Deconstructing the Blockchain to Approach Physical Limits

Vivek Bagaria[⋆], Sreeram Kannan[•], David Tse[⋆], Giulia Fanti[‡], Pramod Viswanath[† ⋆]

[⋆]Stanford University, [•]University of Washington at Seattle, [‡]Carnegie Mellon University, [†]University of Illinois at Urbana-Champaign

Abstract. The concept of a blockchain was invented by Satoshi Nakamoto to maintain a distributed ledger for an electronic payment system, Bitcoin. In addition to its security, important performance measures of a blockchain protocol are its transaction throughput, confirmation latency and confirmation reliability. Existing systems operate far away from these physical limits. 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* the blockchain into its basic functionalities and systematically scaling up these functionalities to approach their physical limits.

Keywords: Blockchain · Base Protocol · Consensus · Optimal Throughput · Optimal latency · Security · Physical limits · Incentive driven.

⋆ Email: vbagaria@stanford.edu, ksreeram@uw.edu, dntse@stanford.edu, gfanti@andrew.cmu.edu, pramodv@illinois.edu. Correspondence can be sent to dntse@stanford.edu.

Table of Contents

1 Introduction . . . 3
1.1 Performance measures . . . 3
1.2 Physical limits . . . 4
1.3 Main contribution . . . 5
1.4 Approach . . . 6
1.5 Outline of paper . . . 13
2 Related work . . . 13
2.1 High-forking protocols . . . 13
2.2 Decoupled consensus . . . 14
2.3 Hybrid blockchain-BFT consensus . . . 15
3 Model . . . 16
3.1 Mining and communication model . . . 16
3.2 Network model . . . 17
4 Approaching physical limits: throughput . . . 17
4.1 Baselines: Bitcoin and GHOST . . . 18
4.1.1 Bitcoin . . . 18
4.1.2 GHOST . . . 20
4.2 Prism 1.0: throughput-optimal protocol . . . 21
4.3 Analysis . . . 23
4.4 Transaction scheduling . . . 24
4.5 Throughput-Latency tradeoff . . . 25
4.6 Discussions . . . 26
5 Near physical limits: latency and throughput . . . 27
5.1 Bitcoin latency . . . 28
5.2 Prism . . . 29
5.2.1 Prism: backbone . . . 29
5.2.2 Prism: transaction structure . . . 32
5.2.3 Generating the ledger . . . 33
5.3 Prism: model . . . 34
5.4 Total transaction ordering at optimal throughput . . . 35
5.5 Fast confirmation of ledger list and honest transactions . . . 37
5.5.1 An example . . . 37
5.5.2 Fast list confirmation . . . 38
5.5.3 Fast confirmation of honest transactions . . . 42
6 Discussions . . . 43
6.1 Prism: incentives . . . 43
6.2 Prism: smart contracts . . . 43
6.3 Prism: Proof-of-Stake . . . 44
A An attack on GHOST . . . 47
B Bitcoin backbone properties revisited . . . 47
C Total ordering for Prism: proofs of Theorems 1 and 2 . . . 54

D Fast list confirmation for Prism: Proof of Theorem 3 60
D.1 Voter chain properties 60
D.2 Fast list confirmation policy 63
D.3 Latency 64
E Fast confirmation for honest transactions: proof of Theorem 4 73
F Others 75
F.1 Reserve proposer blocks by the adversary 75
F.2 Random walk proofs 75

1 Introduction

In 2008, Satoshi Nakamoto invented the concept of a blockchain, a mechanism to maintain a distributed ledger for an electronic payment system, Bitcoin [18]. 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. Solving the puzzle effectively involves randomly trying a hash inequality until success. Since Bitcoin's invention, much work has been done on improving Nakamoto's design; however, it remains unclear what is the best performance achievable by blockchain protocols. In this manuscript, we explore the performance limits of blockchain protocols and propose a new protocol, Prism, that performs close to those limits.

1.1 Performance measures

There are four fundamental performance measures of a PoW blockchain protocol:

1. the fraction β of hashing power the adversary can control without compromising system security;
2. the throughput λ, number of transactions confirmed per second;
3. the confirmation latency, τ, in seconds;
4. the probability ε that a confirmed transaction will be removed from the ledger in the future. ($\log 1/\varepsilon$ is sometimes called the *security parameter* in the literature[1].)

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 the order of several transactions per seconds and confirmation latency of the order of tens of minutes to hours. In fact, there is a tradeoff between the confirmation latency and the confirmation error probability: the smaller the desired the confirmation error probability, the longer the needed latency is in Bitcoin. For example, Nakamoto's calculations [18] show that for $\beta = 0.3$, while it takes a latency of 10 blocks (on the average, 100 minutes) to achieve an error probability of 0.04, it takes a latency of 30 blocks (on the average, 300 minutes) to achieve an error probability

[1] All logarithms in this paper are taken with respect to base e.

of 10^{-4}. This latency arises because in order to provide a low error probability, blocks must be deep in the underlying blockchain to prevent the adversary from growing a longer side chain and overwriting the block in question.

1.2 Physical limits

Bitcoin has strong security guarantees, being robust against an adversary with up to 50% hashing power. However, its throughput and latency performance are poor; in particular high latency is required to achieve very reliable confirmation. Much effort has been expended to improve the performance in these metrics while retaining the security guarantee of Bitcoin. 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 in this manuscript, since a transaction is the basic unit of information in a payment system. 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. Assuming that each transaction needs to be communicated at least once across the network, it is clear that λ, the number of transactions which can be confirmed per second, is at most C, i.e.

$$\lambda < C. \tag{1}$$

One obvious constraint on the confirmation latency τ is that

$$\tau > D. \tag{2}$$

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 transactions, then at most

$$\frac{C}{B} \cdot \tau$$

mined blocks can be communicated across 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 and $C/B\tau$ votes are accumulated over duration τ. This number is maximized at $C\tau$, when the block size is smallest possible, i.e. $B = 1$. 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 $C\tau$, 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.

$$\varepsilon = \exp\{-O(C\tau)\}. \tag{3}$$

Turning this equation around, we have the following lower bound on the latency for a given reliability requirement ε:

$$\tau = \Omega\left(\frac{1}{C} \cdot \log\frac{1}{\varepsilon}\right). \tag{4}$$

Comparing the two constraints (2) and (4), we see that if

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

the latency is limited by the propagation delay; otherwise, it is limited by the confirmation reliability requirement. The quantity CD is analogous to the key notion of *bandwidth-delay product* in networking (see eg. [11]); it is the number of "in-flight" transactions 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 round-the-world speed-of-light propagation delay D of 0.2 seconds. If we take a transaction of size 100 bytes, then $C = 25,000$ transactions per second. The bandwidth-delay product $CD = 5000$ is very large. Hence, the confirmation latency is limited by the propagation delay of 0.2 seconds, but not by the confirmation reliability requirement unless it is astronomically small. Real-world blockchain systems 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 $\beta = 0.3$ [5].

1.3 Main contribution

The main contribution of this work is a new blockchain protocol, Prism, which has the following provable performance guarantees:

1. **security:** Prism is secure up to an adversary power of 50%, i.e. for any $\beta < 0.5$ and for arbitrary adversarial action[2], it can achieve an eventual total ordering of the transactions, with consistency and liveness guarantees.
2. **throughput:** For arbitrary adversarial action, Prism can achieve a throughput
$$\lambda = (1-\beta)C \quad \text{transactions per second.}$$

[2] The precise class of allowable adversarial actions will be defined in the formal model.

