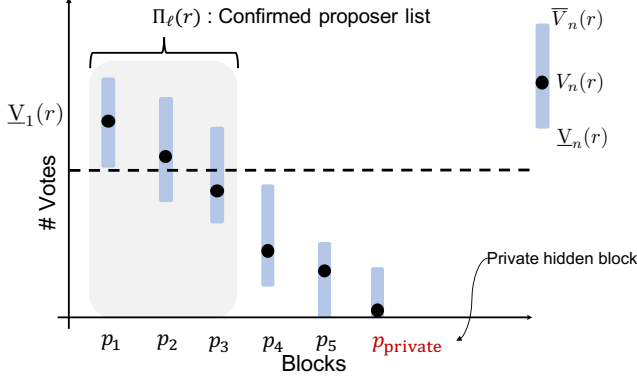


Figure 9: Public proposer block p_1 has the largest lower confidence bound, which is larger than the upper confidence bound of the private block. So list confirmation is possible and the set confirmed is $\Pi_\ell(r) = \{p_1, p_2, p_3\}$.

4.5 Fast confirmation latency

4.5.1 List confirmation latency

We convert the intuition from Section 4.1 to a formal rule for fast confirming a set of proposer blocks, which enables confirming a list of proposer sequences. The idea is to have *confidence intervals* around the number of votes cast on each proposer block. Figure 9 gives an example where there are 5 proposal blocks in public at a given level, and we are currently at round r . The confidence interval $[\underline{V}_n(r), \bar{V}_n(r)]$ for the votes on proposer block p_n bounds the maximum number of votes the block can lose or gain from uncast votes and votes reversed by the adversary. We also consider a potential private proposer block, with an upper bound on the maximum number of votes it can accumulate in the future. We can fast confirm a set of proposal blocks whenever the upper confidence bound of the private block is below the lower confidence bound of the public proposal block with the largest lower confidence bound.

More formally: As defined earlier, $\mathcal{P}_\ell(r) = \{p_1, p_2, \dots\}$ is the set of proposer blocks at level ℓ at round r . Let $V_n^d(r)$ be the number of votes at depth d or greater for proposer block p_n at round r . Define:

$$\delta_d := \max\left(\frac{1}{4\bar{f}_v d}, \frac{1-2\beta}{8 \log m}\right), \epsilon' = 1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{16 \log m}},$$

$$\underline{V}_n(r) := \max_{d \geq 0} \left(V_n^d(r) - 2\delta_d m \right)_+,$$

$$\bar{V}_n(r) := m - \sum_{p_{n'} \in \mathcal{P}_\ell(r) \setminus \{p_n\}} \underline{V}_{n'}(r),$$

$$\underline{V}_{\text{private}}(r) := 0, \quad \bar{V}_{\text{private}}(r) := m - \sum_{p_{n'} \in \mathcal{P}_\ell(r)} \underline{V}_{n'}(r).$$

DEFINITION 4.5. Proposer set confirmation policy: If

$$\max_n \underline{V}_n(r) > \bar{V}_{\text{private}}(r), \quad (14)$$

then we confirm the set of proposer blocks $\Pi_\ell(r)$, where

$$\Pi_\ell(r) := \{p_n : \bar{V}_n(r) > \max_i \underline{V}_i(r)\}. \quad (15)$$

Next, we show that we can confirm proposer sets up to level ℓ with an expected latency independent of ϵ , and the final leader sequence is contained in the outer product of the confirmed sets.

THEOREM 4.6 (LIST COMMON-PREFIX PROPERTY). *Suppose $\beta < 0.5$. Suppose the first proposer block at level ℓ appears at round R_ℓ . Then w.p. ϵ' , we can confirm proposer sets $\Pi_1(r), \dots, \Pi_\ell(r)$ for all rounds $r \geq R_\ell + R_\ell^{\text{conf}}$, where*

$$\mathbb{E}[R_\ell^{\text{conf}}] \leq \frac{2808}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)} + \frac{256}{(1-2\beta)^6 \bar{f}_v m^2}, \quad (16)$$

and $p_{\ell'}^*(r_{\max}) \in \Pi_{\ell'}(r) \quad \forall \ell' \leq \ell$ and $r \geq R_\ell + R_\ell^{\text{conf}}$.

PROOF. See Appendix E. \square

Let us express the latency bound (16) in terms of physical parameters. If we set the voting rate \bar{f}_v equal to the largest possible given the security constraint (10): $\bar{f}_v = \frac{1}{1-\beta} \log \frac{1-\beta}{\beta}$, then according to (11), we have

$$m = \frac{0.1(1-\beta)}{\log(\frac{1-\beta}{\beta})} \cdot \frac{CD}{B_v} - \frac{B_p}{B_v}.$$

With this choice of parameters, and in the regime where the bandwidth-delay product CD/B_v is large so that the second term in (16) can be neglected, the expected latency for list confirmation is bounded by $c_1(\beta)D$ seconds, i.e. proportional to the propagation delay. Here,

$$c_1(\beta) := \frac{2808(1-\beta)}{(1-2\beta)^3 \log \frac{1-\beta}{\beta}} \log \frac{50}{(1-2\beta)}$$

and is positive for $\beta < 0.5$. The confirmation error probability is exponentially small in CD/B_v . This is the constant part of the latency versus security parameter tradeoff of Prism in Fig. 1. Since CD/B_v is very large in typical networks, a confirmation error probability exponentially small in CD/B_v is already very small. To achieve an even smaller error probability ϵ we can reduce the voting rate \bar{f}_v smaller below the security constraint (10) and increase the number of voter chains. More specifically, we set

$$\bar{f}_v = \frac{0.1CD}{B_v \log \frac{1}{\epsilon}}, \quad (17)$$

resulting in $m = \log \frac{1}{\epsilon} - \frac{B_p}{B_v} \approx \log \frac{1}{\epsilon}$, yielding the desired security parameter. Again neglecting the second term in (16), the corresponding latency bound is

$$\frac{c_2(\beta)B_v}{C} \log \frac{1}{\epsilon} \quad \text{seconds,}$$

where $c_2(\beta) := \frac{54000}{(1-2\beta)^3} \log \frac{50}{(1-2\beta)}$. This is the linearly increasing part of the Prism curve in Figure 1, with slope inversely proportional to the network capacity C/B_v .

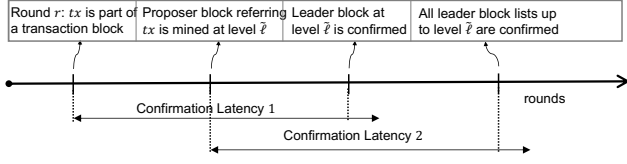


Figure 10: Components of the latency: a) Confirmation latency 1 is analyzed in Theorem 4.7, and b) Confirmation latency 2 is analyzed in Theorem 4.6.

4.5.2 Fast confirmation of honest transactions

In the previous subsection we have shown that one can fast confirm a set of proposer block sequences which is guaranteed to contain the prefix of the final totally ordered leader sequence. As discussed in Section 3, each of these proposer block sequence creates an ordered ledger of transactions using the reference links to the transaction blocks. In each of these ledgers, double-spends are removed to sanitize the ledger. If a transaction appears in *all* of the sanitized ledgers in the list, then it is guaranteed to be in the final total ordered sanitized ledger, and the transaction can be fast confirmed. All honest transactions without double-spends eventually have this *list-liveness* property; when only a single honest proposer block appears in a level and becomes the leader, it adds any honest transactions that have not already appeared in at least one ledger in the list. Due to the positive chain-quality of the leader sequence (Theorem 4.2), an isolated honest level eventually occurs. The latency of confirming honest transactions is therefore bounded by the sum of the latency of list confirmation in Theorem 4.6 plus the latency of waiting for this event to occur (Fig. 10). The latter is given by:

THEOREM 4.7 (LIST-LIVENESS). *Assume $\beta < 0.5$. If an honest transaction without double spends is mined in a transaction block in round r , then w.p. $1 - r_{\max}^2 e^{-\frac{m}{16 \log m}}$ it will appear in all of the ledgers corresponding to proposer block sequences after an expected latency no more than*

$$\frac{2592}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)} \text{ rounds.}$$

PROOF. Appendix F. □

Figure 10 shows the various components of the overall latency we analyzed. We can see that the confirmation latency from the time an honest transaction enters a blocks to the time it is confirmed is bounded by the sum of the latencies in Theorem 4.6 and 4.7.

Repeating the analysis of Thm. 4.3, we get the following:

THEOREM 4.8 (LATENCY). *Theorems 4.6 and 4.7 guarantee that the expected ε -latency for all **honest** transactions (Def. 8) is at most $r(\beta)$ rounds for $\beta < 0.5$, where*

$$r(\beta) := \max \left\{ c_1(\beta), c_2(\beta) \frac{B_v}{DC} \log \frac{1}{\varepsilon} \right\},$$

where

$$c_1(\beta) := \frac{5400(1-\beta)}{(1-2\beta)^3 \log \frac{1-\beta}{\beta}} \log \frac{50}{(1-2\beta)}$$

$$c_2(\beta) := \frac{54000}{(1-2\beta)^3} \log \frac{50}{(1-2\beta)},$$

Therefore the honest transactions are confirmed in

$$\max \left\{ c_1(\beta)D, c_2(\beta) \frac{B_v}{C} \log \frac{1}{\varepsilon} \right\} \text{ seconds.}$$

5 SIMULATIONS

Theorem 4.8 provides a theoretical upper bound on the expected latency, which matches the physical limit of propagation time up to constant factors. Characterizing the exact constants is an interesting research direction, but outside the scope of this paper. On the other hand, one can empirically estimate the average latency values by simulating the Prism protocol and its confirmation rule. The purpose of this section is to conduct such a simulation in the honest setting as well as a variety of adversarial settings.

Setup. We simulate a network with $m = 1,000$ voter chains, in which $D \approx \Delta = 1$ sec. We run our proposer tree and each voter tree at a rate of $\bar{f} = 1$ block /10 sec. Our simulations measure the latency for transaction confirmation under three scenarios: no attack, a balancing attack, and a censorship attack. By design, our confirmation rule is simultaneously robust against the common private Nakamoto attack [16], where the adversary withholds a proposer block as well as corresponding forked voter blocks in order to reverse a confirmed proposal block. In this section, we show figures for an adversary deploying $\hat{\beta} = 0.25$ fraction of total hash power, where $\hat{\beta}$ denotes the fraction of hash power being actually used for the attack (whereas β is the maximum tolerable fraction of adversarial hash power, without losing consistency and liveness). We set the confirmation reliability conservatively at $\varepsilon = e^{-20}$. Experiments for additional parameter settings can be found in Appendix J. We compare against the longest-chain protocol, for the same block generation rate of 1 block per 10 seconds.

No Attack. We start by considering a setting where Prism's parameters are chosen to withstand an attacker of hash power β , but the adversary is not actively conducting any attack. Since the confirmation rule must still defend against β adversarial hash power, latency depends on β . Honest nodes vote on the earliest-seen proposer block, with results shown in Figure 11(a). In Bitcoin, a confirmed transaction has to be deeper in the chain for larger β ; in Prism, the voter blocks have to be deeper. We see that Prism's latency is significantly smaller than that of Nakamoto's longest chain protocol, and much closer to the physical limit. Note that since there is no active adversary, double-spend transactions can be resolved with the same latency as honest transactions.

Balancing Attack. In a balancing attack, the goal of the adversary is to prevent confirmation by casting all of its votes so as to compete with the current proposer leader block. We begin this attack with two competing proposer blocks at the same level (say level 0), A and B . Consider an honest (non-double-spent) transaction that is referred by at least one of the two proposer blocks. The adversary's goal is to prevent the system from confirming this transaction by balancing votes on the two proposer blocks. That is, if block A currently has the majority of votes and the adversary mines a voter block in the i th voter tree: (1) If voter tree i has not yet voted on level 0, the adversary votes on the minority block, B . (2) If voter tree i voted on level 0 for block B , the adversary appends its block to the longest chain, thereby reinforcing the vote for the losing proposer block. (3) If voter tree i voted on level 0 for block A ,

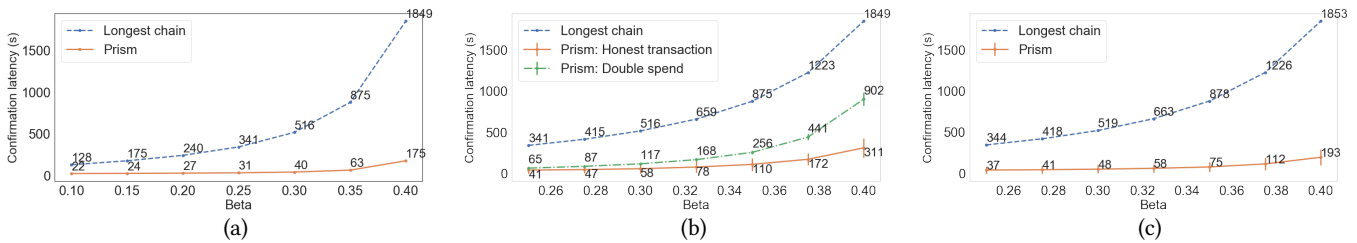


Figure 11: (a) Confirmation latency of honest transactions with no attack. The x-axis denotes the maximum tolerable fraction of adversarial hash power β . (b) Transaction latency in the presence of an adversarial balancing attack from $\tilde{\beta} = 0.25$ active hash power, for honest and double-spent transactions. (c) Confirmation latency under a censorship attack with $\tilde{\beta} = 0.25$ hash power. Honest and double-spent transactions have the same latency, both for Prism and for longest chain.

the adversary tries to fork the i th voter tree to vote for B instead. If there is no vote for B in the voter tree, the adversary creates one. If there is already a fork voting for B , the adversary appends to this fork. The balancing attack is one of the most severe and natural attacks on Prism. The results of this simulation are shown in Figure 11(b). Notice that the latency of honest transaction confirmation increases by a factor of about 2x under a balancing attack, but does not affect the longest-chain protocol. Despite this, Prism’s latency is still far lower than that of the longest-chain protocol.

Next, we consider double-spent transactions. The latency for double-spent transactions is the same as honest transactions in the longest-chain protocol, so the blue curve does not change. However, the double-spent transaction latency for Prism grows substantially, approaching that of the longest-chain protocol. Indeed, as the active $\tilde{\beta}$ fraction approaches 0.5, Prism’s latency on double-spent transactions in the presence of attacks on the confirmation process actually exceeds that of the longest-chain protocol, as discussed in Section 4.1 and seen in Figures 20 and 21.

Censorship Attack Finally, we consider an attacker whose goal is simply to slow down the confirmation of blocks by proposing empty proposer and voter blocks. This has two effects: (1) it delays the creation of a proposer block referencing the transaction block containing the transaction, and (2) it delays the confirmation of such a proposer block by delaying the creation of votes on the proposer tree. The results of this attack are shown in Figure 11(c). The censorship attack adds a delay of between 15-20 seconds to Prism’s confirmation delay compared to the non-adversarial setting. The effect is smaller for the longest-chain protocol, since the only delay comes from delaying the insertion of a transaction into a block. Under a censorship attack, double-spent transactions have the same latency as honest ones.

ACKNOWLEDGEMENT

We thank the Distributed Technologies Research Foundation, the Army Research Office under grant W911NF-18-1-0332-(73198-NS), the National Science Foundation under grants 1705007 and 1651236 for supporting their research program on blockchain technologies. We thank Applied Protocol Research Inc. for support and for providing a conducive environment that fostered this collaborative research. We also thank Andrew Miller and Mohammad Alizadeh for their comments on an earlier draft. We also thank Soubhik Deb for helping us with the simulations.

REFERENCES

- [1] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. *arXiv preprint arXiv:1612.02916*, 2016.
- [2] David Aldous and Jim Fill. Reversible markov chains and random walks on graphs, 2002.
- [3] Gavin Andresen. Weak block thoughts. bitcoin-dev. <https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-September/011157.html>.
- [4] Vitalik Buterin. On slow and fast block times, 2015. <https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/>.
- [5] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.
- [6] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13. ACM, 2016.
- [7] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoinng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.
- [8] Matthias Fitz, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. *Cryptology ePrint Archive*, Report 1119, 2018.
- [9] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.
- [10] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *ACM SIGCOMM computer communication review*, 32(4):89–102, 2002.
- [11] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. 2016.
- [12] Lucianna Kiffer, Rajmohan Rajaraman, et al. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 729–744. ACM, 2018.
- [13] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium*, pages 279–296, 2016.
- [14] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.
- [15] Chenxing Li, Peilun Li, Wei Xu, Fan Long, and Andrew Chi-chih Yao. Scaling nakamoto consensus to thousands of transactions per second. *arXiv preprint arXiv:1805.03870*, 2018.
- [16] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
- [17] Christopher Natoli and Vincent Gramoli. The balance attack against proof-of-work blockchains: The r3 testbed as an example. *arXiv preprint arXiv:1612.09426*, 2016.
- [18] R Pass, L Seeman, and A Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2017.
- [19] R. Pass and E. Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017.
- [20] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *LIPICs-Leibniz International Proceedings in Informatics*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
- [21] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018.