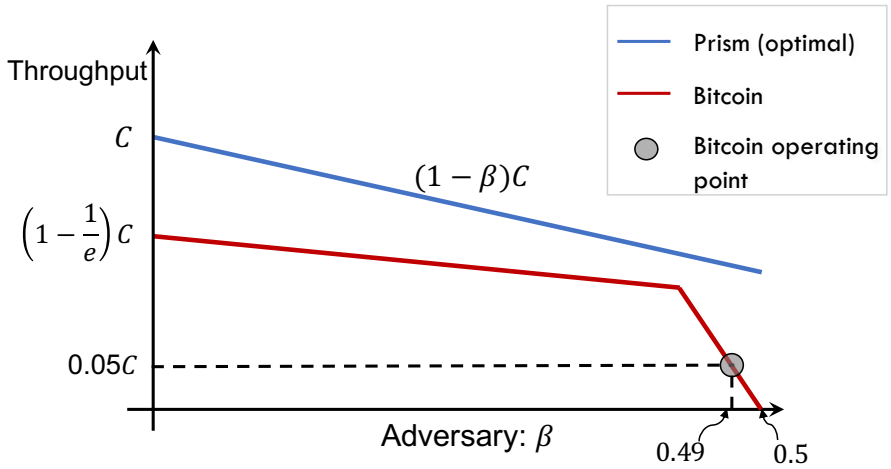
3. **latency:** For any $\beta < 0.5$ and for arbitrary adversarial action, Prism can confirm honest transactions (without public double spends) with an expected latency
$$\mathbb{E}[\tau] < \max\left\{a_1(\beta)D, \frac{a_2(\beta)}{C}\log\frac{1}{\varepsilon}\right\} \quad \text{seconds},$$
with confirmation reliability at least $1-\varepsilon$. Here, $a_1(\beta)$ and $a_2(\beta)$ are constants depending only on β (defined in (28) and (29)).

The results are summarized in Figure 1. Some comments:

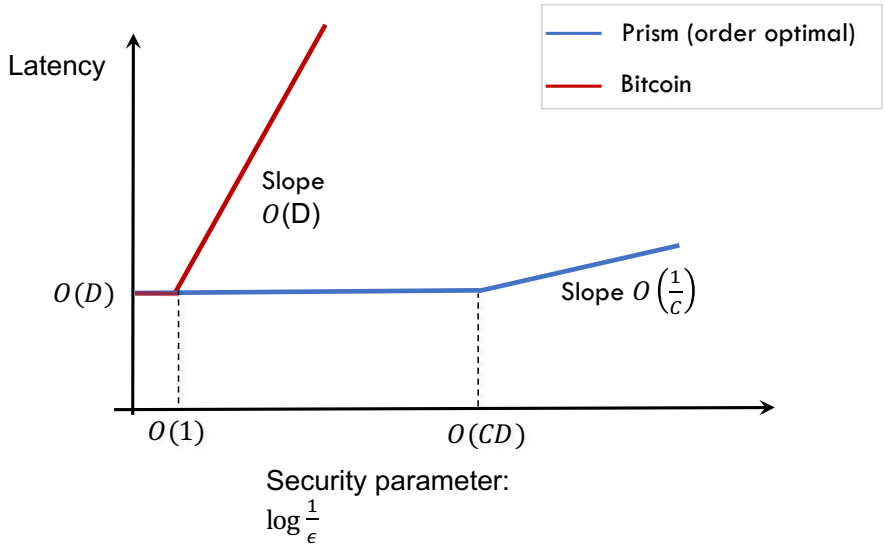
- The security of Prism is as good as Bitcoin: Prism can be robust to an adversary with hashing power up to $\beta = 0.5$.
- Since $1-\beta$ is the fraction of honest hashing power, Prism 's throughput is optimal assuming each transaction needs to be communicated across the network.
- Prism achieves a confirmation latency for honest transactions matching, in order, to the two physical limits (2) and (4). In particular, if the desired security parameter $\log\frac{1}{\varepsilon} \ll CD$, the confirmation latency is of the order of the propagation delay and *independent* of $\log 1/\varepsilon$. Put it another way, one can achieve latency close to propagation delay with a confirmation error probability exponentially small in the bandwidth-delay product CD.
- For a total ordering of all transactions (including double spends), on the other hand, the trade off between latency and the security parameter is similar to that of Bitcoin.

1.4 Approach

A critical parameter of any PoW blockchain protocol is the mining rate, i.e. the rate at which puzzles are successfully solved (also called the PoW solution rate). The mining rate can be easily controlled via adjusting the difficulty of the puzzle, i.e. the threshold at which the hash inequality needs to be satisfied. The mining rate has a profound impact on both the transaction throughput and confirmation latency. Large mining rate can potentially increase the transaction throughput by allowing transactions to be processed quicker, and can potentially reduce the confirmation latency by increasing the rate at which votes are casted to confirm a particular transaction. However, increasing the mining rate has the effect of increasing the amount of forking in the blocktree, because blocks mined by different nodes within the network delay cannot be mined on top of each other and are hence forked. This de-synchronization slows down the growth rate of the longest chain, making the system more vulnerable to private chain attacks, and decreasing the security of the protocol. Indeed, one reason why Bitcoin is highly secure is that the mining rate is set to be very small, one block per 10 minutes. At the current Bitcoin block size of 1 Mbytes, this corresponds to a generated traffic of about 13 kbits/second, much less than capacity of typical communication links [30]. Thus, Bitcoin 's performance is security-limited, not communication-limited, and far away from the physical limits.



(a) Throughput



(b) Latency

Fig. 1: Main results. (a) Throughput vs adversarial fraction β for Prism and Bitcoin. The red curve is an optimized upper bound of Bitcoin's throughput, derived in Section 4.1. Note that the throughput of Prism is a positive fraction of the network capacity all the way up to $\beta = 0.5$, but the throughput of Bitcoin vanishes as a fraction of the capacity as $\beta \to 0.5$. (b) Confirmation latency vs. security parameter for Prism and Bitcoin. The red curve is a lower bound on Bitcoin 's latency, derived in Section 5.1. The latency of Prism is independent of the security parameter value up to order CD and increases very slowly after that (with slope $1/C$). For Bitcoin, latency increases much more rapidly with the security parameter.

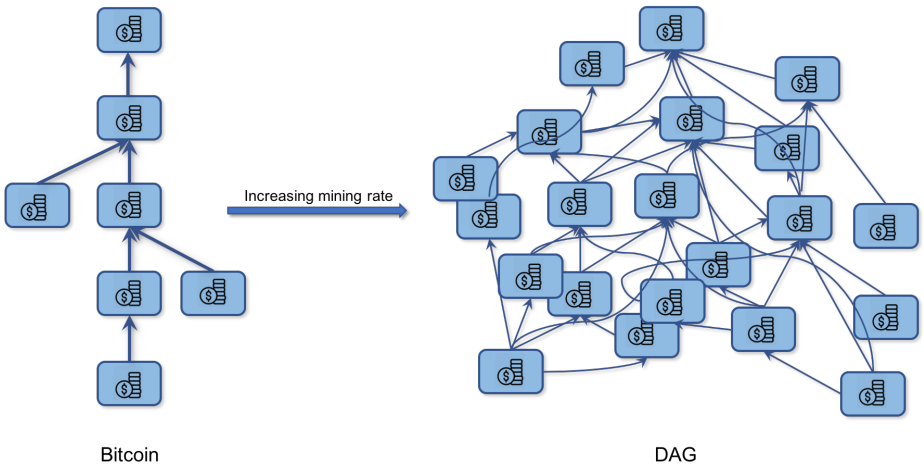


Fig. 2: The DAG approach to increasing the mining rate.