- [22] Peter R Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.
- [23] Y Sompolinsky, Y Lewenberg, and A Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159.
- [24] Y Sompolinsky and A Zohar. Phantom: A scalable blockdag protocol, 2018.
- [25] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.
- [26] TierNolan. Decoupling transactions and pow. Bitcoin Forum. <https://bitcointalk.org/index.php?topic=179598.0>.
- [27] Haifeng Yu, Ivica Nikolic, Ruomu Hou, and Prateek Saxena. OHIE: blockchain scaling made simple. *CoRR*, abs/1811.12628, 2018.

A PSEUDOCODE

Algorithm 1 Prism: Mining

```

1: procedure MAIN()
2:   INITIALIZE()
3:   while True do
4:      $header, Ppf, Cpf = \text{POWMINING}()$ 
5:     // Block contains header, parent, content and merkle proofs
6:     if header is a tx block then
7:        $block \leftarrow \langle header, txParent, txPool, Ppf, Cpf \rangle$ 
8:     else if header is a prop block then
9:        $block \leftarrow \langle header, prpParent, unRfTxBkPool, Ppf, Cpf \rangle$ 
10:    else if header is a block in voter blocktree i then
11:       $block \leftarrow \langle header, vtParent[i], votesOnPrpBks[i], Ppf, Cpf \rangle$ 
12:    BROADCASTMESSAGE(block)
13:  procedure INITIALIZE()
14:    // Blockchain data structure  $C = (prpTree, vtTree)$ 
15:     $prpTree \leftarrow \text{genesisP}$ 
16:    for  $i \leftarrow 1$  to  $m$  do
17:       $vtTree[i] \leftarrow \text{genesisM}_i$ 
18:    // Parent blocks to mine on
19:     $prpParent \leftarrow \text{genesisP}$ 
20:    for  $i \leftarrow 1$  to  $m$  do
21:       $vtParent[i] \leftarrow \text{genesisM}_i$ 
22:    // Block content
23:     $txPool \leftarrow \phi$ 
24:     $unRfTxBkPool \leftarrow \phi$ 
25:     $unRfPrpBkPool \leftarrow \phi$ 
26:    for  $i \leftarrow 1$  to  $m$  do
27:       $votesOnPrpBks(i) \leftarrow \phi$ 
28:  procedure POWMINING()
29:    while True do
30:       $txParent \leftarrow prpParent$ 
31:      // Assign content for all block types/trees
32:      for  $i \leftarrow 1$  to  $m$  do  $vtContent[i] \leftarrow votesOnPrpBks[i]$ 
33:       $txContent \leftarrow txPool$ 
34:       $prContent \leftarrow (unRfTxBkPool, unRfPrpBkPool)$ 
35:      // Define parents and content Merkle trees
36:       $parentMT \leftarrow \text{MerklTree}(vtParent, txParent, prpParent)$ 
37:       $contentMT \leftarrow \text{MerklTree}(vtContent, txContent, prContent)$ 
38:       $nonce \leftarrow \text{RandomString}(1^\kappa)$ 
39:      // Header is similar to Bitcoin
40:       $header \leftarrow \langle parentMT.root, contentMT.root, nonce \rangle$ 
41:      // Sortition into different block types/trees
42:      if  $\text{Hash}(header) \leq mf_v$  then
43:         $i \leftarrow \lfloor \text{Hash}(header)/f_v \rfloor$  and break
44:      else if  $mf_v < \text{Hash}(header) \leq mf_v + f_t$  then
45:         $i \leftarrow m + 1$  and break
46:      else if  $mf_v + f_t < \text{Hash}(header) \leq mf_v + f_t + f_p$  then
47:         $i \leftarrow m + 2$  and break
48:      // Return header along with Merkle proofs
49:      return  $\langle header, parentMT.proof(i), contentMT.proof(i) \rangle$ 
50:  procedure RECEIVEBLOCK(B)
51:    if B is a valid transaction block then
52:       $txPool.removeTxFrom(B)$ 
53:       $unRfTxBkPool.append(B)$ 
54:    else if B is a valid block on  $i^{\text{th}}$  voter tree then
55:       $vtTree[i].append(B)$  and  $vtTree[i].append(B.ancestors())$ 
56:      if  $B.chainlen > vtParent[i].chainlen$  then
57:         $vtParent[i] \leftarrow B$  and  $votesOnPrpBks(i).update(B)$ 
58:    else if B is a valid prop block then
59:      if  $B.level == prpParent.level + 1$  then
60:         $prpParent \leftarrow B$ 
61:        for  $i \leftarrow 1$  to  $m$  do
62:           $votesOnPrpBks(i)[B.level] \leftarrow B$ 
63:      else if  $B.level > prpParent.level + 1$  then
64:        // Miner doesnt have block at level  $prpParent.level + 1$ 
65:        REQUESTNETWORK( $B.PARENT$ )
66:         $prpTree[B.level].append(B)$ ,  $unRfPrpBkPool.append(B)$ 
67:         $unRfTxBkPool.removeTxBkRefsFrom(B)$ 
68:         $unRfPrpBkPool.removePrpBkRefsFrom(B)$ 
69:  procedure RECEIVETX(tx)
70:    if tx has valid signature then  $txPool.append(B)$ 

```

► Broadcast to peers
► All variables are global

► Proposer Blocktree

► Voter *i* blocktree

► Proposer block to mine on

► Voter tree *i* block to mine on

► Tx block content: Txs to add in tx bks

► Prop bk content1: Unreferred tx bks

► Prop bk content2: Unreferred prp bks

► Voter tree *i* bkbk content

► Voter block mined
► on tree *i*

► Tx block mined

► Prop block mined

► Get block from peers

► Add vote on level ℓ on all m trees

Algorithm 2 Prism: Tx confirmation

```
1: procedure ISTXCONFIRMED( $tx$ )
2:    $\Pi \leftarrow \phi$  ▷ Array of set of proposer blocks
3:   for  $\ell \leftarrow 1$  to  $prpTree.maxLevel$  do
4:      $votesNdepth \leftarrow \phi$ 
5:     for  $i$  in  $1$  to  $m$  do
6:        $votesNdepth[i] \leftarrow GETVOTENDEPTH(i, \ell)$ 
7:       if IsPropSetConfirmed( $votesNdepth$ ) then ▷ Refer Def. 4.5
8:          $\Pi[\ell] \leftarrow GetProposerSet(votesNdepth)$  ▷ Refer Eq. 15
9:       else break
10:    // Ledger list decoding: Check if tx is confirmed in all ledgers
11:     $prpBksSeqs \leftarrow \Pi[1] \times \Pi[2] \times \dots \times \Pi[\ell]$  ▷ outer product
12:    for  $prpBks$  in  $prpBksSeqs$  do
13:       $ledger = BUILDLEDGER(prpBks)$ 
14:      if  $tx$  is not confirmed in  $ledger$  then return False ▷ Return true if tx is confirmed in all ledgers
15:      return True
16: procedure GETVOTENDEPTH( $i, \ell$ )
17:    $voterMC \leftarrow vtTree[i].LongestChain()$ 
18:   for  $voterBk$  in  $voterMC$  do
19:     for  $prpBk$  in  $voterBk.votes$  do
20:       if  $prpBk.level = \ell$  then
21:         // Depth is #of children bks of voter bk on main chain
22:         return ( $prpBk, voterBk.depth$ )
23: procedure BUILDLEDGER( $propBlocks$ ) ▷ Input: list of prop blocks
24:    $ledger \leftarrow []$  ▷ List of valid transactions
25:   for  $prpBk$  in  $propBlocks$  do
26:      $refPrpBks \leftarrow prpBk.getReferredPrpBks()$ 
27:     // Get all directly and indirectly referred transaction blocks.
28:      $txBks \leftarrow GetOrderedTxBks(prpBk, refPrpBks)$ 
29:     for  $txBk$  in  $txBks$  do
30:        $txs \leftarrow txBk.getTxS()$  ▷ TxS are ordered in  $txBk$ 
31:       for  $tx$  in  $txs$  do
32:         // Check for double spends and duplicate txs
33:         if  $tx$  is valid w.r.t to  $ledger$  then  $ledger.append(tx)$ 
34:   return  $ledger$ 
35: // Return ordered list of confirmed transactions
36: procedure GETORDEREDCONFIRMEDTXS()
37:    $L \leftarrow \phi$  ▷ Ordered list of leader blocks
38:   for  $\ell \leftarrow 1$  to  $prpTree.maxLevel$  do
39:      $votes \leftarrow \phi$  ▷ Stores votes from all  $m$  voter trees on level  $\ell$ 
40:     for  $i$  in  $1$  to  $m$  do
41:        $votesNDepth[i] \leftarrow GETVOTES(i, \ell)$ 
42:       if IsLeaderConfirmed( $votesNDepth$ ) then ▷ Refer 4.1
43:         // Proposer block with maximum votes on level  $\ell$ 
44:          $L[\ell] \leftarrow GetLeader(votesNDepth)$ 
45:       else break
46:   return BUILDLEDGER(L)
```

B NOTATION

Let $H_i[r]$ and $Z_i[r]$ be the number of voter blocks mined by the honest nodes and by the adversarial node in round r on the i -th voting tree respectively, where $i = 1, 2, \dots, m$. $H_i[r], Z_i[r]$ are Poisson random variables with means $(1 - \beta)f_v\Delta$ and $\beta f_v\Delta$ respectively. Similarly, $H^p[r], Z^p[r]$ are the numbers of proposer blocks mined by the honest nodes and by the adversarial node in round r respectively; they are also Poisson, with means $(1 - \beta)f_p\Delta$ and $\beta f_p\Delta$ respectively. Finally, $H^t[r], Z^t[r]$ are the numbers of transaction blocks mined by the honest nodes and by the adversarial node in round r respectively; they are also Poisson, with means $(1 - \beta)f_t\Delta$ and $\beta f_t\Delta$ respectively. All the random variables are independent of each other.

C BITCOIN BACKBONE PROPERTIES REVISITED

[9] defines three important properties of the Bitcoin backbone: common-prefix, chain-quality and chain-growth. It was shown that, under a certain *typical execution* of the mining process, these properties hold, and the properties are then used to prove the persistence and liveness of the Bitcoin transaction ledger. These three properties, as well as the notion of a typical execution, were *global*, and defined over the *entire* time horizon. While this is appropriate when averaging over time to achieve reliable confirmation, as for Bitcoin, it turns out that for the analysis of fast latency of Prism, where the averaging is over voter chains, we need to formulate finer-grained, *local* versions of these properties, localized at a particular round. Correspondingly, the event under which these local backbone properties are proved is also local, in contrast to the event of typical execution.

In this section, we will focus on a single Bitcoin blocktree, with a mining rate of \bar{f} per round, and we will use the model and notations introduced in Section 2. In addition, we will use the following notation from [9]: if C is a chain of blocks, then $C^{\lceil k}$ is the k -deep prefix of C , i.e. the chain of blocks of C with the last k blocks removed. Additionally, given two chains C and C' , we say that $C \leq C'$ if C is a *prefix* of chain C' .

Definition C.1 (Common-prefix property). The k -deep common-prefix property holds at round r if the k -deep prefix of the longest chain at round r remains a prefix of any longest chain in any future round.

Note that while the common-prefix property in [9] is parameterized by a single parameter k , the property defined here is parameterized by two parameters k and r . It is a property that the prefix of the main chain at round r remains permanently in the main chain in the future.

Definition C.2 (Chain-quality property). The (μ, k) -chain-quality property holds at round r if at most μ fraction of the last k consecutive blocks on the longest chain C at round r are mined by the adversary.

The chain-quality property in [9] is parameterized by two parameters μ and k , however, the property defined here is parameterized by three parameters μ , k and r .

Definition C.3 (Chain-growth property). The chain-growth property with parameters ϕ and s states that for any s rounds there are at least ϕs blocks added to the main chain during this time interval.

We will now show that these three properties hold regardless of adversarial action, provided that certain events on the honest and adversarial mining processes hold.

C.1 Modelling PoW block generation

In section 2, the hash computation of the users are modelled as a random oracle. We now further model the PoW generation as follows: Let $H[r]$ and $Z[r]$ be the number of blocks mined by the honest nodes and by the adversarial node in round r . From section 2, we know that $H[r], Z[r]$ are Poisson random variables with means $(1 - \beta)f_v\Delta$ and $\beta f_v\Delta$ respectively. Note that random variables $\{H[r]\}_{r \in \{0, r_{max}\}}$, $\{Z[r]\}_{r \in \{0, r_{max}\}}$ are independent of each other. We now define auxiliary random variables $X[r]$ and $Y[r]$ as follows: If at round r an honest party mines at least one block, then $X[r] = 1$, otherwise $X[r] = 0$. If at round r an honest party mines exactly one block, then $Y[r] = 1$, otherwise $Y[r] = 0$. Let $r' = \frac{k}{2\bar{f}}$. Define the following events:

$$\begin{aligned}
 E_1[r - r', r] &:= \bigcap_{a, b \geq 0} \left\{ Y[r - r' - a, r + b] - Z[r - r' - a, r + b] > \frac{(1 - 2\beta)k}{8} \right\} \\
 E_2[r - r', r] &:= \{H[r - r', r] + Z[r - r', r] < k\} \\
 E_3[r - r', r] &:= \left\{ X[r - r', r] > \frac{k}{6} \right\} \\
 E[r - r', r] &:= E_1[r - r', r] \cap E_2[r - r', r] \cap E_3[r - r', r].
 \end{aligned} \tag{18}$$

As defined in Section 2, $X[r - r', r]$ and $Y[r - r', r]$ are the number of successful and uniquely successful rounds respectively in the interval $[r - r', r]$, and $Z[r - r', r]$ is the number of blocks mined by adversary

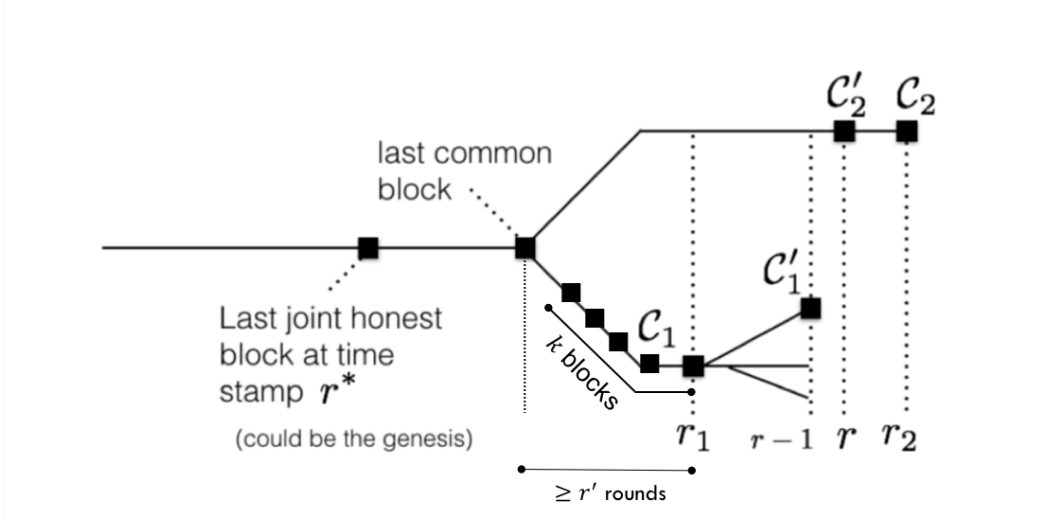


Figure 12: Round r is the first round that the k -deep prefix of the longest chain is changed. (This is a slight modification of Figure 3 from [9].)

in the interval $[r - r', r]$. Note that the honest users mine at least one block in a successful round and mine exactly one block in a uniquely successful round. Therefore, the event $E_1[r - r', r]$ implies that the number of uniquely successful rounds exceed the total blocks mined by the adversary by $\frac{(1-2\beta)k}{8}$ blocks for *all* the intervals containing the interval $[r - r', r]$. Event $E_2[r - r', r]$ implies that the number successful rounds plus the total number of blocks mined by the adversary in the interval $[r - r', r]$ is less than k . Event $E_3[r - r', r]$ implies that the number of successful rounds in the interval $[r - r', r]$ at least $\frac{k}{6}$.

To prove the common-prefix, chain-quality and chain-growth properties, we need the following two lemmas from [9]:

LEMMA C.4 (LEMMA 6 [9]). *Suppose the k -th block, b , of a longest chain C was mined by a honest node in a uniquely successful round. Then the k -th block of a longest chain C' , at possibly a different round, is either b or has been mined by the adversary.*

LEMMA C.5 (LEMMA 7 [9]). *Suppose that at round r_1 the longest chain is of length n . Then by round $r_2 \geq r_1$, the longest chain is of length of least $n + X[r_1, r_2]$.*

LEMMA C.6. *Under the event $E[r - r', r]$, the last k consecutive blocks of the longest chain C at round r are mined in at least r' consecutive rounds.*

PROOF. By definition we know that $E_2[r - r', r] \supseteq E[r - r', r]$. Event $E_2[r - r', r]$ implies that the total number of blocks mined in interval $[r - r', r]$ is less than k . Therefore, the k -th deep block of chain C was mined on or before round $r - r'$. \square

The chain-growth lemma stated below is the localized version of Theorem 13 from [9] and the proof is similar.

LEMMA C.7 (CHAIN-GROWTH). *Under event $E[r - r', r]$, where $r' = \frac{k}{2\beta}$, the longest chain grows by at least $\frac{k}{6}$ blocks in the interval $[r - r', r]$.*

PROOF. From Lemma C.5, we know that the main chain grows by at least $X[r - r', r]$ in the interval $[r - r', r]$. Since $E_3[r - r', r] \supseteq E[r - r', r]$ implies $X[r - r', r] > \frac{k}{6}$ and this completes the proof. \square

We modify the proofs of Lemma 14 and Theorem 15 of [9] by localizing it to a particular round in order to prove our common-prefix property.

LEMMA C.8 (COMMON-PREFIX). *Under the event $E[r_1 - r', r_1]$, where $r' = \frac{k}{2f}$, the k -deep common-prefix property holds at round r_1 .*

PROOF. Consider a longest chain C_1 in the current round r_1 and a longest chain C_2 in a future round r_2 , which violates the common-prefix property, i.e., $C_1^{\lceil k} \not\leq C_2$. Let r be the smallest round $r_1 \leq r \leq r_2$ such that there is a longest chain C'_2 such that $C_1^{\lceil k} \not\leq C'_2$. If $r = r_1$, define $C'_1 = C_1$; otherwise, define C'_1 to be a longest chain at round $r - 1$. Note that $C_1^{\lceil k} \leq C'_1$. Observe that by our assumptions such an r is well-defined (since e.g., r_2 is such a round, albeit not necessarily the smallest one); refer to Figure 12 for an illustration. Consider the last block b^* on the common prefix of C'_1 and C'_2 that was mined by an honest node and let r^* be the round in which it was mined (if no such block exists let $r^* = 0$). Define the set of rounds $S = \{i : r^* < i \leq r\}$. We claim

$$Z[r^*, r] \geq Y[r^*, r]. \quad (19)$$

We show this by pairing each uniquely successful round in S with an adversarial block mined in S . For a uniquely successful round $u \in S$, let j_u be the position of the corresponding block i.e., its distance from the genesis block. Consider the set

$$J := \{j_u : u \text{ is a uniquely successful round in } S\}.$$

Note that $\text{len}(C'_1) \geq \max J$, because the honest node that mined the chain corresponding to $\max J$ position will broadcast it. Since C'_2 is adopted at round r , it should be at least as long as C'_1 , i.e., $\text{len}(C'_2) \geq \text{len}(C'_1)$. As a result, for every $j \in J$, there is a block in position j of either chain. We now argue that for every $j \in J$ there is an adversarial block in the j -th position either in C'_1 or in C'_2 mined after round r^* because C'_1 and C'_2 contains block b^* which is mined by the honest users: if j lies on the common prefix of C'_1 and C'_2 it is adversarial by the definition of r^* ; if not, the argument follows from Lemma C.4.

We assume the event $E[r_1 - r', r_1]$ occurs and under $E_2[r_1 - r', r_1] \supseteq E[r_1 - r', r_1]$, from Lemma C.6, the k -deep block of the chain C_1 was mined on or before round $r_1 - r'$ and this implies $r^* \leq r_1 - r'$. Under the event $E_1[r_1 - r', r_1]$ we know that $Y[r_1 - r' - a, r_1 + b] > Z[r_1 - r' - a, r_1 + b]$ for all $a, b \geq 0$. Since $r^* < r_1 - r'$, $Y[r^*, r] > Z[r^*, r]$, which contradicts Equation (19). \square

We again modify the proof of Theorem 16 of [9] by localizing it to a particular round in order to prove our chain-quality property.

LEMMA C.9 (CHAIN-QUALITY). *Under the event $E[r - r', r]$, where $r' = \frac{k}{2f}$, the (μ, k) -chain quality property holds at round r for $\mu = \frac{7+2\beta}{8}$.*

PROOF. Let C be the longest chain at round r and denote the last k blocks in the chain C by $C[-k] := [b_k, b_{k-1}, \dots, b_2, b_1]$. Now define $N \geq k$ as the the least number of consecutive blocks $C[-N] := [b_N, b_{N-1}, \dots, b_2, b_1]$ s.t block b_N was mined by an honest user. Let block b_N be mined in round r^* . If no such block exists then b_N is the genesis block and $r^* = 0$. Now consider the interval $S = \{i : r^* < i \leq r\} = [r^*, r]$. Let H be the number of blocks mined by honest users in the interval $[r^*, r]$ and say $H < (1 - \mu)k$. Then the number of blocks mined by the adversary in the same interval is at least $N - 1 - H$. This implies $Z[r^*, r] \geq N - 1 - H$, so from the chain-growth Lemma C.5, we have $N - 1 \geq X[r^*, r]$. Putting the last two statements together, we have

$$Z[r^*, r] > X[r^*, r] - (1 - \mu)k. \quad (20)$$

We assume the event $E[r - r', r] = E_1[r - r', r] \cap E_2[r - r', r] \cap E_3[r - r', r]$ occurs. Under $E_2[r - r', r]$, from Lemma C.6, the k -deep block of the chain C , b_k , was mined before round $r - r'$, and since block b_N was mined before block b_k , we have $r^* \leq r - r'$. Under the event $E_1[r - r', r]$, we know that

$$Y[r - r' - a, r + b] > Z[r - r' - a, r + b] + \frac{(1 - 2\beta)k}{8} \quad \forall a, b \geq 0.$$

Since $r^* \leq r - r'$ and $X[r^*, r] \geq Y[r^*, r]$, we obtain

$$X[r^*, r] > Z[r^*, r] + \frac{(1 - 2\beta)k}{8},$$

and this contradicts Equation (20) for $\mu = \frac{7+2\beta}{8}$. Therefore in the interval $[r^*, r]$, at least $(1 - \mu)k$ blocks on $C[-N + 1]$ were mined by honest users. These blocks must be in $C[-k]$ by definition of N . \square

Since the common-prefix, chain-quality and chain-growth properties are all proved assuming the event $E[r - r', r]$ occurs, a natural question is how likely is its occurrence? The next lemma shows that the probability of it occurring approaches 1 exponentially as r' increases. This lemma will be heavily used in our analysis of security and fast confirmation.

LEMMA C.10. Let $\bar{f} \leq \frac{\log(2-2\beta)}{1-\beta}$.⁸ For any r , $\mathbb{P}(E^c[r - r', r]) \leq 4e^{-\gamma\bar{f}r'}$, where $r' = \frac{k}{2\bar{f}}$ and $\gamma = \frac{1}{36}(1 - 2\beta)^2$.

PROOF. The event $E^c[r - r', r]$ is a union of three events. We will upper bound the probability each of these events separately and then use union bound.

LEMMA C.11. For any r , $\mathbb{P}\left(E_1^c[r - r', r]\right) \leq 2e^{-\frac{(1-2\beta)^2\bar{f}r'}{36}}$. Here $r' = \frac{k}{2\bar{f}}$.

PROOF. Let us restate the event $E_1[r - r', r]$ by substituting $k = 2r'\bar{f}$:

$$E_1[r - r', r] := \bigcap_{a, b \geq 0} \left\{ Y[r - r' - a, r + b] - Z[r - r' - a, r + b] > \frac{(1 - 2\beta)\bar{f}r'}{4} \right\}.$$

Observe that the random variable $Y[r - r' - a, r + b] - Z[r - r' - a, r + b]$ can be interpreted the position of a 1-d random walk (starting at the origin) after $r' + a + b$ steps. Here $Y[r - r' - a, r + b]$, $Z[r - r' - a, r + b]$ are the number of steps taken in right and left direction respectively. The value of \bar{f} is chosen s.t the random variables $Y[r - r' - a, r + b] \sim \text{Bin}(r' + a + b, \frac{\bar{f}}{2})$ and as seen before $Z[r - r' - a, r + b] \sim \text{Poiss}((r' + a + b)\bar{f}\beta)$; the random walk has $\frac{\bar{f}(1-2\beta)}{2}$ positive bias per step. In this random walk analogy, event $E_1[r - r', r]$ implies that the random walk is to the right of the point $\frac{(1-2\beta)\bar{f}}{4}$ after first r' steps and remains to the right of that point in all the future steps. We analyze this event by breaking in into two events.

Define a new event $D[r - r', r] = \{Y[r - r', r] - Z[r - r', r] < \frac{1}{3}(1 - 2\beta)\bar{f}r'\}$. In our random walk analogy, this event corresponds to a random walk which starts at the origin and is to the left of the point $\frac{1}{3}(1 - 2\beta)\bar{f}r'$ after r' steps. We upper bound the probability of the event $D[r - r', r]$:

$$\begin{aligned} \mathbb{P}(D[r - r', r]) &= \mathbb{P}(Y[r - r', r] - Z[r - r', r] < \frac{1}{3}(1 - 2\beta)\bar{f}r') \\ &= \mathbb{P}\left(Y[r - r', r] - Z[r - r', r] - \frac{1}{2}(1 - 2\beta)\bar{f}r' < -\frac{1}{6}(1 - 2\beta)\bar{f}r'\right) \\ &\stackrel{(a)}{\leq} e^{-\gamma_1\bar{f}r'}. \end{aligned} \tag{21}$$

The inequality (a) follows by applying Chernoff bound and the value of γ_1 is $\frac{1}{36}(1 - 2\beta)$. We will now use the event $D[r - r', r]$ to calculate the probability of the event $E_1^c[r - r', r]$:

$$\begin{aligned} \mathbb{P}(E_1^c[r - r', r]) &= \mathbb{P}(E_1^c[r - r', r] \cap D[r - r', r]) + \mathbb{P}(E_1^c[r - r', r] \cap D^c[r - r', r]) \\ &\leq \mathbb{P}(D[r - r', r]) + \mathbb{P}(E_1^c[r - r', r] \mid D^c[r - r', r]) \\ &\stackrel{(a)}{\leq} e^{-\gamma_1\bar{f}r'} + e^{-\gamma_2\bar{f}r'} \\ &\leq 2e^{-\gamma\bar{f}r'}. \end{aligned} \tag{22}$$

⁸We will assume this constraint in all our results without stating it explicitly.

In our random walk analogy, the event $\{E_1^c [r - r', r] \cap D^c [r - r', r]\}$ corresponds to a positive biased random walk $Y [r - r', r] - Z [r - r', r]$ starting to the right of the point $\frac{1}{3}\bar{f}(1 - 2\beta)r'$ and hitting the point $\frac{1}{4}\bar{f}(1 - 2\beta)r'$ in a future round. This event is analyzed in Lemma G.3 and using this lemma we obtain inequality (a) with $\gamma = \gamma_2 = \frac{1}{36}(1 - 2\beta)^2$. \square

LEMMA C.12. For any r , $\mathbb{P}\left(E_2^c [r - r', r]\right) \leq e^{-\bar{f}r'}$. Here $k = r'\bar{f}$.

PROOF. Let us restate the event $E_2 [r - r', r]$ by substituting $k = 2r'\bar{f}$:

$$E_2 [r - r', r] := \{X [r - r', r] + Z [r - r', r] < 2\bar{f}r'\}.$$

As defined in Section 2, the total number of block mined by the honest users in interval $[r - r', r]$ is $H[r - r', r] \sim \text{Poiss}((1 - \beta)\bar{f}r')$ and we have $H[r - r', r] \geq X[r - r', r]$. Using this we have

$$\begin{aligned} \mathbb{P}\left(E_2^c [r - r', r]\right) &= \mathbb{P}\left(H [r - r', r] + Z [r - r', r] \geq 2\bar{f}r'\right) \\ &= \mathbb{P}\left(\text{Poiss}((1 - \beta)\bar{f}r') + \text{Poiss}(\beta\bar{f}r') \geq 2\bar{f}r'\right) \\ &= \mathbb{P}\left(\text{Poiss}(\bar{f}r') \geq 2\bar{f}r'\right) \\ &\leq e^{-\bar{f}r'}. \end{aligned}$$

The last inequality follows from Chernoff bound⁹. \square

LEMMA C.13. For any r , $\mathbb{P}\left(E_3^c [r - r', r]\right) \leq e^{-\frac{\bar{f}r'}{36}}$. Here $r' = \frac{k}{2\bar{f}}$.

PROOF. Let us restate the event $E_3 [r - r', r]$ by substituting $k = 2r'\bar{f}$:

$$E_3 [r - r', r] := \left\{X [r - r', r] > \frac{\bar{f}r'}{3}\right\}.$$

We know that $Y [r - r', r] \leq X [r - r', r]$ and $Y [r - r', r] \sim \text{Bin}(r', \frac{\bar{f}}{2})$. Thus we have

$$\begin{aligned} \mathbb{P}\left(E_3^c [r - r', r]\right) &= \mathbb{P}\left(X [r - r', r] < \frac{\bar{f}r'}{3}\right) \\ &\leq \mathbb{P}\left(Y [r - r', r] < \frac{\bar{f}r'}{3}\right) \\ &= \mathbb{P}\left(\text{Bin}(r', \frac{\bar{f}}{2}) < \frac{\bar{f}r'}{3}\right) \\ &\leq e^{-\frac{\bar{f}r'}{36}}. \end{aligned}$$

The last inequality also follows from Chernoff bound. \square

Combining Lemmas C.11, C.12 and C.13, we obtain

$$\begin{aligned} \mathbb{P}\left(E^c [r - r', r]\right) &\leq \mathbb{P}\left(E_1^c [r - r', r]\right) + \mathbb{P}\left(E_2^c [r - r', r]\right) + \mathbb{P}\left(E_3^c [r - r', r]\right) \\ &\leq 2e^{-\frac{(1-2\beta)^2\bar{f}r'}{36}} + e^{-\bar{f}r'} + e^{-\frac{\bar{f}r'}{36}} \\ &\leq 4e^{-\frac{(1-2\beta)^2\bar{f}r'}{36}}. \end{aligned}$$

\square

⁹<https://github.com/ccanonne/probabilitydistributiontoolbox/blob/master/poissonconcentration.pdf>

D TOTAL ORDERING FOR PRISM: PROOFS OF THEOREMS 4.1 AND 4.2

In Appendix C, we proved three chain properties – chain-growth, common-prefix and chain-quality – for the Bitcoin backbone under events defined in Equation (18). The voter blocktrees in Prism also follow the longest chain protocol, hence these three chain properties will directly hold for each of the m voter blocktree under the corresponding events:

Similar to section C.1, let $H_j[r]$ and $Z_j[r]$ be the number of blocks mined by the honest nodes and by the adversarial node in round r on the i^{th} voter tree for $j \in [m]$. From section 2 and the sorition technique, we know that $H_j[r], Z_j[r]$ are Poisson random variables with means $(1 - \beta)f_v\Delta$ and $\beta f_v\Delta$ respectively. Note that random variables $\{H_j[r]\}_{r \in \{0, r_{\max}\}, j \in [m]}, \{Z_j[r]\}_{r \in \{0, r_{\max}\}, j \in [m]}$ are independent of each other. We now define auxiliary random variables $X_j[r]$ and $Y_j[r]$ as follows: If at round r an honest party mines at least one block on voter tree j , then $X_j[r] = 1$, otherwise $X_j[r] = 0$. If at round r an honest party mines exactly one block on voter tree i , then $Y_j[r] = 1$, otherwise $Y_j[r] = 0$. Let $r' = \frac{k}{2\bar{f}}$.

$$\begin{aligned} E_{1,j}[r - r', r] &:= \bigcap_{a, b \geq 0} \left\{ Y_j[r - r' - a, r + b] - Z_j[r - r' - a, r + b] > \frac{(1 - 2\beta)k}{8} \right\} \\ E_{2,j}[r - r', r] &:= \{H_j[r - r', r] + Z_j[r - r', r] < k\} \\ E_{3,j}[r - r', r] &:= \left\{ X_j[r - r', r] > \frac{k}{6} \right\} \\ E_j[r - r', r] &:= E_{1,j}[r - r', r] \cap E_{2,j}[r - r', r] \cap E_{3,j}[r - r', r]. \end{aligned} \quad (23)$$

Note the similarity between the above events and events defined in Equation (18). We know that $X_j[r - r', r]$ and $Y_j[r - r', r]$ are the number successful and uniquely successful rounds respectively in the interval $[r - r', r]$ on the blocktree j . Along the same lines, $Z_j[r - r', r]$ is the number of voter blocks mined by the adversary on the blocktree j in the interval $[r - r', r]$. Events $E_{1,j}[r - r', r]$, $E_{2,j}[r - r', r]$ and $E_{3,j}[r - r', r]$ have corresponding interpretation of the events $E_1[r - r', r]$, $E_2[r - r', r]$ and $E_3[r - r', r]$.

Typical event: For a given r' , define the following event:

$$E_j(r') := \bigcap_{\tilde{r} \geq r'} \bigcap_{0 \leq r \leq r_{\max}} E_j[r - \tilde{r}, r]. \quad (24)$$

LEMMA D.1. For any j , $\mathbb{P}\left(E_j^c(r')\right) \leq 4r_{\max}^2 e^{-\gamma \bar{f}_v r'}$, where $\gamma = \frac{1}{36}(1 - 2\beta)^2$.

PROOF. Use Lemma C.10 and apply union bound. \square

Let the first proposer block at level ℓ appear in round R_ℓ . We will now prove common-prefix and chain-quality for the leader block sequence defined in Equation (11).