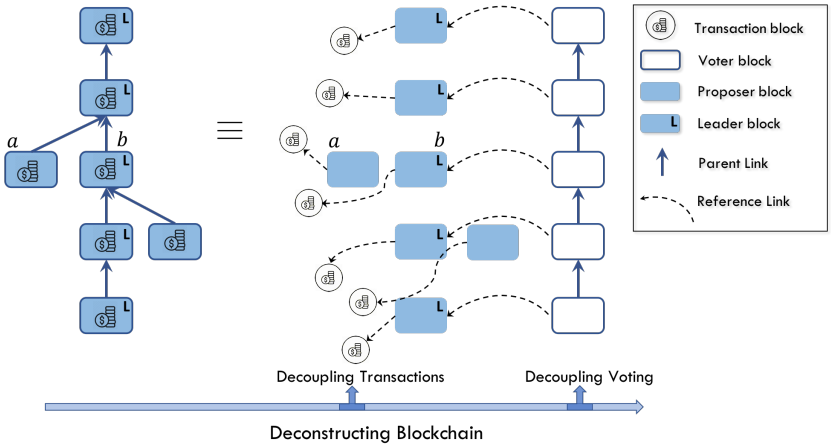


Fig. 3: Deconstructing the blockchain into transaction blocks, partially ordered proposal blocks arranged by level, and voter blocks organized in a voter tree. The main chain is selected through voter blocks, which vote among the proposal blocks at each level to select a leader block. For example, at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

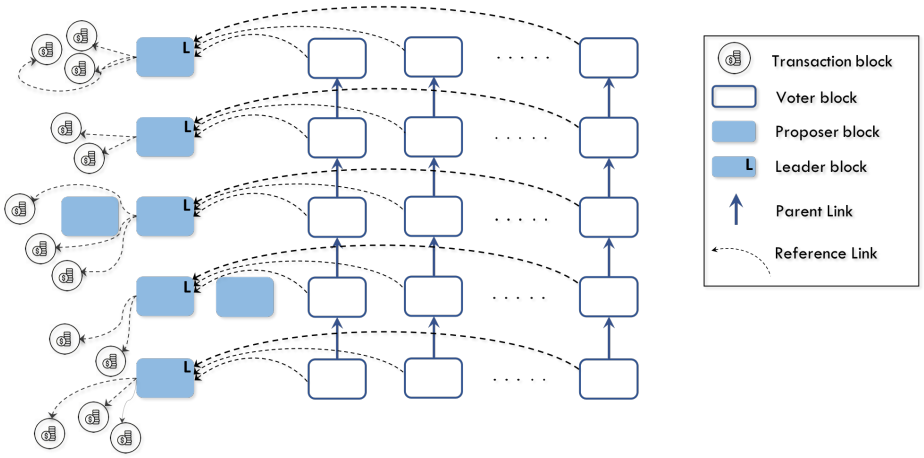


Fig. 4: Prism. Throughput, latency and reliability are scaled to the physical limits by increasing the number of transaction blocks and the number of parallel voting chains per proposal block.

To increase the mining rate while maintaining security, one line of work in the literature has used more complex fork choice rules and/or added reference links to convert the blocktree into more complex structures such as a directed acyclic graph (DAG). This allows a block to be confirmed by other blocks that are not necessarily its descendents on a main chain. (Figure 2). Examples of such works are GHOST [29], Inclusive [15], Spectre [28], Phantom[27] and Conflux [16]. However, as discussed in more details in the related work section, GHOST, Phantom, and Conflux all have security issues, and Spectre does not provide total ordering of transactions. It is fair to say that handling a highly forked blocktree is challenging.

In this work, we take a different approach. We start by *deconstructing* the basic blockchain structure into its atomic functionalities, illustrated in Figure 3. 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 elect 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. 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 blocktree 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 voter blocks are organized in their own blocktree and support each other through parent links. Thus, the parent links in the original blocktree have two implicit functionalities which are explicitly separated in this representation: 1) they provide a partial ordering of the proposal blocks according to their levels, and 2) they help the voting blocks to vote for each other.

This alternative representation of the traditional blockchain, although seemingly more complex than the original blockchain representation, provides a natural path for scaling the performance of blockchain protocols to approach physical limits (Figure 4). 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, with many voters voting on the proposal blocks in parallel, 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 three kinds of blocks and the links between them, is a DAG, but a structured DAG.

This complexity management presupposes a way to provide *sortition* in the mining process: when miners mine for blocks, they should not know in advance whether the block will become a proposal block, a transaction block, or a voting block, and if it is a voting block, it should not know in advance what particular chain the voting block will be in. Otherwise an adversary can focus its hashing power to attack a particular part of the structure. This sortition can be accomplished by using the random hash value when a block is successfully mined; this is similar to the 2-for-1 PoW technique used in [10], which is also used in Fruitchains [22] 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. This line of work will be discussed in depth in Section 2, but its focus is only on decoupling the transactions-carrying functionality. In our work, we broaden this principle to decouple *all* functionalities.

In Bitcoin, the irreversibility of a block in the longest chain is achieved by a *law of large numbers* effect: the chance that an adversary with less that 50% hashing power can grow a private chain without the block and longer than the public chain diminishes with the depth of the block in the public chain. This is the essence of the random walk analysis in Nakamoto's original paper [18] and is also implicit in the formal security analysis of Bitcoin in [10] (through the definition of *typical execution*). The law of large numbers allows the averaging of the randomness of the mining process, so that the chance of the adversary getting lucky and mining many blocks in quick succession is small. This averaging

is achieved over time, and comes at the expense of long latency, which increases with the desired level of reliability.

Prism also exploits the law of large numbers, but over the number of parallel voter trees instead of over time. Due to the sortition mechanism, the mining processes of both the adversary and the honest nodes are independent across the voting trees. By having many such trees, many votes are cast on the proposer blocks at a given level, and the chance of an adversary with less than 50% hashing power being able to reverse many of these votes decreases exponentially with m, the number of voter trees. The number of voter trees is m, and hence the rate of vote generation, is limited only by the physical capacity C of the network. Thus, we can attain irreversibility of a large fraction of the votes with high probability (approaching 1 exponentially fast in the bandwidth-delay product CD) without waiting for a long time. We show that this irreversibility of votes allows fast confirmation of a final leader block among a list of proposer blocks at a given level. In particular, it is guaranteed that the adversary cannot propose another block in the future that has enough votes to become the final leader block at that level. The ability to do this for all levels up to a given level generates a list of transaction ledgers, one of which must be a prefix of the eventual totally-ordered ledger (Figure 5). Together with liveness of honest transactions, we show that this "list decoding" capability is sufficient for fast confirmation of all honest transactions[3]. If a given block obtains a substantial enough majority of votes, then the list can be narrowed to contain only that block, which can then be declared the leader block. In the worst case, when votes are tied between two or more proposer blocks at the same level (due to active intervention by the adversary, for example), the irreversibility of *all* of the votes and a content-dependent tie-breaking rule is needed to come to global consensus on a unique leader block; this requires higher latency. Hence, Prism requires high latency in the worst case to guarantee total ordering of all transactions.

The above discussion gives some intuition behind Prism, but a formal analysis is needed to rigorously establish security, latency and throughput performance guarantees. Such a formal analysis was done on Bitcoin in [10] in a synchronous round-by-round model and subsequently extended in [21] to an asynchronous model with an upper bound on block network delay. In particular, [10] pioneered the backbone protocol analysis framework where it was shown that two key properties, the common-prefix property and the chain-quality property, of the Bitcoin backbone guarantee persistence and liveness of the ledger maintained by Bitcoin respectively. We leverage this framework to provide a formal analysis of Prism in the synchronous round-by-round model (we conjecture that similar results can be established in the more sophisticated asynchronous model of [21]). Technically, the most challenging part of the analysis is on fast latency confirmation, where we show that: 1) the common-prefix property of the vote trees guarantee *vote persistence*, so that a large fraction of the votes will not be

[3] List decoding is a concept in coding theory. Instead of decoding to a unique codeword, list decoding generates a list of possible codewords which is guaranteed to contain the true codeword.