Common prefix property: The common-prefix property of the leader sequence gives us the *confirmation policy*. We derive this property using the common prefix and the chain-quality properties of the voter blocks. Refer Figure 13.

LEMMA D.2 (COMMON-PREFIX). At round $r \geq R_\ell$, if every voter blocktree has a voter block mined by the honest users after round R_ℓ which is at least k -deep, then w.p $1 - \varepsilon_k$, the leader block sequence up to level ℓ is permanent i.e.,

$$\text{LedSeq}_\ell(r) = \text{LedSeq}_\ell(r_{\max}).$$

Here $\varepsilon_k \leq 4mr_{\max}^2 e^{-\gamma k/2}$ and $\gamma = \frac{1}{36}(1 - 2\beta)^2$.

PROOF. Fix a voter blocktree j and denote its k -deep voter block in round r by b_j . From the definition in Equation (24) and common-prefix Lemma C.8 we know for under the event $E_j(r')$, for $r' = \frac{k}{2\bar{f}_v}$, the k -deep voter block and its ancestors permanently remain on the main chain of voter blocktree j . From Lemma D.1 we know that $\mathbb{P}\left(E_j^c(r')\right) \leq \frac{\varepsilon_k}{m}$. Therefore, the k -deep voter block on the voter blocktree j is permanent w.p $1 - \frac{\varepsilon_k}{m}$. On applying union bound we conclude *all* the k -deep voter block on the m voter blocktrees are permanent w.p

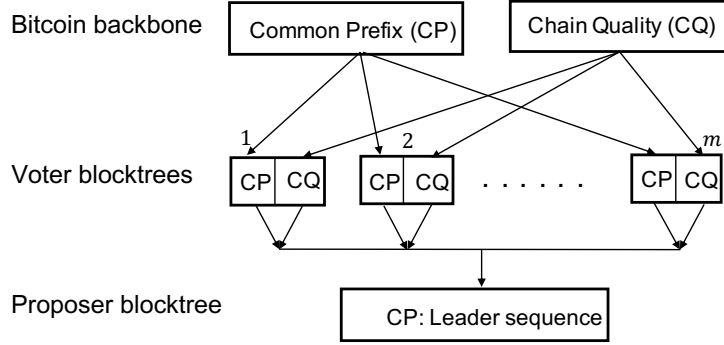


Figure 13: Dependencies of properties required to prove the common-prefix property of the leader sequence.

$1 - \varepsilon_k$. Each of these voter blocks, b_j 's, are mined by the honest users after round R_ℓ . Therefore, by the voter mining policy defined in Section 3, the main chain of the voter blocktree j until voter block b_j has votes on proposer blocks on all the levels $\ell' \leq \ell$ and all these votes are permanent w.p $1 - \varepsilon_k$. Therefore, for each level $\ell' \leq \ell$ has m permanent votes and this implies that the leader block at level ℓ' is also permanent w.p $1 - \varepsilon_k$. \square

Therefore, to confirm leader blocks with $1 - \varepsilon$ security, votes on all the m voter blocktrees should be at least $k = \frac{2}{\gamma} \log \frac{4mr_{\max}}{\varepsilon}$ deep. The natural question is: how long does it take to have (at least) k -deep votes on *all* m voter blocktrees? The next lemma answers this question.

LEMMA D.3. *By round $R_\ell + r_k$, wp $1 - \varepsilon'_k$, all the voter blocktrees have an honest voter block mined after round R_ℓ and is at least k -deep, where $r_k \leq \frac{64k}{(1-2\beta)\bar{f}_v}$ and $\varepsilon'_k \leq 8mr_{\max}^2 e^{-\frac{\gamma\bar{f}_v r_k}{8}}$.*

PROOF. Fix a blocktree j . Using the chain growth Lemma C.7 under the event $E_j(r_k)$, we know that the main chain of voter blocktree j grows by $k_1 \geq \frac{r_k \bar{f}_v}{3}$ voter blocks. Next, using the chain-quality Lemma C.9 under the second event $E_j\left(\frac{k_1}{2\bar{f}_v}\right)$, we know that at least $\frac{1-2\beta}{8}$ fraction of these k_1 voter blocks are mined by the honest users and the earliest of these voter block, say b_j , is at least k_2 -deep, where $k_2 \geq \frac{(1-2\beta)k_1}{8} \geq \frac{(1-2\beta)\bar{f}_v r_k}{24} := k$. It is important to note that the depth k_2 is *observable* by all the users. The probability of failure of either of these two events is

$$\begin{aligned}
\mathbb{P}\left(E_j^c(r_k) \cup E_j^c\left(\frac{k_1}{2\bar{f}_v}\right)\right) &\leq \mathbb{P}\left(E_j^c(r_k)\right) + \mathbb{P}\left(E_j^c\left(\frac{k_1}{2\bar{f}_v}\right)\right) \\
&\stackrel{(a)}{\leq} \mathbb{P}\left(E_j^c(r_k)\right) + \mathbb{P}\left(E_j^c\left(\frac{r_k}{6}\right)\right) \\
&\stackrel{(b)}{\leq} 2\mathbb{P}\left(E_j^c\left(\frac{r_k}{6}\right)\right) \\
&\stackrel{(c)}{\leq} \frac{\varepsilon'_k}{m}.
\end{aligned} \tag{25}$$

From Lemma D.1, we see that as r' decreases, $\mathbb{P}\left(E_j^c(r')\right)$ increases, and because $\frac{k_1}{2\bar{f}_v} \geq \frac{r_k}{6}$, we have the inequality (a). The inequality (b) also follows by the same logic. The last inequality (c) is given by Lemma D.1. Now applying union bound on Equation (25) over m blocktree gives us the required result. \square

Proof of Theorem 4.1:

PROOF. From Lemma D.3 we know that by round $R_\ell + r(\varepsilon)$, all the voter blocktrees will have a k -deep honest voter blocks w.p at least $1 - \frac{\varepsilon}{2}^{10}$ for $k \geq \frac{2}{\gamma} \log \frac{8mr_{\max}^2}{\varepsilon}$. Now applying Lemma D.2 for $k \geq \frac{2}{\gamma} \log \frac{8mr_{\max}^2}{\varepsilon}$, we obtain that all these honest voter blocks are permanent w.p $1 - \frac{\varepsilon}{2}$. On combining these two, we obtain that by round $R_\ell + r(\varepsilon)$ the leader block sequence up to level ℓ is permanent w.p $1 - \varepsilon$. \square

Worst Case vs Average Case: The confirmation policy in Lemma D.3 is stated for the worst case adversarial attack: when there are two (or more) proposer blocks at a given level have equal number of votes. Consider an ‘average case’ scenario with two proposer blocks at a level, where the first block has $2m/3$ votes and the second block as $m/3$ votes. In this scenario one can intuitively see that we don’t need to guarantee permanence of all the m votes but a weaker guarantee suffices: permanence of $m/6$ of the $2m/3$ votes of first block. This weaker guarantee can be achieved within a few rounds and translates to short latency in Prism.

COROLLARY D.4. Bitcoin’s latency is the time required to mine a single honest $\frac{1}{\varepsilon}$ -deep block on a voter chain of Prism and it is lesser than $\frac{2304}{\bar{f}_v(1-2\beta)^2} \log \frac{8r_{\max}^2}{\varepsilon}$ rounds to provide $1 - \varepsilon$ reliability to confirm blocks and the transactions in it.

Definition D.5 (Leader-sequence-quality). The (μ, k) -leader-sequence-quality property holds at round r if at most μ fraction of the last k consecutive leader blocks on the proposer blocktree at round r are mined by the adversary.

Similar to section C.1, let $H^P[r]$ and $Z^P[r]$ be the number of blocks mined by the honest nodes and by the adversarial node on proposer tree in round r . From section 2 along with sortition technique, we know that $H^P[r], Z^P[r]$ are Poisson random variables with means $(1 - \beta)f_v\Delta$ and $\beta f_v\Delta^{11}$ respectively. Note that random variables $\{H^P[r]\}_{r \in \{0, r_{\max}\}}$, $\{Z^P[r]\}_{r \in \{0, r_{\max}\}}$ are independent of each other. We now define auxiliary random variables $X^P[r]$ and $Y^P[r]$ as follows: If at round r an honest party mines at least one block on the proposer tree, then $X^P[r] = 1$, otherwise $X^P[r] = 0$. If at round r an honest party mines exactly one block on the proposer tree, then $Y^P[r] = 1$, otherwise $Y^P[r] = 0$. Let $r' = \frac{k}{2\bar{f}}$.

Let us define the following events on the proposer blocktree:

$$\begin{aligned} E_1^P[r - r', r] &:= \bigcap_{a, b \geq 0} \left\{ Y^P[r - r' - a, r + b] - Z^P[r - r' - a, r + b] > \frac{(1 - 2\beta)k}{8} \right\} \\ E_2^P[r - r', r] &:= \{ H^P[r - r', r] + Z^P[r - r', r] < k \} \\ E_3^P[r - r', r] &:= \left\{ X^P[r - r', r] > \frac{k}{6} \right\} \\ E^P[r - r', r] &:= E_1^P[r - r', r] \cap E_2^P[r - r', r] \cap E_3^P[r - r', r]. \end{aligned} \quad (26)$$

We know that $X^P[r - r', r]$ and $Y^P[r - r', r]$ are the number of users in successful and uniquely successful rounds respectively in the interval $[r - r', r]$ on proposer blocktree, and $Z^P[r - r', r]$ is the number of proposer blocks mined by adversary in the interval $[r - r', r]$. These events have corresponding interpretation of the events defined in Equations (18) and (23).

LEMMA D.6 (LEADER-SEQUENCE-QUALITY). *The (μ, k) -leader-sequence-quality property holds at round r for $\mu = \frac{7+2\beta}{8}$ w.p at least $1 - 4r_{\max}^2 e^{-(1-2\beta)^2 k/72}$.*

PROOF. Unlike the longest chain in Bitcoin, the leader sequence in Prism does not form a chain. Therefore, one cannot directly use Lemma C.9 and we need to adapt its proof to prove the required property here.

¹⁰ $8mr_{\max}^2 e^{-\frac{\gamma \bar{f}_v}{8} \frac{64}{\gamma \bar{f}_v (1-2\beta)} \log \frac{8mr_{\max}^2}{\varepsilon}} = 8mr_{\max}^2 e^{-\frac{8}{1-2\beta} \log \frac{8mr_{\max}^2}{\varepsilon}} < \frac{\varepsilon}{2}$.
¹¹ $f_v = f_p$

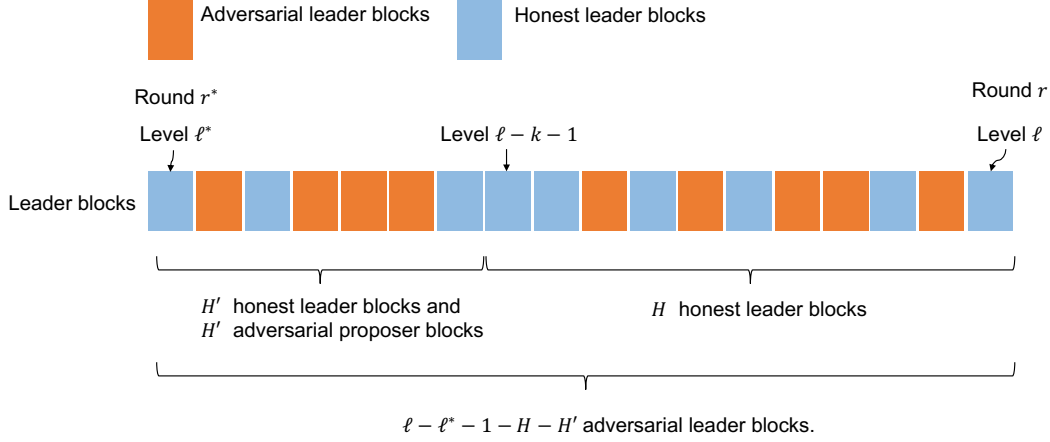


Figure 14: The leader blocks in levels $[\ell^*, \ell]$.

Let r be the current round and ℓ be the last level on the proposer blocktree which has proposer blocks at round r . Consider the k consecutive leader blocks on levels $[\ell - k, \ell] := \{\ell - k + 1, \dots, \ell\}$ on the leader sequence $\text{LedSeq}_\ell(r)$ and define:

$$\ell^* := \max \{ \tilde{\ell} \leq \ell - k + 1 \text{ s.t. the honest users mined the first proposer block on level } \tilde{\ell} \}$$

Let r^* be the round in which the first proposer block was mined on level ℓ^* and define the interval $S := \{r^* < i \leq r\} = [r^*, r]$. From the definition of ℓ^* we have the following two observations:

- (1) The adversary has mined at least one proposer block on all levels in $[\ell^*, \ell - k + 1]$.
- (2) All the proposer blocks on levels $[\ell^*, \ell]$ are mined in the interval S because there are no proposer blocks on level ℓ^* before round r^* and hence no user can mine a proposer block on a level greater than ℓ^* before round r^* .

Let H be the number of honest leader blocks on the levels $[\ell - k, \ell]$ and say

$$H < (1 - \mu)k. \tag{27}$$

Let H' be the number of honest leader blocks on the levels $[\ell^*, \ell - k]$. The adversary has mined $\ell - \ell^* - 1 - H - H'$ leader blocks in the interval S . From our first observation, we know that the number of proposer blocks mined by the adversary on the levels $[\ell^*, \ell - k]$ which are *not* leader blocks is at least H' , and from our second observation, these proposer blocks are mined in the interval S . Therefore, the number of proposer blocks mined by the adversary in the interval S satisfies

$$\begin{aligned} Z^P[r^*, r] &\geq (\ell - \ell^* - H - H' - 1) + H' \\ &\geq \ell - \ell^* - 1 - H \\ \text{(From Equation (27))} &> \ell - \ell^* - 1 - (1 - \mu)k. \end{aligned} \tag{28}$$

Refer Figure 14 for an illustration. From the chain growth Lemma C.7, we know that $\ell - \ell^* - 1 \geq X^P[r^*, r]$ and combining this with Equation (28) gives us

$$Z^P[r^*, r] > X^P[r^*, r] - (1 - \mu)k. \tag{29}$$

Let $r' := \frac{k}{2f_v}$. Define an event $E^P(r') := \bigcap_{\tilde{r} \geq r'} \bigcap_{r \leq r_{\max}} E^P[r - \tilde{r}, r]$ and assume the event $E^P(r')$ occurs. Under the event $E_1^P[r - r', r] \supseteq E^P(r')$, we know that

$$Y^P[r - r' - a, r + b] > Z^P[r - r' - a, r + b] + \frac{(1 - 2\beta)k}{8} \quad \forall a, b \geq 0.$$

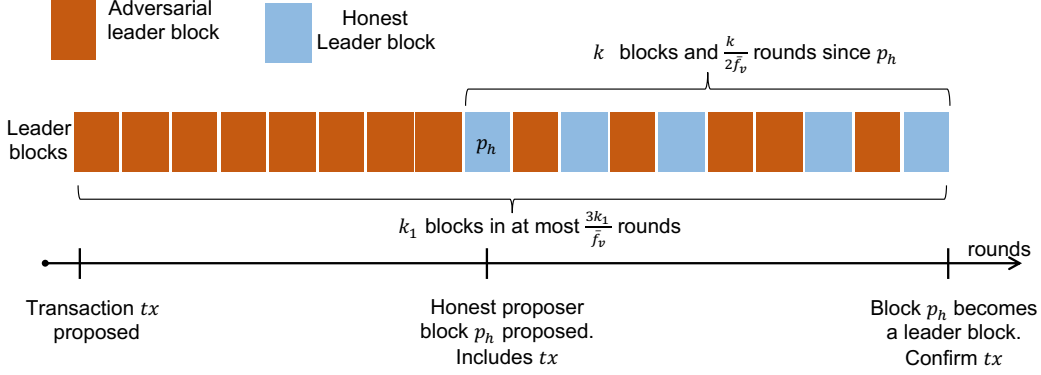


Figure 15

The first proposer block on the level ℓ is mined before round r . Under the event $E_2^P[r - r', r] \supseteq E^P(r')$, from Lemma C.6, the first proposer block on level $\ell - k + 1$ was mined before round $r - r'$, and hence $r^* \leq r - r'$. This combined with $X^P[r^*, r] \geq Y^P[r^*, r]$, gives us

$$X^P[r^*, r] > Z^P[r^*, r] + \frac{(1 - 2\beta)k}{8},$$

and this contradicts Equation (29) for $\mu = \frac{7+2\beta}{8}$. Therefore on the levels $[\ell - k, \ell]$, at least $\frac{1-2\beta}{8}$ fraction of the leader blocks are mined by honest users. From Lemma D.1, we know that the event $E^P(r')$ occurs w.p $1 - 4r_{\max}^2 e^{-\gamma k/2}$, where $\gamma = \frac{1}{36}(1 - 2\beta)^2$, and this completes the proof. \square

The leader sequence quality defined in D.5 is parameterized by two parameters r and k , whereas its counterpart definition of chain quality in [9], is parameterized only by a single parameter k . Even though our definition of ‘quality’ is a weaker, we show that it suffices to ensure liveness.