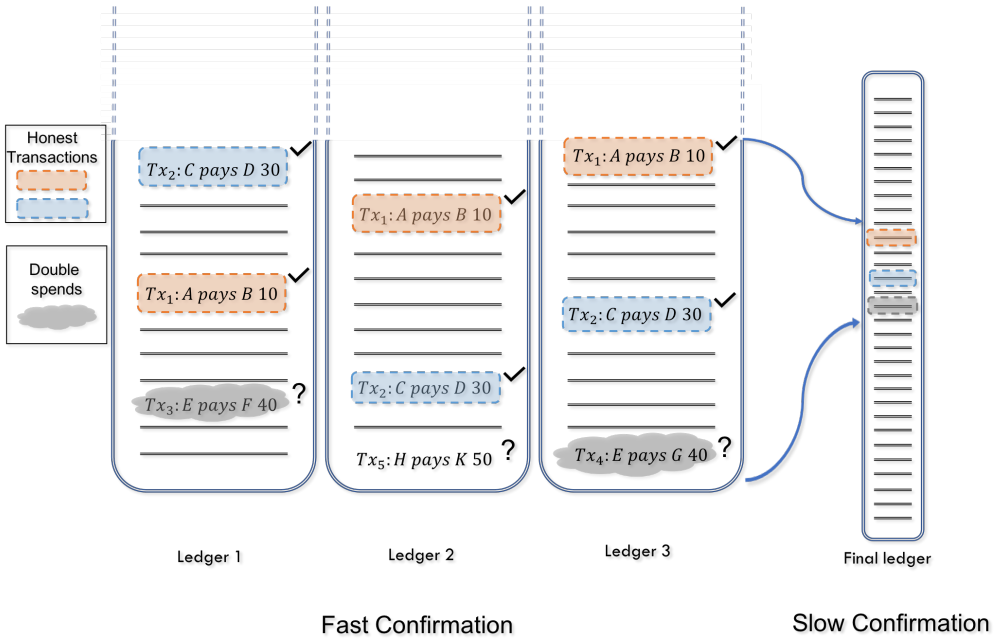


Fig. 5: List ledger confirmation. An example where one can fast-confirm that the final ledger is one of three possibilities. Honest transactions that appear in all three ledgers can be fast-confirmed. Double spends cannot appear in all ledgers and are therefore not fast-confirmed, although one of them will be slow-confirmed.

reversed; 2) the chain-quality of the main chains of the vote trees guarantee vote liveness, so that a large fraction of the vote trees will contain honest votes on the proposer blocks at each level of the proposal tree.

1.5 Outline of paper

In Section 2, we discuss other lines of work in relation to our approach. In Section 3, we review the synchronous model used in [10] and introduce our network model that ties the blockchain parameters to physical parameters of the underlying network. In Section 4, we focus on throughput, and discuss a simplified version of the protocol, Prism 1.0, which achieves full security and optimal throughput. Since Prism 1.0 lacks voter blocktrees, it has latency equivalent to Bitcoin. In Section 5, we add vote trees to the protocol, and perform a formal analysis of its security and fast latency. The result is a protocol, Prism, which can achieve full security, optimal throughput and near physical limit latency on ledger list decoding and confirmation of honest transactions. In Section 6, we will discuss the issue of incentivization, as well as applications of our results to Proof-of-Stake and smart contracts systems.

2 Related work

In this section, we discuss and compare our approach to several lines of work.

2.1 High-forking protocols

As discussed in the introduction, one approach for increasing throughput and decreasing latency is the use of more sophisticated fork choice and voting rules to deal with the high-forking nature of the blocktree. Examples of such high-forking protocols include GHOST [29], Inclusive [15], Spectre [28], Phantom [27], and Conflux [16]. The earliest of these schemes, GHOST, handles forking through a fork-choice rule that builds on the heaviest subtree [29]. The authors observed that in order to improve throughput, we must increase the block mining rate, f. However, as f grows, so too does the number of blocks mined in parallel, which are wasted under Bitcoin's longest-chain fork choice rule, thereby reducing security. GHOST's heaviest-subtree rule allows the system to benefit from blocks that were mined in parallel by honest nodes since such blocks are counted in the main chain calculation. While it was shown in [29] that GHOST is secured against a 50% purely private attack, it turns out that GHOST is vulnerable to a public-private balancing attack [19], where the adversary can use minimal effort to split the work of the honest nodes across two subtrees of equal weight, and then launch a private attack. It turns out that counting side-chain blocks in selecting the main chain allows the adversary to withhold earlier mined blocks and use them at later times to balance the growth of the two subtrees. We present an analysis of this attack in the Appendix and show that this attack restricts

the mining rate f of GHOST to be similar to that of Bitcoin, thus minimizing the advantage of GHOST.

To improve security at high mining rates, another popular idea is to add reference links between blocks in addition to traditional parent links, resulting in a DAG-structured blockchain. Each block in a DAG can reference multiple previous blocks instead of a unique ancestor (as in Bitcoin). The pertinent challenges are how to choose the reference links and how to obtain a total ordering of blocks from the observed DAG in a way that is secure. In a family of protocols, Inclusive, Spectre and Phantom, every block references all previous orphan blocks. These reference links are interpreted in differing ways to give these different protocols. For example, in [15], the key observation is that the reference link structure provides enough information to emulate any main-chain protocol, such as the longest-chain or GHOST protocol, while in addition providing the ability to pull in stale blocks into a valid ledger. However, the security guarantee remains the same as that of Bitcoin (namely, tending to zero as the mining rate grows), and it does not achieve optimal throughput.

Spectre is an innovative scheme that builds upon the the DAG idea to achieve low confirmation time by interpreting the reference links as votes to compare between pairs of blocks [28]. However, the fast confirmation is restricted to honest transactions and the system does not guarantee liveness for double-spends as well as not having the ability to confirm smart contracts that need a totally-ordered ledger. Since complete ordering is important for core blockchain applications (e.g., cryptocurrencies), a later work, Phantom, builds on Spectre to achieve consensus on a total ordering of transactions by having participants topologically sort their local DAGs [27]. The authors suggest that by combining Spectre and Phantom, one may be able to achieve low confirmation latency for honest transactions as well as eventual total ordering. However, a recent work [16] demonstrates a liveness attack on Phantom. Furthermore, the proposed hybrid scheme cannot confirm non-contentious smart contracts with fast latency. Although Prism uses a DAG to order transactions, it diverges from prior DAG schemes by separating block proposal from block ordering in the protocol. This helps because an adversarial party that misbehaves during block proposal does not affect the security of transaction ordering, and vice versa; it provides a degree of algorithmic sandboxing.

Conflux is another DAG-based protocol whose goal is to increase throughput [16]. However, Conflux's reference links are not used to determine where to mine blocks or how to confirm them; they are only used to include side-chain blocks into the main chain to improve throughput. The main chain itself is selected by the GHOST rule. Due to the vulnerability of GHOST to the balancing attack, the secured throughput of Conflux is limited to Bitcoin levels. (See discussions in Section 4.6.)

2.2 Decoupled consensus

Our design approach is based on the principle of *decoupling* the various functionalities of the blockchain. This decoupling principle has already been applied

in various earlier works, but mainly in decoupling the transactions. We review these works here.

BitcoinNG [8] elects a single leader to propose a predetermined number of transaction blocks, called an epoch. At the end of this epoch, a new leader is elected. Thus, there is a decoupling of proposal blocks and transaction blocks, the goal being to increase the throughput. However, since the transaction blocks are not mined but are put on the chain by the leader after the leader is elected, this protocol is subject to potential bribery and DDoS attacks on the leaders, whereby an adversary can corrupt a block proposer after learning its identity. In contrast, Prism does not reveal the identity of a block proposer *a priori*.

The objective of Fruitchains [22] is to provide better chain quality compared to Bitcoin; at a high level, chain quality refers to the fraction of blocks in the main chain belonging to the adversary. In Bitcoin, adversaries can augment this fraction relative to their computational power by using strategic mining and block release policies, such as selfish mining [9,26,20]. Fruitchains mechanically resembles Nakamoto consensus, except miners now mine separate mini-blocks, called fruits, for each transaction. Fairness is achieved because the fraction of fruits a miner can mine is proportional to its computational power. As in BitcoinNG , the fruits (transactions) are decoupled from the proposal blocks in the blocktree, but for a different reason: to improve fairness.