Proof of Theorem 4.2:

PROOF. Let $k := \frac{2048}{(1-2\beta)^3} \log(\frac{32mr_{\max}}{\epsilon})$ and $k_1 := \frac{8k}{1-2\beta}$. Using Lemma D.6 we know that w.p at least $1 - \epsilon/4$, the last k_1 leader blocks have at least k honest leader blocks. From Lemma C.6 and D.1, w.p at least $1 - \epsilon/4$, the deepest of these k honest leader block was proposed before the round $r - \frac{k}{2\bar{f}_v}$. Here $\frac{k}{2\bar{f}_v} = \frac{1024}{(1-2\beta)^3 \bar{f}_v} \log(\frac{32mr_{\max}}{\epsilon})$ and now using Theorem 4.1, this deepest honest leader block is permanent w.p $1 - \epsilon/4$. Therefore, the honest transaction will be permanently added to the blockchain after k_1 proposer blocks are mined. Using chain growth Lemma C.7, we know that the k_1 proposer blocks will be mined in no more than $\frac{3k_1}{\bar{f}_v}$ rounds w.p $1 - \epsilon/4$. Therefore, w.p $1 - \epsilon$, the transaction will be part of the permanent leader sequence in

$$\frac{3k_1}{\bar{f}_v} = \frac{3 \times 2^{14}}{(1 - 2\beta)^3 \bar{f}_v} \log \frac{32mr_{\max}}{\epsilon} \text{ rounds.} \quad (30)$$

Refer Figure 15 for an illustration. Note that the constants in Equation (30) have not be optimized for the sake of readability. The scaling w.r.t $1 - 2\beta$, \bar{f}_v and $\log \frac{1}{\epsilon}$ is the main take away. \square

E FAST LIST CONFIRMATION FOR PRISM: PROOF OF THEOREM 4.5 AND 4.6

E.1 Voter chain properties

In Appendix D, we proved the common-prefix and the leader-sequence-quality properties by requiring the typical event defined in Equation (24) to hold for every voting chain, i.e. at the *microscopic* scale. The typicality of

each such event was obtained by averaging over rounds and as a consequence the confirmation of leader blocks with $1 - \varepsilon$ guarantee required averaging over $O(\log \frac{1}{\varepsilon})$ rounds. In this section we obtain faster confirmation time by relaxing the notion of typicality to a notion of *macroscopic* typicality, one which concerns the mining processes of a large fraction of the voter chains. This event guarantees macroscopic versions of the chain-growth, common-prefix and chain-quality properties. That is, these properties are guaranteed to be satisfied by a large fraction of the voter chains, but *not* all. These macroscopic properties of the voting chains turn out to be sufficient to allow fast confirmation. For this section, we will define:

$$\begin{aligned}
\gamma &:= \frac{1}{36}(1 - 2\beta)^2 \\
c_1 &:= \frac{1 - 2\beta}{16} \\
r_{\min} &:= \frac{2 \log \frac{200}{\gamma c_1}}{\gamma \bar{f}_v} \\
k_{\min} &:= \frac{4}{\gamma} \log \frac{200}{\gamma c_1} \\
\rho_{r'} &:= \max\left(\frac{c_1}{1 + 4\bar{f}_v r'}, \frac{(1 - 2\beta)c_1}{1 + 32 \log m}\right) \\
\delta_k &:= \max\left(\frac{c_1}{1 + 2k}, \frac{(1 - 2\beta)c_1}{1 + 32 \log m}\right) \\
\varepsilon_m &:= r_{\max}^2 e^{-\frac{(1-2\beta)c_1 m}{2+64 \log m}}. \tag{31}
\end{aligned}$$

LEMMA E.1 (MACROSCOPIC TYPICALITY). *The macroscopic typical event \mathbb{T} defined below occurs with probability $1 - \varepsilon_m$.*

$$\begin{aligned}
\mathbb{T}[r - r', r] &:= \left\{ \frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j^c[r - r', r]) \geq 1 - \delta_k \right\} \\
\mathbb{T} &:= \bigcap_{0 \leq r \leq r_{\max}, r' \geq r_{\min}} \mathbb{T}[r - r', r],
\end{aligned}$$

where $r' = \frac{k}{2\bar{f}_v}$. Note that $\delta_k = \rho_{r'}$.

PROOF. For a fixed r, r' , the indicator random variables $\mathbf{1}(\mathbb{E}_j^c[r - r', r])$ are identical and independent $\forall j \in [m]$. The mean of the random variable $\mathbf{1}(\mathbb{E}_j^c[r - r', r])$ is μ , and it is at most $4e^{-\gamma \bar{f}_v r'}$ by Lemma C.10. Using Chernoff bound¹² for Bernoulli random variables, $\forall a \geq 0$, we have

$$\begin{aligned}
&\mathbb{P}\left\{\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j^c[r - r', r]) \geq \mu + a\right\} \leq e^{-\frac{ma^2}{a+2\mu}} \\
&\stackrel{(1)}{\Rightarrow} \mathbb{P}\left\{\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j^c[r - r', r]) \geq 4e^{-\gamma \bar{f}_v r'} + a\right\} \leq e^{-\frac{ma^2}{a+4e^{-\gamma \bar{f}_v r'}}} \\
&\stackrel{(2)}{\Rightarrow} \mathbb{P}\left\{\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j^c[r - r', r]) \geq (\delta + 1)4e^{-\gamma \bar{f}_v r'}\right\} \leq e^{-4me^{-\gamma \bar{f}_v r'} \frac{\delta^2}{\delta+2}}. \tag{32}
\end{aligned}$$

¹²<http://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf>

Step (1) follows because $\mu \leq 4e^{-\gamma \bar{f}_v r'}$, and step (2) is obtained by substituting $a = \delta 4e^{-\gamma \bar{f}_v r'}$. For $r' \geq r_{\min}$ ¹³, we have $\frac{1}{4}e^{\gamma \bar{f}_v r'} \rho_{r'} > 10$. On substituting $\delta = \frac{1}{4}e^{\gamma \bar{f}_v r'} \rho_{r'} - 1$ in Equation (32), for all values of $r' \geq r_{\min}$, we get

$$\begin{aligned} \mathbb{P}(\mathbb{T}^c[r - r', r]) &= \mathbb{P}\left\{\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j^c[r - r', r]) \geq \rho_{r'}\right\} \leq e^{-m \rho_{r'} \frac{\delta^2}{(\delta+1)(\delta+2)}} \\ &\stackrel{(a)}{\leq} e^{-m \rho_{r'}/2} \\ &\stackrel{(b)}{\leq} e^{-\frac{(1-2\beta)c_1 m}{2+64 \log m}}. \end{aligned}$$

The inequality (a) follows from because $\delta > 9$ and inequality (b) follows because $\rho_{r'} > \frac{(1-2\beta)c_1}{1+32 \log m}$. Since r, r' can take at most r_{\max} values, the event \mathbb{T}^c is a union of at most r_{\max}^2 $\mathbb{T}^c[r - r', r]$ events. Using union bound we prove that the event \mathbb{T}^c occurs w.p at most $\varepsilon_m = r_{\max}^2 e^{-\frac{(1-2\beta)c_1 m}{2+64 \log m}}$ and this combined with $\delta_k = \rho_{r'}$ proves the required result. \square

LEMMA E.2 (MACROSCOPIC CHAIN-GROWTH). *Under the event \mathbb{T} , for $k \geq k_{\min}$ and $r' = \frac{k}{2\bar{f}_v}$, the longest chain grows by at least $\frac{k}{6}$ blocks in the interval $[r - r', r]$ on at least $1 - \delta_k$ fraction of voter blocktrees.*

PROOF. From the typicality Lemma E.1, we know that under the event $\mathbb{T}[r - r', r] \supseteq T$,

$$\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j[r - r', r]) \geq 1 - \delta_k.$$

Applying Lemma C.7 on events $\mathbb{E}_j[r - r', r]$ for $j \in [m]$ gives us the required result. \square

LEMMA E.3 (MACROSCOPIC COMMON-PREFIX). *Under the event \mathbb{T} , for $k \geq k_{\min}$ and $r' = \frac{k}{2\bar{f}_v}$, the k -deep common-prefix property holds at round r for at least $1 - \delta_k$ fraction of voter blocktrees.*

PROOF. From the typicality Lemma E.1, we know that under the event $\mathbb{T}[r - r', r] \supseteq T$,

$$\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j[r - r', r]) \geq 1 - \delta_k.$$

Applying Lemma C.8 on events $\mathbb{E}_j[r - r', r]$ for $j \in [m]$ gives us the required result. \square

LEMMA E.4 (MACROSCOPIC CHAIN-QUALITY). *Under the event \mathbb{T} , for $k \geq k_{\min}$ and $r' = \frac{k}{2\bar{f}_v}$, the (μ, k) -chain quality property holds at round r for $\mu = \frac{7+2\beta}{8}$ for at least $1 - \delta_k$ fraction of voter blocktrees.*

PROOF. From the typicality Lemma E.1, we know that under the event $\mathbb{T}[r - r', r] \supseteq T$,

$$\frac{1}{m} \sum_{j=1}^m \mathbf{1}(\mathbb{E}_j[r - r', r]) \geq 1 - \delta_k.$$

Applying Lemma C.7 on events $\mathbb{E}_j[r - r', r]$ for $j \in [m]$ gives us the required result. \square

In Appendix D, we used microscopic properties of each voter chain to obtain the common-prefix and the leader sequence quality properties for the blocktree. The voter chains require long interval of rounds to individually satisfy the microscopic properties and that results in large latency. Here we change use a different strategy: we use macroscopic properties of the voter chains to obtain the common-prefix and the leader sequence quality properties. The voter chains satisfy macroscopic properties for short interval of rounds and this directly translates to short latency.

¹³The value of r_{\min} was precisely chosen to satisfy this inequality.

E.2 Fast list confirmation policy

We repeat the definitions from Section 4.5.1. $\mathcal{P}_\ell(r) = \{p_1, p_2, \dots\}$ is the set of proposer blocks at level ℓ at round r . Let $U_\ell(r)$ be the number of voter blocktrees which have not voted for any proposer block in the set $\mathcal{P}_\ell(r)$. Let $V_n^k(r)$ be the number of votes at depth k or greater for proposer block p_n in round r . Let $V_{-n}^k(r)$ be the number of votes at depth k or greater for proposer blocks in the subset $\mathcal{P}_\ell(r) - \{p_n\}$. Note that $V_n^k(r)$ and $V_{-n}^k(r)$ are observable quantities. The following lemma bounds the future number of votes on a proposer block.

LEMMA E.5. *With probability at least $1 - \varepsilon_m$, the number of votes on any proposer block p_n in any future round $r_f \geq r$, $V_n(r_f)$, satisfies*

$$\underline{V}_n(r) \leq V_n(r_f) \leq \bar{V}_n(r),$$

where

$$\underline{V}_n(r) := \max_{k \geq k_{\min}} (V_n^k(r) - \delta_k m)_+, \quad (33)$$

$$\bar{V}_n(r) := V_n(r) + \left(V_{-n}(r) - \max_{k \geq k_{\min}} (V_{-n}^k(r) - \delta_k m)_+ \right) + U_\ell(r). \quad (34)$$

PROOF. From the typicality Lemma E.1, we know that the typical event T occurs w.p $1 - \varepsilon_m$. We will use this to prove $V_n(r_f) \geq (V_n^k(r) - \delta_k m)_+$ for all values of $k \geq k_{\min}$. For a fixed k , let $r' = \frac{k}{2f_v}$. Under the event T , from Lemma E.3, we know that the k -deep common-prefix property holds for at least $1 - \delta_k$ fraction of voter blocktrees. Therefore $V_n(r_f)$ is at least $(V_n^k(r) - \delta_k m)_+$ for all $r_f \geq r$. Since this holds for all values of $k \geq k_{\min}$, we have $\underline{V}_n(r) := \max_{k \geq k_{\min}} (V_n^k(r) - \delta_k m)_+$.

Following the same line of reasoning, $\underline{V}_{-n}(r) := \max_{k \geq k_{\min}} (V_{-n}^k(r) - \delta_k m)_+$ is a lower bound on $V_{-n}(r')$. Therefore, at most $(V_{-n}(r) - \underline{V}_{-n}(r))$ votes can be removed from proposer blocks in the set $\mathcal{P}_\ell(r) - \{p_n\}$ and added to the proposer block p_n . Also the $U_\ell(r)$ voter blocktrees which have not yet voted could also vote on block p_n . Combining these both gives us the upper bound on $V_n(r_f)$. \square

Any private block $p_{\text{private}} \notin \mathcal{P}_\ell(r)$ by definition has zero votes at round r . The future number of votes on the proposer block p_{private} w.p $1 - \varepsilon_m$ satisfies

$$V_{\text{private}}(r_f) \leq \bar{V}_{\text{private}}(r) := m - \sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r) \quad \forall r_f \geq r, \quad (35)$$

because each proposer block p_n has $\underline{V}_n(r)$ permanent votes w.p $1 - \varepsilon_m$ and p_{private} could potentially get the rest of the votes.

Fast list confirmation policy: If $\max_n \underline{V}_n(r) > \bar{V}_{\text{private}}(r)$, confirm the following proposer block list at level ℓ :

$$\Pi_\ell(r) := \{p_i : \bar{V}_i(r) > \max_n \underline{V}_n(r)\}. \quad (36)$$

Figures 9 illustrate one such example. The definition of $\Pi_\ell(r)$ is precisely designed to prevent private proposer blocks from becoming the leader blocks in the future rounds.

LEMMA E.6. *If the proposer lists are confirmed for all levels $\ell' \leq \ell$ by round r , then w.p $1 - \varepsilon_m$, the final leader sequence up to level ℓ satisfies*

$$p_{\ell'}^*(r_{\max}) \in \Pi_{\ell'}(r) \quad \forall \ell' \leq \ell.$$

PROOF. We prove by contradiction. Say the final leader block at level $\ell' \leq \ell$ is $p_{\ell'}^*(r_{\max}) = b_i$ and $b_i \notin \Pi_{\ell'}(r)$. Without loss of generality, let us assume proposer block p_1 has the largest $\underline{V}_n(r)$ in round r . We have

$$V_i(r_f) \stackrel{(a)}{\leq} \bar{V}_i(r) \stackrel{(b)}{<} \underline{V}_1(r) \stackrel{(c)}{\leq} V_1(r_f) \quad \forall r_f \geq r. \quad (37)$$

The inequality (b) is by definition of $\Pi_{\ell'}(r)$, and the inequalities (a) and (c) are due to confidence intervals from Lemma E.5. Equation (37) gives us $V_i(r_f) < V_1(r')$, and therefore the proposer block b_i cannot be the leader block

in any future rounds $r_f \geq r$, which includes the final round r_{\max} . Therefore, we have $p_{\ell'}^*(r_{\max}) \in \Pi_{\ell'}(r) \forall \ell' \leq \ell$ and this proves the required result. \square

Lemma E.6 proves that the proposer lists obtained via the fast list confirmation policy contains the final leader blocks. The natural question is: how long does it take to satisfy the constraint for the fast list confirmation? We answer this question next.

E.3 Latency

The first proposer block at level ℓ appears in round R_ℓ . For ease of calculations we assume that the proposer blocktree mining rate $\tilde{f}_p = \tilde{f}_v$. Define $\Delta_0 := \frac{12r_{\min}}{1-2\beta}$.

LEMMA E.7. *By round $r = R_\ell + \Delta_r$, for $\Delta_r \geq \Delta_0$, w.p $1 - \varepsilon_m$, at least $1 - 4\rho_{\Delta_r}$ fraction of the voter blocktrees have an honest voter block which is mined after round R_ℓ and is at least k -deep on the main chain. Here $k \geq \frac{(1-2\beta)\tilde{f}_v\Delta_r}{24}$ and is also greater than k_{\min} .*

PROOF. From the typicality Lemma E.1, we know that the event T occurs w.p $1 - \varepsilon_m$. Using the chain-growth Lemma E.2 under the event T , we know that by round r , $1 - \rho_{\Delta_r}$ fraction of the voter blocktree's main chain grows by $k_1 \geq \frac{\Delta_r\tilde{f}_v}{3}$ voter blocks. Let $r' = \frac{k_1}{2\tilde{f}_v}$. Next, using chain-quality Lemma E.4 under the event T , we know that for at least $1 - \delta_{k_1}$ fraction of voter blocktrees, the deepest of these honest voter blocks, mined after round R_ℓ , is at least k -deep, where $k \geq \frac{(1-2\beta)k_1}{8} \geq \frac{(1-2\beta)\tilde{f}_v\Delta_r}{24}$. Therefore, at least $1 - \rho_{\Delta_r} - \delta_{k_1}$ fraction of the blocktrees have an honest voter block mined after round R_ℓ which is at least k -deep on the main chain. The constants satisfy $\delta_{k_1} = \rho_{\frac{\Delta_r}{3}} < 3\rho_{\Delta_r}$, and this completes the proof. It is important to note that the depth of votes on all the m voter blocktree are *observable* by the users. \square

Define random variable $N_\ell(r) := |P_\ell(r)|$ as the number of proposer blocks on level ℓ at round r and let $c_1 = \frac{1-2\beta}{16}$ and $c_2 := \frac{16}{\tilde{f}_v(1-2\beta)^3}$.