2.3 Hybrid blockchain-BFT consensus

Another line of work to improve throughput and latency combines ideas from Byzantine fault tolerant (BFT) along with blockchains. *Hybrid consensus* uses a combination of traditional mining under a longest-chain fork choice rule with Byzantine fault tolerant (BFT) consensus [23]. The basic idea is that every k blocks, a BFT protocol is run by an elected committee of nodes. Hybrid consensus is designed to provide *responsiveness*, which describes systems whose performance depends on the actual network performance rather than an *a priori* bound on network delays. The authors show that no responsive protocol can be secure against more than 1/3 adversarial power, and hybrid consensus achieves this bound. In this work, our focus is not on being responsive to network delay, but close to the propagation delay physical limit and small error probability.

A closely-related protocol called Thunderella includes a slow Nakamoto consensus blockchain, as well as a faster confirmation chain that is coordinated by a leader and verified by a BFT voting protocol [24]. Thunderella achieves low latency under optimistic conditions, including having a honest leader and a $\beta < 0.25$, while having consistency under worst case condition ($\beta = 0.5$). In contrast, our protocol achieves low latency under all conditions, but for list-decoding and confirmation of honest transactions.

3 Model

3.1 Mining and communication model

Let $\mathcal{N}$ denote the set of participating nodes in the network. Each transaction is a cryptographically secure payment message. When a transaction arrives at the network, it is assumed to be instantaneously broadcast to all nodes in the network. A *block* consists of an ordered list of B transactions and a few reference links to other blocks. Each node $n \in \mathcal{N}$ controls p_n fraction of total hashing power and it create blocks from the transactions and mines them with Poisson process rate fp_n blocks per second. There are two types of nodes – honest nodes, $\mathcal{H} \subset \mathcal{N}$, who strictly follow the protocol, and the adversarial nodes, $\mathcal{N}/\mathcal{H}$, who are allowed to not follow the protocol. The adversarial nodes control β fraction of hashing power i.e, $\sum_{v \in \mathcal{N}/\mathcal{H}} p_v = \beta$, whereas the honest nodes control the other $1-\beta$ fraction of hashing power[4]. As a consequence, the honest nodes mine blocks with Poisson process rate $\sum_{v \in \mathcal{H}} fp_v = (1-\beta)f$ and the adversarial nodes mine blocks with Poisson process rate $\sum_{v \in \mathcal{N}/\mathcal{H}} fp_v = \beta f$. Without loss of generality we can assume a single adversarial node with β fraction of hashing power.

The nodes exchange blocks via a broadcast channel. The time taken transmitting a block from one honest node to another honest node is assumed to be Δ seconds. On the other hand, the adversary can transmit and receive blocks with arbitrary delay, up to delay Δ.

To simplify our analysis, we discretize the above continuous-time model into the discrete-time round-by-round synchronous model proposed in [10]. Each round in this model corresponds to Δ seconds in the continuous-time model above. In the rth round, let $H[r]$ and $Z[r]$ be the number of blocks mined by the honest nodes and by the adversarial nodes respectively. The random variables $H[r]$ and $Z[r]$ are Poisson distributed with means $(1-\beta)f\Delta$ and $\beta f\Delta$ respectively and are independent of each other and independent across rounds. The $H[r]$ blocks are broadcast to all the nodes during the round, while the adversary may choose to keep some or all of the $Z[r]$ blocks in private. The adversary may also choose to broadcast any private block it mined from previous rounds. The adversary is allowed to first observe $H[r]$ and then take its action for that round. At the end of each round, all nodes in the network have a common view of the public blocktree.

An important random variable is $Y[r]$, which equals 1 when $H[r] = 1$ and 0 otherwise. This is the indicator random variable for whether the rth round is a *uniquely successful* round, i.e. a round in which only one block is mined by the honest nodes [10]. Note that $Y[r]$ has a Bernoulli distribution with parameter $(1-\beta)f\Delta e^{-(1-\beta)f\Delta}$. Another important random variable is $X[r]$, which equals 1 when $H[r] \geq 1$. We denote $H[r_1, r_2] := \sum_{r=r_1+1}^{r_2} H[r]$, similarly for X, Y and Z.

The location of the $H[r]$ honest blocks in the block data structure after the rth round is protocol-dependent. In Bitcoin, for example, all honest blocks are

[4] β for **b**ad.

appended to the longest chain of the public blocktree from the previous round. Adversarial blocks can instead be mined on any public or private block from the previous round.

Following the Bitcoin backbone protocol model [10], we consider protocols that execute for a finite number of rounds, $r_{\max}$ which we call the execution horizon. We note that we do not consider cryptographic failure events, such as insertion in the blockchain, since it has been demonstrated already in the backbone protocol paper that for a polynomial number of rounds $r_{\max}$ in the hash-length, these events have negligible probability.

3.2 Network model

To connect to the physical parameters of the network, we assume a very simple network model. The network delay Δ is given by:

$$\Delta = \frac{B}{C} + D, \tag{5}$$

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 [29], based on empirical data in [7], as well in [25]. However, here, we put an additional qualification: this expression is valid only assuming the network is stable, i.e. the total workload of communicating the blocks is less than the network capacity. In terms of our parameters:

$$fB < C. \tag{6}$$

For a given block size, (6) imposes a physical constraint on the total mining rate f. This *stability constraint* sets our model apart 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.

Note that the protocols discussed in this manuscript can be used in any network setting. This simple network model is only used as a common baseline to evaluate how well a particular protocol performs relative to the physical limits. In particular, the delay model (5) ignores queuing delay due to randomness of the times of the block transmission across the network.

4 Approaching physical limits: throughput

In this section, we study the optimal throughput λ achievable under worst-case adversarial behavior for a given adversarial power β. The main results are summarized in Figure 6, which show plots of $\bar{\lambda} := \lambda/C$ versus β for various protocols. The metric $\bar{\lambda}$ is the throughput as a fraction of the network capacity and is a measure of the efficiency of a protocol. The plot shows upper bounds on the efficiency of two baseline blockchain protocols, Bitcoin and GHOST (a version of GHOST is used in Ethereum). Note that the throughput efficiency of both protocols vanishes as β approaches 0.5. In contrast, we design a protocol,

Prism 1.0, which attains a throughput efficiency of $1 - \beta$. This efficiency does not vanish as β approaches 0.5 and is in fact the best possible if only honest nodes are working. We will see that the difference between Prism 1.0 and the two baseline protocols is that while the throughput of the two baseline protocols are *security-limited* for large β, the throughput of Prism 1.0 is only limited by the physical network capacity for *all* $\beta < 0.5$.

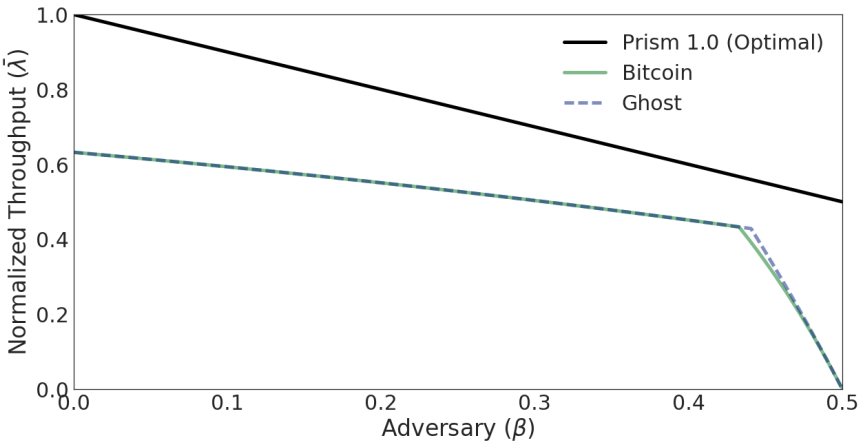


Fig. 6: Throughput efficiency versus β tradeoff of baseline protocols and Prism 1.0 . The tradeoffs for the baseline protocols are upper bounds, while that for Prism 1.0 is exact.