LEMMA E.8. *The proposer list at level ℓ can be confirmed w.p $1 - \varepsilon_m$ in round $r = R_\ell + \Delta_r$ for for $\Delta_r \geq \Delta_0$ if*

$$\text{Case 1. } N_\ell(R_\ell + \Delta_r) + 1 < \frac{c_1}{\rho_{\Delta_r}}, \quad (38)$$

$$\text{Or Case 2. } \Delta_r = c_2 m.$$

PROOF. Let us first consider Case 1. All the events here are $1 - \varepsilon_m$ probability events. From Lemma E.7, we know that by round $r = R_\ell + \Delta_r$, at least $1 - 4\rho_{\Delta_r}$ fraction of voter blocktrees have k -deep votes on proposer blocks in $\mathcal{P}_\ell(r)$ where $k \geq \frac{(1-2\beta)\tilde{f}_v\Delta_r}{24}$. This implies $\sum_{p_n \in \mathcal{P}_\ell(r)} V_n^k(r) \geq m(1 - 4\rho_{\Delta_r})$ and from Lemma E.5, we have

$$\sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r) \geq m(1 - 4\rho_{\Delta_r} - \delta_k), \quad (39)$$

where the constant δ_k satisfies $\delta_k \leq \frac{12\rho_{\Delta_r}}{(1-2\beta)}$. Without loss of generality we assume $\underline{V}_1(r) \geq \underline{V}_i(r) \forall p_i \in \mathcal{P}_\ell(r)$, and therefore from Equation (39), we have

$$\underline{V}_1(r) \geq \frac{m}{N_\ell(r)} \left(1 - \frac{16\rho_{\Delta_r}}{1-2\beta} \right)^{14}. \quad (40)$$

On the other hand, the upper bound on the votes on a private proposer block, p_{private} , by Equation (35) is :

$$\begin{aligned} \bar{V}_{\text{private}}(r) &< m - \sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r) \\ &\stackrel{(a)}{<} \frac{(1-2\beta)\rho_{\Delta_r}}{16}, \end{aligned} \quad (41)$$

¹⁴Note that this inequality is extremely weak.

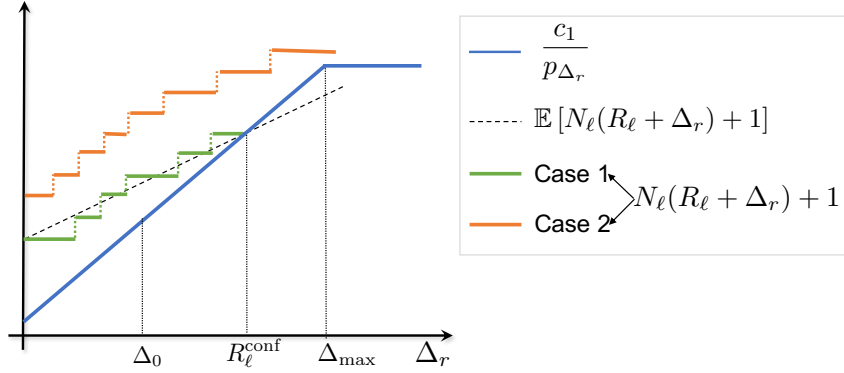


Figure 16: Sample paths of $\text{rv } N_\ell(R_\ell + \Delta_r) + 1$ falling under Case 1 and Case 2 in Lemma E.8. The values of $\frac{c_1}{\rho\Delta_r} = \min(1 + 4\bar{f}_v\Delta_r, 1 + 32\log m)$, $\Delta_0 = \frac{12}{1-2\beta}r_{\min}$, $\Delta_{\max} = 8\log m$.

where the inequality (a) follows from from Equation (39). From Equations (40) and (41), it is easy to see that

$$N_\ell(r) + 1 < \frac{16}{(1-2\beta)\rho\Delta_r} \implies \underline{V}_1(r) > \bar{V}_{\text{private}}(r),$$

and therefore, the proposer list at level ℓ can be confirmed by round r . This proves the claim in Case 1. Now let us consider Case 2. From the proof of Theorem 4.1, we know that *all* the m votes are permanent w.p $1 - \varepsilon$ for

$$r(\varepsilon) = \frac{1024}{\bar{f}_v(1-2\beta)^3} \log \frac{8mr_{\max}}{\varepsilon}.$$

Substituting $\varepsilon = \varepsilon_m$ in the above equation, we conclude that for $r(\varepsilon_m) = c_2m$, the upper bound on the number of votes on private block, $\bar{V}_{\text{private}}(r) = 0$ and $\underline{V}_1(R_\ell + k) \geq 1 > \bar{V}_{\text{private}}(R_\ell + k)$ w.p $1 - \varepsilon_m$. \square

We now use the above Lemma E.8 to calculate the expected number of rounds to confirm the proposer block list at level ℓ . For Case 1 (38) let us define the random variable:

$$R_\ell^{\text{stop}} := \min \Delta_r > \Delta_0 \text{ s.t } N_\ell(R_\ell + \Delta_r) + 1 < \frac{c_1}{\rho\Delta_r}. \quad (42)$$

Note that $R_\ell^{\text{stop}} = \infty$ if the inequality condition in Equation (42) is not satisfied for any Δ_r . From Lemma E.8, the proposer list at level ℓ can be confirmed in $\min(R_\ell^{\text{stop}}, c_2m)$ rounds and the next lemma calculates its expectation.

LEMMA E.9. *The proposer list at level ℓ can be confirmed by round $R_\ell + \min(R_\ell^{\text{stop}}, c_2m)$ and we have*

$$\mathbb{E}[\min(R_\ell^{\text{stop}}, c_2m)] \leq \frac{13}{(1-2\beta)}r_{\min} + \frac{48}{\bar{f}_v(1-2\beta)^3m^3}.$$

PROOF. The honest users do not mine new proposer blocks on level ℓ after round R_ℓ , however, the adversary could potentially mine new proposer blocks on level ℓ after round R_ℓ . Therefore, the random variable $N_\ell(R_\ell + \Delta_r)$ satisfies

$$N_\ell(R_\ell + \Delta_r) \leq H^P[R_\ell] + W_\ell^P(R_\ell) + Z_\ell^P[R_\ell, R_\ell + \Delta_r].$$

- (1) $H^P[R_\ell]$ corresponds to the number of proposer blocks mined by the honest users on level ℓ . From Section 2, we know that $H^P[R_\ell] \sim \text{Poiss}((1-\beta)\bar{f}_v)$.
- (2) $W_\ell^P(R_\ell)$ denotes the upper bound on number of proposer blocks at level ℓ in store by the adversary by round R_ℓ . It is shown in Appendix G.1 that $W_\ell^P(R_\ell) \sim \text{Geometric}(1-2\beta)$.

- (3) $Z_\ell^p[R_\ell, R_\ell + \Delta_r]$ denotes the number of proposer blocks mined by the adversary at level ℓ in the interval $[R_\ell, R_\ell + \Delta_r]$. From Section 2, we know that $Z_\ell^p[R_\ell, R_\ell + \Delta_r] \sim \text{Poiss}(\bar{f}_v \beta \Delta_r)$.

The mean of random variable $N_\ell(R_\ell + \Delta_r)$ is affine in Δ_r , and $\frac{c_1}{\rho \Delta_r}$ is also affine in Δ_r with a higher slope (by design). Therefore, intuitively the expected value of R_ℓ^{stop} defined in Equation (42) should be constant which depends only on β . Two examples are illustrated in Figure 16. We now formalize this intuition. Let us define $\Delta_{\max} = \frac{8 \log m}{\bar{f}_v(1-2\beta)}$. We will calculate $\mathbb{P}(R_\ell^{\text{stop}} > \Delta_r)$ separately for three intervals: $[0, \Delta_0]$, $(\Delta_0, \Delta_{\max})$, $[\Delta_{\max}, \infty)$.

- (1) *Interval* $[0, \Delta_0]$: Since $R_\ell^{\text{stop}} \geq \Delta_0$ by definition, we have

$$\mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) = 1 \quad \forall \Delta_r \leq \Delta_0. \quad (43)$$

- (2) *Interval* $[\Delta_{\max}, \infty)$: For $\Delta_r \geq \Delta_{\max}$, we have

$$\begin{aligned} \{R_\ell^{\text{stop}} > \Delta_r\} &= \bigcap_{x \leq \Delta_r} \left\{ N_\ell(R_\ell + x) + 1 > \frac{c_1}{\rho \Delta_r} \right\} \\ &\subseteq \bigcap_{x \leq \Delta_{\max}} \left\{ N_\ell(R_\ell + \Delta_{\max}) + 1 > \frac{c_1}{\rho \Delta_{\max}} \right\} \\ &= \{R_\ell^{\text{stop}} > \Delta_{\max}\}. \end{aligned} \quad (44)$$

This implies

$$\mathbb{P}\{R_\ell^{\text{stop}} > \Delta_r\} \leq \mathbb{P}\{R_\ell^{\text{stop}} > \Delta_{\max}\} \quad \forall \Delta_{\max} \leq \Delta_r. \quad (45)$$

- (3) *Interval* $(\Delta_0, \Delta_{\max})$: Using Equation (31), we have

$$\frac{c_1}{\rho \Delta_r} = 1 + 4\bar{f}_v \Delta_r \quad \forall \Delta_r < \Delta_{\max}. \quad (46)$$

For $\Delta_0 < \Delta_r < \Delta_{\max}$, we bound the tail event:

$$\begin{aligned} \{R_\ell^{\text{stop}} > \Delta_r\} &= \bigcap_{x \leq \Delta_r} \left\{ N_\ell(R_\ell + x) + 1 > \frac{c_1}{\rho \Delta_r} \right\} \\ &\subseteq \left\{ N_\ell(R_\ell + \Delta_r) + 1 > \frac{c_1}{\rho \Delta_r} \right\} \\ &\subseteq \left\{ H^p[R_\ell] + W_\ell^p(R_\ell) + Z_\ell^p[R_\ell, R_\ell + \Delta_r] + 1 > \frac{c_1}{\rho \Delta_r} \right\} \\ &= \left\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \right. \\ &\quad \left. > \frac{c_1}{\rho \Delta_r} - (1 + \mathbb{E}[H^p[R_\ell]] + \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \right\} \\ &\stackrel{(a)}{=} \left\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \right. \\ &\quad \left. > 1 + 4\bar{f}_v \Delta_r - (1 + (1 - \beta)\bar{f}_v + \beta\bar{f}_v \Delta_r) \right\} \\ &\subseteq \left\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \right. \\ &\quad \left. > (\bar{f}_v \Delta_r + \bar{f}_v \Delta_r + \bar{f}_v \Delta_r) \right\} \\ \Rightarrow \{R_\ell^{\text{stop}} > \Delta_r\} &\subseteq F_1 \cup F_2 \cup F_3, \end{aligned} \quad (47)$$

where the events are:

$$\begin{aligned} F_1 &:= \{H^P[R_\ell] - \mathbb{E}[H^P[R_\ell]] \geq \bar{f}_v \Delta_r\} \\ F_2 &:= \{W_\ell^P(R_\ell) \geq \bar{f}_v \Delta_r\} \\ F_3 &:= \{Z_\ell^P[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^P[R_\ell, R_\ell + \Delta_r]] > \bar{f}_v \Delta_r\}. \end{aligned}$$

The equality (a) follows from Equation (46). Using Chernoff bounds, we upper bound the probabilities of events the F_1 , F_2 and F_3 :

$$\mathbb{P}(F_1) \leq e^{-\frac{\bar{f}_v \Delta_r}{2}} \quad (48)$$

$$\mathbb{P}(F_2) \leq (2\beta)^{\bar{f}_v \Delta_r} \leq e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}} \quad (49)$$

$$\mathbb{P}(F_3) \leq e^{-\frac{\bar{f}_v \Delta_r}{2}}. \quad (50)$$

From Equations (47), (48), (49) and (50), for $\Delta_0 < \Delta_r < \Delta_{\max}$, we have

$$\begin{aligned} \mathbb{P}(\{R_\ell^{\text{stop}} > \Delta_r\}) &\leq e^{-\frac{\bar{f}_v \Delta_r}{2}} + e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}} + e^{-\frac{\bar{f}_v \Delta_r}{2}} \\ &\leq 3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}}. \end{aligned} \quad (51)$$

From Equations (43), (45) and (51), we have

$$\mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) \leq \begin{cases} 1 & \Delta_r \leq \Delta_0 \\ 3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}} & \Delta_0 < \Delta_r < \Delta_{\max} \\ 3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_{\max}}{2}} & \Delta_{\max} \leq \Delta_r \end{cases} \quad (52)$$

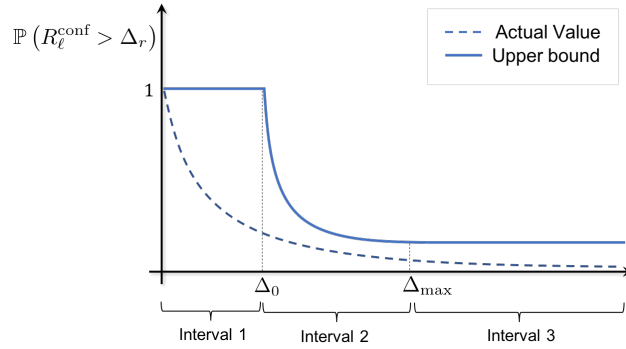


Figure 17

Using the above expression, the expectation of $\max(R_\ell^{\text{stop}}, c_2m)$ is given by

$$\begin{aligned}
\mathbb{E}[\min(R_\ell^{\text{stop}}, c_2m)] &= \sum_{\Delta_r=0}^{\Delta_{\max}} \mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) + c_2m \mathbb{P}(R_\ell^{\text{stop}} > \Delta_{\max}) \\
&\leq \Delta_0 + \sum_{\Delta_r=\Delta_0}^{\Delta_{\max}} \mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) + 3c_2me^{-(1-2\beta)\bar{f}_v\Delta_{\max}/2} \\
&\leq \Delta_0 + \sum_{\Delta_r=\Delta_0}^{\infty} \left(3e^{-\frac{(1-2\beta)\bar{f}_v\Delta_r}{2}} \right) + 3c_2me^{-4\log m} \\
&= \Delta_0 + \frac{6e^{-\frac{(1-2\beta)\bar{f}_v\Delta_0}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3c_2}{m^3} \\
&\leq \frac{12}{(1-2\beta)}r_{\min} + \frac{6}{(1-2\beta)\bar{f}_v} + \frac{3c_2}{m^3} \\
\mathbb{E}[\min(R_\ell^{\text{stop}}, c_2m)] &\leq \frac{13}{(1-2\beta)}r_{\min} + \frac{48}{\bar{f}_v(1-2\beta)^3m^3}. \tag{53}
\end{aligned}$$

□

Lemma E.9 upper bounds the expected number of rounds to confirm the proposer block list at level ℓ . However, our goal in Lemma E.6 is to confirm proposer list for *all* the levels $\ell' \leq \ell$. From Lemma E.8, we know that the proposer list at level ℓ' is confirmed by round $R_{\ell'} + \min(R_{\ell'}^{\text{stop}}, c_2m)$. Therefore, all the proposer list up to level ℓ are confirmed in the following number of rounds:

$$\begin{aligned}
R_\ell^{\text{conf}} &:= \max_{\ell' \leq \ell} (R_{\ell'} + \min(R_{\ell'}^{\text{stop}}, c_2m) - R_\ell) \\
&= \max_{\ell' \leq \ell} (\min(R_{\ell'}^{\text{stop}}, c_2m) - D_{\ell', \ell}), \tag{54}
\end{aligned}$$

where $D_{\ell', \ell} = R_\ell - R_{\ell'}$. Expression (54) is a maximum of random variables associated with each level up to level ℓ . It turns out max is dominated by random variable associated with level ℓ and in fact it's expectation, calculated in the next lemma, is very close to expectation of $\min(R_\ell^{\text{stop}}, c_2m)$. We now calculate the expectation of the random variable in expression (54).