4.1 Baselines: Bitcoin and GHOST

We derive upper bounds on the achievable throughput under worst-case adversarial behavior of two key baselines: Bitcoin and GHOST. Throughput can be altered by tuning two parameters: the mining rate f and block size B. We are interested in the maximum achievable throughput efficiency ($\bar{\lambda} := \frac{\lambda}{C}$), optimized over B and f. To simplify notation, we suppress the dependence of $\bar{\lambda}$ on β.

4.1.1 Bitcoin

The security and consensus properties of Bitcoin have been studied by Nakamoto [18], and formally by [10] in the synchronous model, followed by the analysis of [21] in the asynchronous model. These works and others (e.g., [29,13]) show that choice of f and B in Nakamoto consensus has tradeoffs. As the mining rate f grows, forking increases and the maximum tolerable adversarial fraction β shrinks, Similarly, as the block size B grows, the network delay Δ also grows, which causes forking.

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 $\mathbb{P}(H(r) > 0)$, i.e.

$$1 - e^{-(1-\beta)f\Delta} \text{ blocks per round,} \tag{7}$$

Notice that (7) 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, [10] 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:

$$1 - e^{-(1-\beta)f\Delta} > \beta f \Delta. \tag{8}$$

Equation (8) 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:

$$1 - e^{-(1-\beta)\bar{f}} = \beta \bar{f}. \tag{9}$$

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

$$\begin{aligned} \lambda_{\text{BTC}} &\leq [1 - e^{-(1-\beta)\bar{f}_{\text{BTC}}(\beta)}]B/\Delta \\ &= \beta \bar{f}_{\text{BTC}}(\beta)B/\Delta, \end{aligned} \tag{10}$$

where the last equality follows from (9). 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. (6)). Substituting into (7) 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 6. 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 $\beta \to 0.5$. 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., $\beta \approx 0.43$.

4.1.2 GHOST

The GHOST [29] 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.

$$1 - e^{-(1-\beta)f\Delta} \quad \text{blocks per round,} \tag{11}$$

(cf. (7)). 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 (12) for Bitcoin:

$$\bar{\lambda}_{\mathsf{GHOST}} \leq 1 - e^{\beta - 1}. \tag{12}$$

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 Appendix A. 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:

$$\mathbb{E}[|H_1[r] - H_2[r]|] > \beta f \Delta, \tag{13}$$

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:

$$\bar{\lambda}_{\mathsf{GHOST}} \leq \beta \bar{f}_{\mathsf{GHOST}}(\beta), \tag{14}$$

where $\bar{f}_{\mathsf{GHOST}}(\beta)$ is the value of $f\Delta$ such that (13) is satisfied with equality instead of inequality.

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

$$\bar{\lambda}_{\mathsf{GHOST}} \leq \min\left\{\beta \bar{f}_{\mathsf{GHOST}}(\beta), 1 - e^{\beta - 1}\right\}. \tag{15}$$

The throughput is plotted in Figure 6. 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 [29], 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.

4.2 Prism 1.0: throughput-optimal protocol

We propose a protocol, Prism 1.0, that achieves optimal throughput efficiency $\bar{\lambda} = 1 - \beta$, which does not vanish as β approaches 0.5. We will build on Prism 1.0 in Section 5 to obtain our full protocol Prism.

Forking is the root cause of Bitcoin's and GHOST's inability to achieve optimal throughput. In designing Prism 1.0, our key insight is that we can create a secure blockchain by running Bitcoin at low mining rate with little forking, but incorporate additional transaction blocks, created via sortition, through reference links from the Bitcoin tree (Figure 7). This allows us to decouple the throughput from the mining rate f, and can increase the former without increasing the latter. In the context of the overall deconstruction approach (Figure 4), this decoupling is achieved by decoupling the transaction blocks from the core blockchain. Let us call the blocks in the core Bitcoin blockchain *core blocks*. Later, when we discuss latency, we will further split the functionalities of the core blocks into proposer and voter blocks to build a more complex consensus protocol, but for now we will just run Bitcoin as the basic consensus mechanism.

We now describe the structure of Prism 1.0.

1. There are two hash-threshold parameters α_c and α_t, such that $\alpha_c \leq \alpha_t$. A node mines blocks using a nonce. If the hash is less than the stringent threshold α_c, the block is a core block. If the hash is less than the relaxed threshold α_t but greater than α_c, the block is transaction block. This is a sortition of blocks into two types of blocks, and the adversary does not know which type of block it is mining until after the block has been mined.
2. The core blocks are used to determine the structure of the main chain. Each core block will reference several transaction blocks, that are then assumed to be incorporated into the ledger.
3. A block consists of the following data items.
   (a) Public key for reward
   (b) Transactions
   (c) The hash pointer to the current core block on which it is mining.
   (d) Hash pointers (references) to transaction blocks that the miner knows of and that have not been referenced in the current main chain.
   (e) Nonce, which is mined by miners.

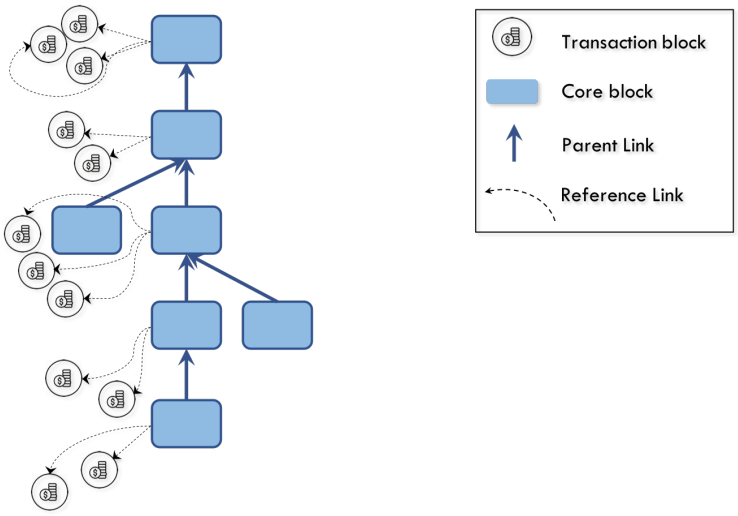


Fig. 7: Prism 1.0. Decoupling the transaction blocks from the core blocks in the Bitcoin blockchain.

If the block is a transaction block, then the hash-pointers to the current core block as well as the hash-pointers to transaction blocks are not used. If the block is a core block, then the list of transactions is not utilized. We note that the structure of the block shown in Figure 7 allows us to only package the information necessary for each type of block. The ordered list of transaction blocks is produced by ordering the transaction blocks in the order in which they are referenced by the core blocks in the main chain. For example, if the core blocks are $c_1, .., c_k$, and $R(c_i)$ denotes the list of referenced transaction blocks by c_i, then the ordered list is given by $R(c_1), .., R(c_k)$. From the ordered list of blocks, we produce an ordered list of transactions. From this ordered list of transactions, the ledger is produced by sanitizing this list to ensure that no transaction appears twice, and for every double spend, only the first spend is kept in the sanitized ledger.

Now that the key components of the protocol have been mentioned, we now explain how the protocol is run by various nodes.

- Each new transaction is broadcast to all the nodes in the network.
- Each node maintains a queue of outstanding transactions. The input to the queue is observed transactions. A transaction is cleared from the queue if the node knows of a transaction block containing the transaction.
- Each node maintains a blocktree of observed core blocks and transaction blocks.
- A node attempts to mine its new block(s) on top of the current longest chain of core blocks in its local blocktree.

1. The node includes in its block all transactions from its transaction queue that are valid with respect to the ledger formed from the current longest chain of the core blocktree.
2. The node gives reference links to all transaction blocks not currently referenced by its core main chain.

– A node that hears of a block determines its validity by checking the hash. Unlike in Bitcoin, there is no transaction validity check for a block, since the ledger is sanitized later.

In the context of the round-by-round synchronous model, the $H[r]$ honest blocks mined in the rth round are now split into $H^c[r] \sim \mathsf{Poiss}((1-\beta)f_c\Delta)$ honest core blocks and $H^t[r] \sim \mathsf{Poiss}((1-\beta)f_t\Delta)$ honest transaction blocks, where $f_c + f_t = f$. Similarly, the $Z[r]$ adversarial blocks mined in the rth round are split into $Z^c[r] \sim \mathsf{Poiss}(\beta f_c\Delta)$ adversarial core blocks and $Z^t[r] \sim \mathsf{Poiss}(\beta f_t\Delta)$ adversarial transaction blocks. The parameters f_c and f_t can be specified by choosing the appropriate value of the hash threshold α_c.

4.3 Analysis

We now analyze the proposed protocol in our network model. It is clear that the security of the protocol is the same as the security of the Bitcoin core blockchain. By setting f_c to be appropriately small (depending on β), we know that we can keep the core blockchain secure. More specifically, [10] gives one such sufficient condition, obtained by requiring that the rate of arrival of uniquely successful rounds exceeds the rate of work of the adversary:

$$f_c\Delta < \frac{1}{1-\beta} \ln \frac{1-\beta}{\beta} \tag{16}$$

Under this condition, [10] showed that the longest chain satisfies the common-prefix property as well as has positive chain quality. Similar to the argument in Conflux [16], the honest blocks in the longest chain can provide a total ordering of *all* the blocks, not just the core blocks. Hence, the throughput is given by the overall mining rate $f = f_c + f_t$. By choosing f_t such that we are at the capacity limit, i.e. $f = C/B$, we can get a total throughput of $(1-\beta)C/B$ blocks per second, or $(1-\beta)C$ transactions per second, assuming a worst case that only honest blocks carry transactions.

This seems to give us the optimal throughput efficiency $\bar{\lambda} = 1-\beta$. However, there is a catch: blocks that are mined in the same round may contain the same transactions, since transactions are broadcasted to the entire network. In the worst case, we have to assume that blocks mined at the same round contain an identical set of transactions. In this case, mining more than one block per round does not add to the (useful) throughput. Hence, the throughput, in terms of number of non-redundant blocks, is simply:

$$\mathbb{P}(H[r] > 0) = 1 - e^{-(1-\beta)f\Delta} \quad \text{blocks per second.}$$

Comparing to (7), we see that this is exactly the longest chain growth rate of Bitcoin. Since Prism 1.0 can operate at $f = C/B$, we are achieving exactly the communication-limited throughput of Bitcoin (c.f. (12)), i.e.

$$\bar{\lambda} = 1 - e^{\beta - 1}, \qquad \beta \in [0, 0.5).$$

The difference with the throughput-security tradeoff of Bitcoin is that Prism 1.0 is operating at the communication-limited regime for β all the way up to 0.5; there is no security-limited regime anymore. This is because we have decoupled transaction blocks from the core blockchain and the throughput is not security limited. In particular, the throughput does not go to zero as β goes to 0.5. But we are still not achieving the optimal throughput efficiency of $\bar{\lambda}^* = 1 - \beta$.

4.4 Transaction scheduling

To achieve optimal throughput, one can minimize the transaction redundancy in the blocks by scheduling different transactions to different blocks. Concretely, one can 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 now 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 $(1-\beta)f\Delta/q$, and so the throughput of non-redundant blocks of a given color is

$$1 - e^{-(1-\beta)f\Delta/q} \text{ blocks per round.}$$

The total throughput of non-redundant honest blocks of all colors is

$$q\left[1 - e^{-(1-\beta)f\Delta/q}\right] \text{ blocks per round.} \tag{17}$$

For large q, this approaches

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

which equals $(1-\beta)C$ transactions per second when we set $f = C/B$. Thus, we achieve the optimal throughput efficiency

$$\bar{\lambda}^* = 1 - \beta.$$

This performance is shown in the upper plot in Figure 6.

Interestingly, this maximum throughput of Prism 1.0 can be achieved whatever the choice of the block size B. In contrast, the block size B has to be set large compared to the bandwidth-delay product CD to optimize the throughput in both Bitcoin and GHOST. This extra degree of freedom in Prism 1.0 has significant implications on the tradeoff between throughput and transaction latency, which we turn to next.

4.5 Throughput-Latency tradeoff

So far we have focused on achieving the maximum throughput of Prism 1.0, without regard to latency. But transaction latency is another important performance metric. The overall latency experienced by a transaction in Prism 1.0 is the sum of two components:

1. **Processing latency** τ_p: the latency from the time the transaction enters the transaction queue to the first time a block containing that transaction is mined.
2. **Confirmation latency** τ: the latency from the time the transaction is first mined to the time it is confirmed.

We will discuss in great depth the confirmation latency in Section 5, but for now let us focus on the processing latency τ_p. It turns out that there is a tradeoff between the throughput λ and the processing latency τ_p.

We can calculate τ_p by considering the dynamics of an individual transaction queue. Let us make the simplifying assumption that transactions enter this queue at a deterministic rate. For a given total throughput λ and q, the number of transaction queues, the arrival rate into this queue is λ/q transactions per second. For stability, these transactions must also be cleared at a rate of λ/q. Thus it takes time qB/λ seconds to clear a block of B transactions from the queue and enters the blockchain. Hence,

$$\tau_p = \frac{qB}{\lambda} \quad \text{seconds.} \tag{18}$$

On the other hand, from (17), we see that the throughput λ, at the capacity limit, is given by

$$\lambda = q\left[1 - e^{-(1-\beta)C\Delta/(Bq)}\right]\frac{B}{\Delta} \quad \text{transactions per second} \tag{19}$$

We see that increasing with the number of transaction queues q increases the throughput but also increases the processing latency , as the effective arrival rate decreases. Hence tuning q can effect a tradeoff between throughput and latency. To see the tradeoff explicitly, we can eliminate q from (18) and (17) and obtain:

$$\bar{\lambda} = \frac{1-\beta}{\bar{\tau}_p \log\left(\frac{1}{1-\frac{1}{\bar{\tau}_p}}\right)} \qquad 1 < \bar{\tau}_p < \infty, \tag{20}$$

where $\bar{\tau}_p := \frac{\tau_p}{\Delta}$.

We see that as $\bar{\tau}_p$ goes to infinity, the throughput efficiency $\bar{\lambda}$ approaches $1-\beta$, the maximum throughput derived in previous section. This maximum throughput does not depend on the choice of the block size B, and this fact is consistent with our previous observation. However, for a given latency τ_p, the throughput achieved depends on the network delay Δ, which does depend on the block size B. By choosing the block size B small such that $B \ll CD$, Δ achieves

the minimum value of the propagation delay D, optimizing the tradeoff. Under this choice of the block size B, (20) becomes a tradeoff between $\bar{\lambda}$, the throughput as a fraction of network capacity, and $\bar{\tau}_p$, the processing latency as a multiple of the propagation delay (Figure 8). Thus Prism 1.0 is achieving throughput and processing latency *simultaneously* near their physical limits. Note that Bitcoin and GHOST are not only sub-optimal in their maximum throughput, their throughput-latency tradeoff is also much worse. In particular, to achieve a non-zero throughput efficiency, the block size of these protocols is much larger than the bandwidth-delay product CD, and as a consequence, the processing latency of these protocols needs to be much larger than the propagation delay.

The remaining question is whether the *confirmation* latency can also be made close to the propagation delay. This is not the case in Prism 1.0 since its confirmation latency is the same as that of Bitcoin. This latency scales with $\log 1/\varepsilon$, where ε is the confirmation error probability; this security parameter $\log 1/\varepsilon$ can be many multiples of the network delay. The question is whether we can improve upon Prism 1.0 to make the confirmation latency of the same order as the processing latency. This will be addressed in Section 5.