LEMMA E.10. *All the proposer lists up to level ℓ will get confirmed in the following number of rounds in expectation:*

$$\begin{aligned}
\mathbb{E}[R_\ell^{\text{conf}}] &\leq \frac{13}{(1-2\beta)}r_{\min} + \frac{256}{(1-2\beta)^6\bar{f}_vm^2} \\
&\leq \frac{2808}{(1-2\beta)^3\bar{f}_v} \log \frac{50}{(1-2\beta)} + \frac{256}{(1-2\beta)^6\bar{f}_vm^2}.
\end{aligned}$$

PROOF. Let us define

$$\begin{aligned}
F(\{D_{\ell', \ell}\}_{\ell' \leq \ell}) &:= \mathbb{E} \left[R_\ell^{\text{conf}} | \{D_{\ell', \ell}\}_{\ell' \leq \ell} \right] \\
&= \mathbb{E} \left[\max_{\ell' \leq \ell} (\min(R_{\ell'}^{\text{stop}}, c_2m) - D_{\ell', \ell}) | \{D_{\ell', \ell}\}_{\ell' \leq \ell} \right] \\
&\leq \Delta_0 + \mathbb{E} \left[\max_{\ell' \leq \ell} (\min(R_{\ell'}^{\text{stop}} - \Delta_0, c_2m) - D_{\ell', \ell}) | \{D_{\ell', \ell}\}_{\ell' \leq \ell} \right] \\
&\leq \Delta_0 + \sum_{\ell' \leq \ell} \mathbb{E} \left[(\min(R_{\ell'}^{\text{stop}} - \Delta_0, c_2m) - D_{\ell', \ell})_+ | D_{\ell', \ell} \right]. \tag{55}
\end{aligned}$$

We bound each term in the summation the Equation (55) similar to steps used to Equations (53).

$$\mathbb{E} \left[\left(\min(R_{\ell'}^{\text{stop}} - \Delta_0, c_2 m) - D_{\ell', \ell} \right)_+ \mid \{D_{\ell', \ell}\}_{\ell' \leq \ell} \right] \quad (56)$$

$$\begin{aligned} &= \sum_{\Delta_r = D_{\ell', \ell} + \Delta_0}^{\Delta_{\max}} \mathbb{P}(R_{\ell'}^{\text{stop}} > \Delta_r \mid \{D_{\ell', \ell}\}_{\ell' \leq \ell}) + (c_2 m - D_{\ell', \ell})_+ \mathbb{P}(R_{\ell'}^{\text{stop}} > \Delta_{\max} \mid \{D_{\ell', \ell}\}_{\ell' \leq \ell}) \\ &\stackrel{(a)}{=} \sum_{\Delta_r = D_{\ell', \ell} + \Delta_0}^{\Delta_{\max}} \mathbb{P}(R_{\ell'}^{\text{stop}} > \Delta_r) + (c_2 m - D_{\ell', \ell})_+ \mathbb{P}(R_{\ell'}^{\text{stop}} > \Delta_{\max}) \\ &\leq \sum_{\Delta_r = D_{\ell', \ell}}^{\infty} \left(3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}} \right) + 3(c_2 m - D_{\ell', \ell})_+ e^{-4 \log m} \\ &\leq \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell', \ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell', \ell})_+}{m^4}. \end{aligned} \quad (57)$$

The inequality (a) follows because the random variable $R_{\ell'}^{\text{stop}}$ is independent of proposer block mining on levels other than ℓ' and depends only on the mining on voting blocktrees and proposer blocks on level ℓ' . Using Equation (57) in Equation (55) we get

$$F \left(\{D_{\ell', \ell}\}_{\ell' \leq \ell} \right) \leq \Delta_0 + \sum_{\ell' \leq \ell} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell', \ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell', \ell})_+}{m^4}. \quad (58)$$

Intuitively, if the first proposer block on every level is mined by the honest users then $D_{\ell', \ell}$ is a geometric random variable with mean $\frac{2(\ell - \ell')}{\bar{f}_v}$ i.e. linear in $\ell - \ell'$. Taking expectation on Equation (58) and substituting $D_{\ell', \ell}$ with $\frac{2(\ell - \ell')}{\bar{f}_v}$ would give us a finite bound. However this intuition is incorrect because the adversary could present proposer blocks on *multiple* levels in the same round and thus the value of $D_{\ell', \ell}$ depends on the adversarial strategy. We overcome this problem by showing that irrespective of the adversary's strategy, the honest users will propose the first proposer blocks for sufficient number of levels.

Let levels $\{L_1, L_2, \dots, L_i, \dots, L_n\}$ be the levels lesser than ℓ on which the honest users presented the first proposer block. Let $L_{n+1} = \ell$. Here L_i 's are a random variables and the first proposer block at level L_i is produced in round R_{L_i} . If the adversary produces the first proposer block at level ℓ' for $L_i < \ell' < L_{i+1}$, then from the monotonicity of the growth of the proposer blocktree, we have the following constraint $R_{L_i} \leq R_{\ell'} \leq R_{L_{i+1}}$. Let us use this in Equation (58).

$$\begin{aligned} &F \left(\{D_{\ell', \ell}\}_{\ell' \leq \ell} \right) \\ &\leq \Delta_0 + \sum_{\ell' \leq \ell} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell', \ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell', \ell})_+}{m^4}. \\ &\leq \Delta_0 + \sum_{i \in [n]} \sum_{L_i < \ell' \leq L_{i+1}} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell', \ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell', \ell})_+}{m^4} \\ &\stackrel{(a)}{\leq} \Delta_0 + \sum_{i \in [n]} (L_{i+1} - L_i) \left(\frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{L_{n+1}, L_{i+1}}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{L_{n+1}, L_{i+1}})_+}{m^4} \right). \end{aligned} \quad (59)$$

The inequality (a) follows because $R_{\ell'} \leq R_{L_{i+1}}$. Let G_j be i.i.d random variables s.t $G_j \sim \text{Geometric}(\bar{f}_v)$. Since the levels L_i and L_{i+1} are mined by the honest users, we have $D_{L_{i+1}, L_i} \geq \sum_{j=L_i}^{L_{i+1}} G_j$ and $D_{L_{n+1}, L_i} = \sum_{j=L_{i+1}}^{L_{n+1}} G_j$.

Using this in Equation (59), we get

$$F\left(\{D_{\ell', \ell}\}_{\ell' \leq \ell}\right) \leq \Delta_0 + \sum_{i \in [n]} (L_{i+1} - L_i) \left(\frac{6e^{-\frac{(1-2\beta)\bar{f}_v \sum_{j=L_{i+1}}^{L_{n+1}} G_j}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2m - \sum_{j=L_{i+1}}^{L_{n+1}} G_j)_+}{m^4} \right).$$

We now take expectation over G_j 's gives us

$$\begin{aligned} \mathbb{E}\left[F\left(\{D_{\ell', \ell}\}_{\ell' \leq \ell}\right) \mid \{L_i\}_{i=1}^n\right] &\leq \Delta_0 + \sum_{i \in [n]} (L_{i+1} - L_i) \left(\frac{6}{(1-2\beta)\bar{f}_v} \left(\frac{1}{1+(1-2\beta)} \right)^{\frac{2(L_{n+1}-L_{i+1})}{\bar{f}_v}} \right. \\ &\quad \left. + \frac{3(c_2m - \frac{L_{n+1}-L_{i+1}}{\bar{f}_v})_+}{m^4} \right). \end{aligned}$$

Since the honest user have $1 - \beta$ fraction of mining power, we have $(L_{i+1} - L_i) \sim \text{Geometric}(1 - \beta)$ and on taking expectation over L_i 's we get:

$$\begin{aligned} \mathbb{E}\left[R_\ell^{\text{conf}}\right] &= \mathbb{E}\left[F\left(\{D_{\ell', \ell}\}_{\ell' \leq \ell}\right)\right] \\ &\leq \Delta_0 + \frac{1}{1-\beta} \sum_{i \in [n]} \left(\frac{6}{(1-2\beta)\bar{f}_v} \left(\frac{1}{1+(1-2\beta)} \right)^{\frac{(n-i)}{\bar{f}_v}} + \frac{(c_2m - \frac{(n-i)}{\bar{f}_v})_+}{m^4} \right) \\ &\leq \Delta_0 + \frac{1}{1-\beta} \sum_{i=0}^{\infty} \left(\frac{6}{(1-2\beta)\bar{f}_v} \left(\frac{1}{1+(1-2\beta)} \right)^{\frac{i}{\bar{f}_v}} + \frac{(c_2m - \frac{i}{\bar{f}_v})_+}{m^4} \right) \\ &\leq \frac{2}{(1-2\beta)} r_{\min} + 2 \left(\frac{6}{(1-2\beta)^2 \bar{f}_v} + \frac{128}{(1-2\beta)^6 \bar{f}_v m^2} \right) \\ &\leq \frac{13}{(1-2\beta)} r_{\min} + \frac{256}{(1-2\beta)^6 \bar{f}_v m^2} \\ &\leq \frac{2808}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)} + \frac{256}{(1-2\beta)^6 \bar{f}_v m^2}. \end{aligned}$$

□

F FAST CONFIRMATION FOR HONEST TRANSACTIONS: PROOF OF THEOREM 4.7

This section uses ideas from the proof of Lemma D.6. Let the transaction tx enters the system¹⁵ in round r and let ℓ be the last level on the proposer blocktree which has proposer blocks at round r . Define

$$\ell^* := \max\{\tilde{\ell} \leq \ell \text{ s.t. the honest users mined the first proposer block on level } \tilde{\ell}\}$$

Let r^* be the round in which the first proposer block was mined on level ℓ^* . From the definition of ℓ^* we have the following two observations:

- (1) All the proposer blocks on levels greater or equal to ℓ^* are mined on or after round r^* because by definition there are no proposer blocks on level ℓ^* before round r^* and hence no user can mine a proposer block on a level greater than ℓ^* before round r^* .
- (2) The adversary has mined at least one proposer block on all levels in $[\ell^*, \ell]$.

Define $\Delta_0 := \frac{12r_{\min}}{1-2\beta}$. For $r_f \geq r$, let us define the following event:

$$A_{r_f} = \{Y^P[r^*, r_f - \Delta_0] - Z^P[r^*, r_f] > 0\}. \quad (60)$$

¹⁵As a part of a transaction block.

LEMMA F.1. *If event A_{r_f} occurs, then the transaction tx is included in a block b which is proposed in round $r(b) \leq r_f - \Delta_0$ and confirmed as a leader block by round r_f .*

PROOF. From our first observation, $Y^P[r^*, r_f - \Delta_0] > Z^P[r^*, r_f]$ implies that by round r_f there exists a level $\tilde{\ell} \geq \ell^*$ which has only one honest proposer block proposed in interval $[r^*, r_f - \Delta_0]$. Our second observation says that the adversary has mined a proposer block on all levels in $[\ell^*, \ell]$ and therefore, we have $\tilde{\ell} > \ell$. From Lemma E.9, the single proposer block at level $\tilde{\ell}$ is confirmed as a final leader block of its level w.p $1 - \varepsilon_m$ by round r_f . Since this proposer block was mined after round r , it will include the transaction tx . \square

Let us define the following random variable:

$$R_f := \min r_f \geq r \text{ s.t. } A_{r_f} \text{ occurs.}$$

LEMMA F.2.

$$\mathbb{E}[R_f - r] \leq \frac{24(1-\beta)r_{\min}}{(1-2\beta)^2} \leq \frac{2592}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)}. \quad (61)$$

PROOF. Consider the following random walk

$$W_{r_f} := Y^P[r^* + \Delta_0, r_f] - Z^P[r^*, r_f - \Delta_0]. \quad (62)$$

and a random variable $V \sim \text{Bin}(\Delta_0, \bar{f}_v/2)$ which is independent of W_{r_f} . It is easy to see that $Y^P[r^* + \Delta_0, r_f] - Z^P[r^*, r_f - \Delta_0] \stackrel{d}{=} W_{r_f} - V$ in distribution. Therefore, event A_{r_f} implies $W_{r_f} > V$ and we have

$$R_f = \min r_f \geq r \text{ s.t. } W_{r_f} > V \text{ occurs.}$$

The random walk W_{r_f} has a positive drift of $\frac{(1-2\beta)\bar{f}_v}{2}$. For a fixed value of V , the conditional expectation is

$$\mathbb{E}[R_f - r^* | V] = \Delta_0 + \frac{2V}{(1-2\beta)\bar{f}_v}.$$

Taking expectation on V , we get

$$\mathbb{E}[R_f - r^*] = \Delta_0 + \frac{\Delta_0}{1-2\beta} = \frac{24(1-\beta)r_{\min}}{(1-2\beta)^2}. \quad (63)$$

Since $r^* \leq r$, we have $\mathbb{E}[R_f - r] \leq \frac{24(1-\beta)r_{\min}}{(1-2\beta)^2}$. Therefore, the transaction tx is included in all the ledgers in less than $\frac{24(1-\beta)r_{\min}}{(1-2\beta)^2}$ rounds in expectation. Substituting r_{\min} from (31) give us the required result. \square

From Lemma F.1 and F.2, we conclude that a transaction, which is part of a transaction block mined in round r , is referred by a proposer block at level (say) ℓ and the leader block at this level confirmed before round $r + \frac{2592}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)}$ in expectation. This proves the main claim of Theorem 4.7.

G OTHERS

G.1 Reserve proposer blocks by the adversary

Say the honest users mine the first proposer block at level ℓ in round R_ℓ . Let $W_\ell^P(R_\ell)$ denote that the number of hidden proposal blocks on level ℓ by the adversary. In order to maximize $W_\ell^P(R_\ell)$, all these hidden proposer blocks should have a common honest parent proposer block at level (say) ℓ_{honest} linked via private proposal blocks as shown in the Figure 18. The total number of reserve blocks is given by

$$W_\ell^P(R_\ell) = \max_{\ell_{\text{honest}} \leq \ell} Z^P[R_{\ell_{\text{honest}}} + 1, R_\ell] - Y^P[R_{\ell_{\text{honest}}} + 1, R_\ell] + 1. \quad (64)$$

The random variable $Y^P[R_{\ell_{\text{honest}}}, R_\ell] - Z^P[R_{\ell_{\text{honest}}}, R_\ell]$ is a random walk in the variable ℓ_{honest} with a net drift of $\frac{(1-2\beta)\bar{f}_v}{2}$. The ratio of left drift to the right drift is 2β and from [2], we have

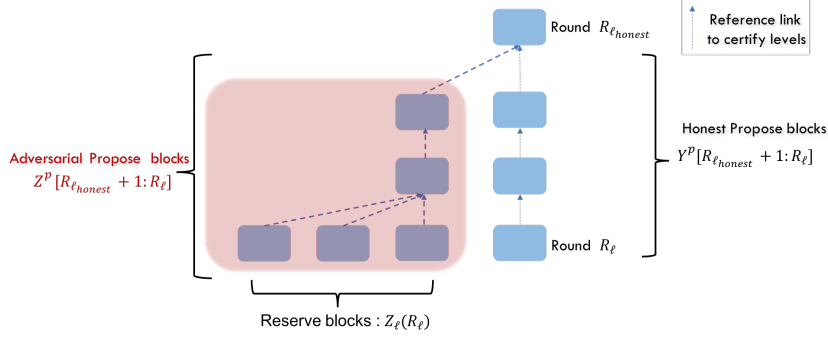


Figure 18: Number of reserved blocks by the adversary on level ℓ in round R_ℓ .

$$\begin{aligned} \mathbb{P}(W_\ell^P(R_\ell) > k) &= \mathbb{P}\left(\max_{\ell_{\text{honest}} \leq \ell} Z^P[R_{\ell_{\text{honest}}} + 1, R_\ell] - Y^P[R_{\ell_{\text{honest}}} + 1, R_\ell] \geq k\right) \\ &= (2\beta)^k. \end{aligned}$$

Therefore $W_\ell^P(R_\ell) \sim \text{Geometric}(1 - 2\beta)$.

G.2 Random walk proofs

Consider the following event from Equation (18)

$$E_1[r - r', r] := \bigcap_{a, b \geq 0} \left\{ Y[r - r' - a, r + b] - Z[r - r' - a, r + b] > \frac{(1 - 2\beta)k}{8} \right\},$$

for $r' = \frac{k}{2f_v}$. The random variable $W[r - r', r] = Y[r - r', r] - Z[r - r', r]$ is a random walk with drift $\frac{(1 - 2\beta)f}{2}$.

LEMMA G.1. *If $W[r - r', r] > c_1k$, for $c_2 < c_1$ we have*

$$\begin{aligned} \mathbb{P}(W[r - r', r + a] \geq c_2k \ \forall a > 0 \mid W[r - r', r] > c_1k) &= 1 - (2\beta)^{(c_1 - c_2)k} \\ &= 1 - e^{\log(2\beta)(c_1 - c_2)k}. \end{aligned}$$

PROOF. Refer [2]. □

If the random walk is to the right of c_1k after r' steps, the above lemma calculates the probability of that the random walk remains to the right of c_2k in all future rounds.

LEMMA G.2. *If $W[r - r', r] > c_1k$, for $c_2 < c_1$, then we have*

$$\begin{aligned} \mathbb{P}(W[r - r' - b, r] \geq c_2k \ \forall b > 0 \mid W[r - r', r] > c_1k) &= 1 - (2\beta)^{(c_1 - c_2)k} \\ &= 1 - e^{\log(2\beta)(c_1 - c_2)k}. \end{aligned}$$

PROOF. Refer [2]. □

The above lemma is mathematically characterizing the same event as Lemma G.1.

LEMMA G.3. *If $W[r - r', r] > c_1 k$, then for $c_3 < c_1$, then we have*

$$\begin{aligned} \mathbb{P}(W[r - r' - b, r + a] \geq c_3 k \mid W[r - r', r] > c_1 k) &\geq 1 - 2(2\beta)^{(c_1 - c_3)k/2} \\ &= 1 - 2e^{\log(2\beta)(c_1 - c_3)k/2} \\ &\stackrel{(a)}{\geq} 1 - 2e^{-(1-2\beta)(c_1 - c_3)k/2}. \end{aligned}$$

PROOF. Using $c_2 = (c_1 - c_3)/2$ in the above two Lemmas G.1 and G.2, we get the required result. The inequality (a) uses $\log 2\beta < 2\beta - 1$ for $\beta > 0$. \square

H THROUGHPUT OF BITCOIN

H.1 For $\beta \approx 0.5$

Indeed, in order for Bitcoin to be secured against Nakamoto's private attack [16] in that regime, it is necessary that $f\Delta$, the expected number of blocks mined per network delay round, approaches 0 so that very little forking occurs among the honest nodes and the honest nodes can grow the longest chain faster than the adversary. Note that for a given block size B , the throughput is bounded by:

$$fB = f\Delta \cdot B/\Delta = f\Delta \cdot B/(B/C + D) < f\Delta C \text{ tx/second}$$

Hence, in the regime where $\beta \rightarrow 0.5$, Bitcoin can only achieve a *vanishing* fraction of the network capacity.

H.2 For general $\beta < 0.5$

An upper bound on the worst case throughput (worst case over all adversary actions) is the rate at which the longest chain grows when no adversary nodes mine. The longest chain grows by one block in a round exactly when at least one honest block is mined. Hence the rate of growth is simply $\mathbb{P}(\# \text{blocks mined in round } r > 0)$, i.e.

$$1 - e^{-(1-\beta)f\Delta} \text{ blocks per round,} \quad (65)$$

Notice that (65) is monotonically increasing in f ; hence to maximize throughput, we should choose as high a mining rate as possible.

However, we are simultaneously constrained by security. For Bitcoin's security, [9] shows that the main chain must grow faster in expectation than any adversarial chain, which can grow at rates up to $\beta f\Delta$ in expectation. Hence we have the following (necessary) condition for security:

$$1 - e^{-(1-\beta)f\Delta} > \beta f\Delta. \quad (66)$$

Equation (66) gives the following upper bound on $f\Delta$, the mining rate per round:

$$f\Delta < \bar{f}_{\text{BTC}}(\beta),$$

where $\bar{f}_{\text{BTC}}(\beta)$ is the unique solution to the equation:

$$1 - e^{-(1-\beta)\bar{f}} = \beta\bar{f}. \quad (67)$$

This yields an upper bound on the throughput, in transactions per second, achieved by Bitcoin as:

$$\begin{aligned} \lambda_{\text{BTC}} &\leq [1 - e^{-(1-\beta)\bar{f}_{\text{BTC}}(\beta)}]B/\Delta \\ &= \beta\bar{f}_{\text{BTC}}(\beta)B/\Delta, \end{aligned} \quad (68)$$

where the last equality follows from (67). Substituting in $\Delta = B/C + D$ and optimizing for B , we get the following upper bound on the maximum efficiency of Bitcoin :

$$\bar{\lambda}_{\text{BTC}} \leq \beta\bar{f}_{\text{BTC}}(\beta),$$

achieved when $B \gg CD$ and $\Delta \gg D$.

Another upper bound on the throughput is obtained by setting f at the capacity limit: $f = C/B$ (cf. section (2)). Substituting into (65) and optimizing over B , this yields

$$\bar{\lambda}_{\text{BTC}} \leq 1 - e^{\beta-1},$$

achieved when $f\Delta = 1$, $B \gg CD$ and $\Delta \gg D$.

Combining the above two bounds, we get:

$$\bar{\lambda}_{\text{BTC}} \leq \min \left\{ \beta \bar{f}_{\text{BTC}}(\beta), 1 - e^{\beta-1} \right\}$$

This is plotted in Figure 8. Note that for large values of β , the first upper bound is tighter; this is a *security-limited* regime, in which the throughput efficiency goes to zero as $\beta \rightarrow 0.5$. This is a manifestation of the (well-known) fact that to get a high degree of security, i.e. to tolerate β close to 0.5, the mining rate of Bitcoin must be small, resulting in a low throughput. Bitcoin currently operates in this regime, with the mining rate one block per 10 minutes; assuming a network delay of 1 minute, this corresponds to a tolerable β value of 0.49 in our model.

For smaller β , the second upper bound is tighter, i.e. this is the *communication-limited* regime. The crossover point is the value of β such that

$$1 - e^{\beta-1} = \beta,$$

i.e., $\beta \approx 0.43$.

I THROUGHPUT OF GHOST

The GHOST [25] protocol uses a different fork choice rule, which uses the heaviest-weight subtree (where weight is defined as the number of blocks in the subtree), to select the main chain. To analyze the throughput of GHOST, we first observe that when there are no adversarial nodes working, the growth rate of the main chain of GHOST is upper bounded by the growth rate of the main chain under the longest chain rule. Hence, the worst-case throughput of GHOST, worst-case over all adversary actions, is bounded by that of Bitcoin, i.e.

$$1 - e^{-(1-\beta)f\Delta} \quad \text{blocks per round,} \quad (69)$$

(cf. (65)). Notice that once again, this bound is monotonically increasing in f and we would like to set f largest possible subject to security and network stability constraints. The latter constraint gives the same upper bound as (70) for Bitcoin:

$$\bar{\lambda}_{\text{GHOST}} \leq 1 - e^{\beta-1}. \quad (70)$$

?? We now consider the security constraint on f . Whereas our security condition for Bitcoin throughput was determined by a Nakamoto private attack (in which the adversary builds a longer chain than the honest party), a more severe attack for GHOST is a balancing attack, analyzed in the next subsection. As shown in that analysis, the balancing attack implies that a necessary condition on f for robustness against an adversary with power β is given by:

$$\beta f \Delta < \mathbb{E}[|H_1[r] - H_2[r]|], \quad (71)$$

where $H_1[r], H_2[r]$ are two independent Poisson random variables each with mean $(1-\beta)f\Delta/2$. Repeating the same analysis as we did for Bitcoin, we get the following upper bound on the maximum efficiency of GHOST:

$$\bar{\lambda}_{\text{GHOST}} \leq \beta \bar{f}_{\text{GHOST}}(\beta), \quad (72)$$

where $\bar{f}_{\text{GHOST}}(\beta)$ is the value of $f\Delta$ such that (71) is satisfied with equality instead of inequality.

Combining this expression with the network stability upper bound, we get:

$$\bar{\lambda}_{\text{GHOST}} \leq \min \left\{ \beta \bar{f}_{\text{GHOST}}(\beta), 1 - e^{\beta-1} \right\}. \quad (73)$$

The throughput is plotted in Figure 8. As in Bitcoin, there are two regimes, communication-limited for β small, and security-limited for β large. Interestingly, the throughput of GHOST goes to zero as β approaches 0.5, just like Bitcoin. So although GHOST was invented to improve the throughput-security tradeoff of Bitcoin, the mining rate f still needs to vanish as β gets close to 0.5. The reason is that although GHOST is indeed secure against Nakamoto private attacks for any mining rate f [25], it is not secure against balancing attacks for f above a threshold as a function of β . When β is close to 0.5, this threshold goes to zero.

I.1 Mining rate constraint

Similar to the balancing attack in [17], we would like to analyze its constraint on the mining rate f which in turns constrains the throughput. The adversary strategy is to divide the work of honest users by maintaining two forks:

- (1) Say **two** blocks b_1, b_2 are mined over the main chain block b_0 in the first round. Say the adversary mines b_1 and the honest nodes mine b_2 . The adversary will broadcast both these blocks (and all previous blocks) to all the honest users. This is when the attack starts.
- (2) At this time instance (say $r = 1$) all the honest nodes have the same view of the blocktree – which has two main chains ending at blocks b_1 and b_2 .
- (3) The honest users are divided into two equal groups G_1 and G_2 , mining over b_1 and b_2 respectively, each at average rate $(1 - \beta)f\Delta/2$ blocks per round each.
- (4) The adversary’s goal is to maintain the **forking** - make sure that G_1 chooses block b_1 in its main chain, whereas G_2 chooses block b_2 in its main chain. To do this, it divides its own resources into two equal parts A_1 and A_2 , each with average mining rate $f\Delta/2$ blocks per round. The first part A_1 mines only (direct) children of block b_1 and second part mines A_2 (direct) children of block b_2 . Suppose at round r , $H_1[r], H_2[r] \sim \text{Poiss}(1 - \beta)f\Delta/2$ honest blocks are mined in subtree 1 (below b_1) and subtree 2 (below b_2) respectively.
- (5) *Attack Strategy:*
 - If $H_1[r] = H_2[r]$, then the adversary does nothing.
 - If say $H_1[r]$ is larger, then adversary releases $H_1[r] - H_2[r]$ blocks that it has mined in subtree 2 (either in private or just mined in this round). Vice versa for the case when $H_2[r]$ is larger. This (re)balances the weight of the two subtrees and the honest work is again split in the next round.
- (6) *Analysis:* The expected number of blocks the adversary needs to release in subtree 1 per round is $\mathbb{E}[(H_2[r] - H_1[r])^+]$. A necessary condition for this attack to not be able to continue indefinitely with non-zero probability is

$$\beta f \Delta < \mathbb{E}[(H_2[r] - H_1[r])^+]/2,$$

J ADDITIONAL SIMULATIONS

Here we show our simulations from Section 5 under additional parameter settings. First, we consider an active adversary of hash power $\tilde{\beta} = 0.3$ and $\tilde{\beta} = 0.15$ with confirmation reliability $\varepsilon = e^{-20}$ in Figure 19. Notice that for $\tilde{\beta} = 0.3$, the latency of confirming double-spent transactions exceeds that of the longest-chain protocol, as explained in Section 4. The numeric latency values of the double-spent transaction curve are colored green to clarify which numbers belong to which curve.

Since $\varepsilon = e^{-20} \approx 2.1 \times 10^{-9}$ is fairly conservative (this corresponds to a latency on the order of 1 day at Bitcoin’s current settings of 1 block every 10 minutes with $\beta = 0.4$), we also consider a weaker confirmation guarantee of $\varepsilon = e^{-10} \approx 4.5 \times 10^{-5}$. The results for this weaker confirmation reliability are shown in Figure 20. As expected, all confirmation times are reduced, both for Bitcoin and for Prism. Another key difference relates to double-spent transaction latency under balancing attacks. As ε grows, Prism’s latency overtakes that of the longest-chain protocol for smaller values of β . (Recall that $\tilde{\beta}$ denotes the current fraction of hash power that is actively launching the attack, while β is the maximum tolerable fraction of adversarial hash power against which the system is secure.) This observation is expected. Prism’s latency does not grow significantly as ε changes; notice the similarity in Prism’s numeric latency values between Figures 19 and 20. However, as ε decreases, it significantly increases the latency of the longest-chain protocol.

To illustrate this effect more explicitly, Figure 21 shows the latency of Prism and the longest-chain protocol as we vary ε for a fixed $\beta = 0.4$ and $\tilde{\beta} = 0.3$. Notice that indeed, Prism’s latency changes very little as ε scales, whereas the longest-chain latency grows much faster. Additionally, we observe that censorship attacks do not appear to significantly affect latency compared to the non-adversarial setting, whereas balancing attacks can incur a much higher latency than the longest-chain protocol, with an especially pronounced difference at small values of ε . This disparity becomes less notable for smaller values of $\tilde{\beta}$, as seen in the earlier plots.

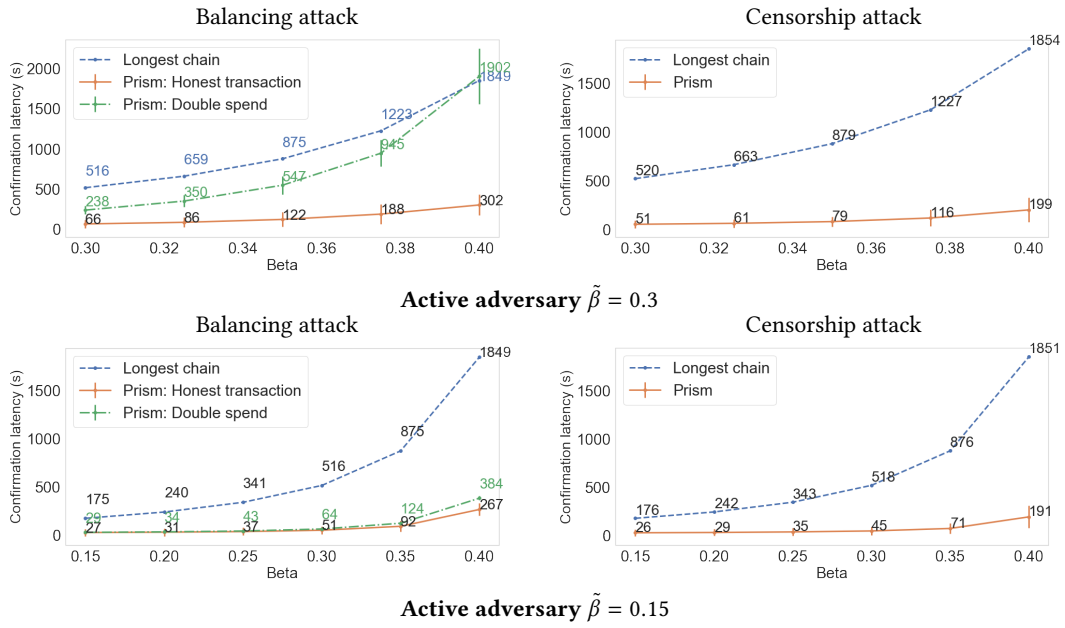


Figure 19: Active adversary at confirmation reliability $\varepsilon = e^{-20}$.

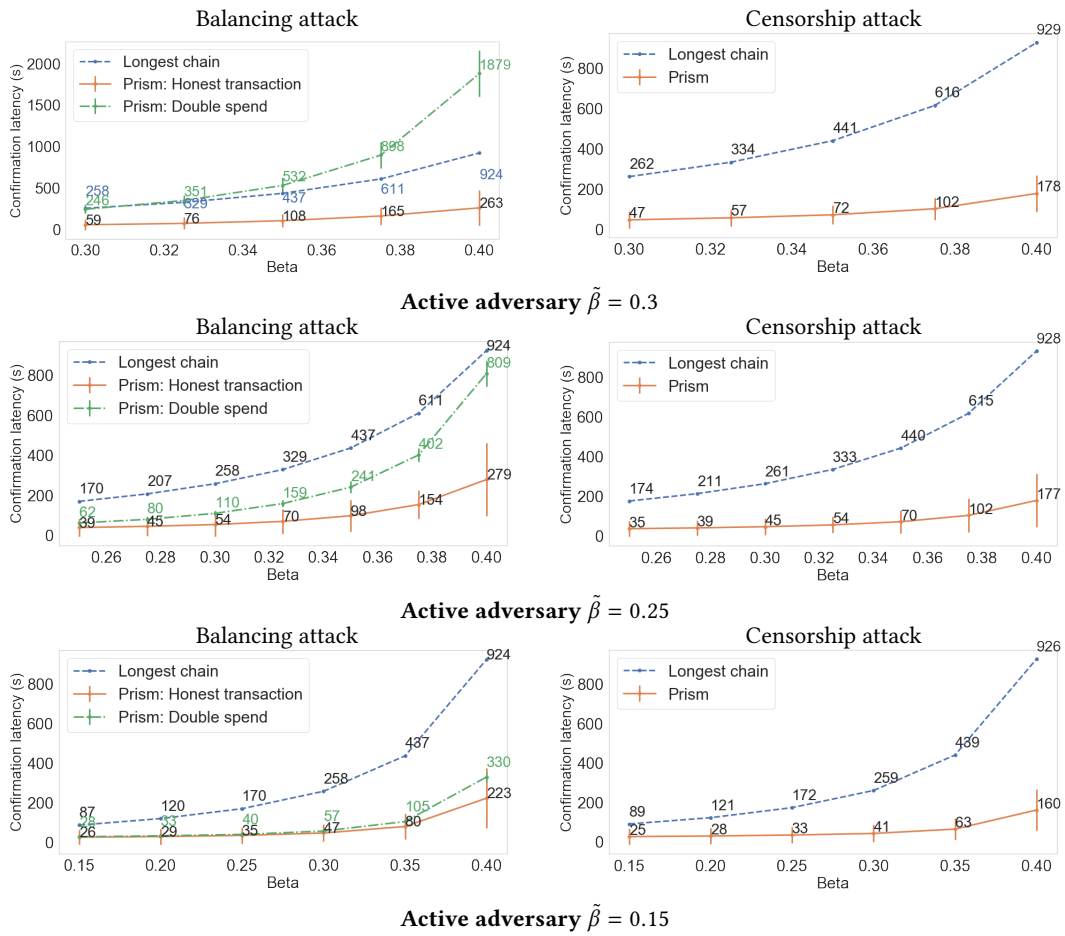


Figure 20: Active adversary at confirmation reliability $\varepsilon = e^{-10}$.

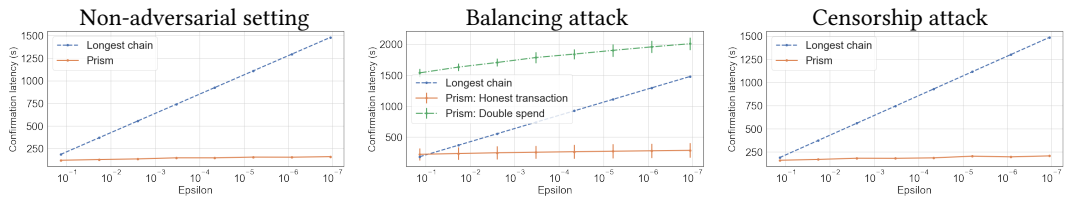


Figure 21: Latency vs. ε for a non-adversarial setting, a balancing attack, and a censorship attack when $\beta = 0.4$ and the active adversarial hash power $\tilde{\beta} = 0.3$.