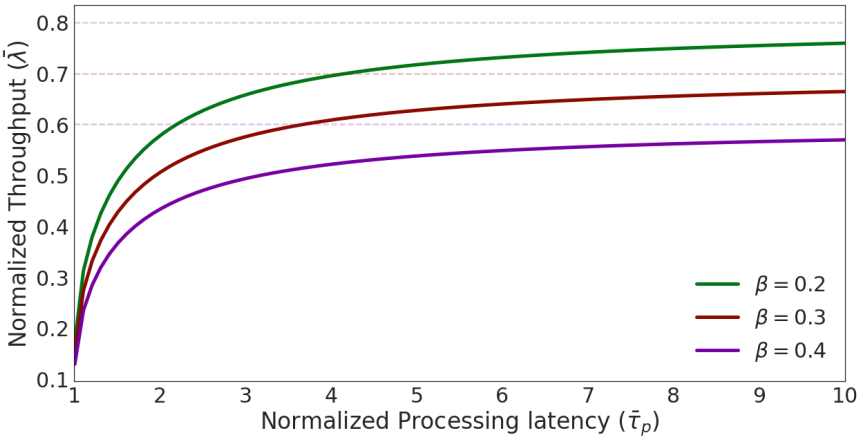


Fig. 8: Tradeoff between $\bar{\lambda}$ and $\bar{\tau}_p$ for different values of β. Throughput is normalized as a fraction of the network capacity, and the processing latency is normalized as a multiple of the speed-of-light propagation delay.

4.6 Discussions

We discuss the relationships of our protocol with several existing protocols.

1. Conflux [16] separates *links* into two types: main-chain links and reference links, but all the *blocks* go into the same blocktree. As a result, the Conflux's

security is limited by GHOST, and because Conflux is not done in conjunction with transaction scheduling, its throughput-β tradeoff is exactly the same as that of GHOST shown in Figure 6. In contrast, Prism 1.0 not only separates links into two types but also separates *blocks* into two types: core blocks, which go into the core blockchain, and transaction blocks, which are referenced by the core blocks. This separation allows the protocol to have high security and high throughput simultaneously.

2. Prism 1.0 can be viewed as similar to BitcoinNG [8] but avoiding the risk of bribery attacks since the core block does not control which transactions to put into the ledger. Moreover, the core blocks incorporate transaction blocks from various nodes, thus increasing decentralization and fairness, unlike BitcoinNG where the leaders are entitled to propose blocks till a new leader comes up.
3. Fruitchains [22] was designed as a mechanism to increase reward fairness and Prism 1.0 is designed for a totally different purpose of maximizing throughput, but the structure of Prism 1.0 has similarity to Fruitchains. The transaction blocks are roughly analogous to fruits, though there are a few differences. The fruits hang-off an earlier block in Fruitchains for short-term reward equitability, but we do not need that for throughput optimality. The 2-for-1 mining protocol [10,22] used in Fruitchains is somewhat different from our protocol. But more importantly, as we saw, transaction scheduling is crucial for achieving optimal throughput but is not present in Fruitchains.
4. Our two-threshold protocol is also similar to the ones used in mining pools [14]. Indeed, in mining-pools, partial Proof-of-Work using a higher hash threshold is used by participants to prove that they have been working (since they may be unable to generate full proof-of-work messages at regular intervals).
5. Our protocol is reminiscent of an approach called subchains [25] or weak blocks [3,31]. Both methods employ blocks with lower hash threshold ("weak blocks") along with regular blocks. However, unlike our protocol, these weak blocks have to form a chain structure. Thus, if the PoW rate of weak blocks is increased significantly, it will lead to high forking on the weak blocks, thus leading to lower throughput.
6. We note that a version of transaction scheduling appears in Inclusive [15] for incentive compatibility. In order to maximize the reward gained, selfish users select a random subset of transactions to include in the ledger. In our protocol, we show this is required to maximize transaction throughput, even with altruistic users.

5 Near physical limits: latency and throughput

Prism 1.0 scales throughput to the network capacity limit by decoupling transaction blocks from the core blockchain, so that we can run Bitcoin on the core blockchain for high security and simultaneously maximize throughput by having many transaction blocks. However, the confirmation latency of Prism 1.0 is the

same as Bitcoin, which is poor. In this section, our goal is to upgrade Prism 1.0 to design Prism, which has fast latency (on list ledger decoding and on honest transactions) as well as high throughput. The key idea is to further decouple the core blocks into proposer and voter blocks.

We start by describing the latency of Bitcoin, our baseline, in Section 5.1. In Section 5.2, we specify the Prism protocol. There are two parts to the specification: 1) the backbone (in the spirit of [10]), which specifies how the proposer blocks and voter blocks are organized, 2) how the transactions are linked from the proposer blocks. In Section 5.3, we provide a formal model for Prism based on a refinement of the model in Section 3. In Section 5.4, we prove several key properties of the Prism backbone, analogous to the common-prefix and chain-quality properties of Bitcoin proved in [10], and use it to show that it can achieve total ordering of all transactions and has optimal throughput. Finally in 5.5, we show that Prism can achieve ledger list confirmation and honest transaction confirmation with fast latency.

5.1 Bitcoin latency

Bitcoin runs the longest chain protocol where each node mines blocks on the longest chain. These blocks have two roles: proposing to become a leader and voting on its ancestor blocks for them to be elected leaders. In this protocol, a current main chain block remains in the future main-chain with probability $1-\varepsilon$ if on the order of $\log 1/\varepsilon$ successive blocks are mined over it. It can be shown (Corollary 1) that at a mining rate of f, it takes on average:

$$\mathbb{E}[\tau] = \frac{O(1)}{(1-2\beta)^2 f} \log \frac{1}{\varepsilon} \quad \text{seconds}$$

to provide $1-\varepsilon$ reliability to confirm blocks and the transactions in it. Since the expected latency τ is inversely proportional to the mining rate f, one might believe that increasing the mining rate will reduce latency. However, in the previous sections we have seen that naively increasing the mining rate will also increase forking, which reduces security in terms of β. To be more precise, Equation (8) limits the mining rate per round $\bar{f} := f\Delta$ to satisfy:

$$1 - e^{-(1-\beta)\bar{f}} > \beta\bar{f}.$$

For β close to 0.5, this leads to the following upper bound on $\bar{f}$:

$$\bar{f} < \frac{1-2\beta}{(1-\beta)^2}.$$

Therefore, this imposes a lower bound on the the expected latency of

$$\mathbb{E}[\tau] > \frac{O(1)\Delta(1-\beta)^2}{(1-2\beta)^3} \log \frac{1}{\varepsilon} \quad \text{seconds.} \tag{21}$$

Recall that physical limits impose two lower bounds on the latency: (1) the propagation delay D, and (2) $1/C \log 1/\varepsilon$. The above lower bound on Bitcoin

latency is far from these physical limits, for two reasons. First, the network delay $\Delta = B/C + D$ depends on the block size B as well as the propagation delay. From the analysis of the throughput of Bitcoin, we know from (10) that to have decent throughput, the block size B should be chosen to be significantly larger than the bandwidth-delay product CD. But this implies that the network delay Δ is significantly larger than the propagation delay. Second, the Bitcoin latency lower bound's dependency on the security parameter is much larger than $1/C \log \frac{1}{\varepsilon}$. This is because the mining rate f of Bitcoin is limited by security and hence the voting rate is much less than what is allowed by the network capacity.

By decoupling transaction blocks from the blockchain, we learnt from our analysis of Prism 1.0 that we can choose the block size B small to keep the network delay near the speed-of-light propagation delay while achieving optimal throughput. Prism inherits this property of Prism 1.0, which overcomes the first reason why Bitcoin's latency is far from the physical limit. The focus of the remaining section is the design and analysis of a voting architecture to overcome the second issue, i.e. to increase the voting rate to the physical capacity limit.

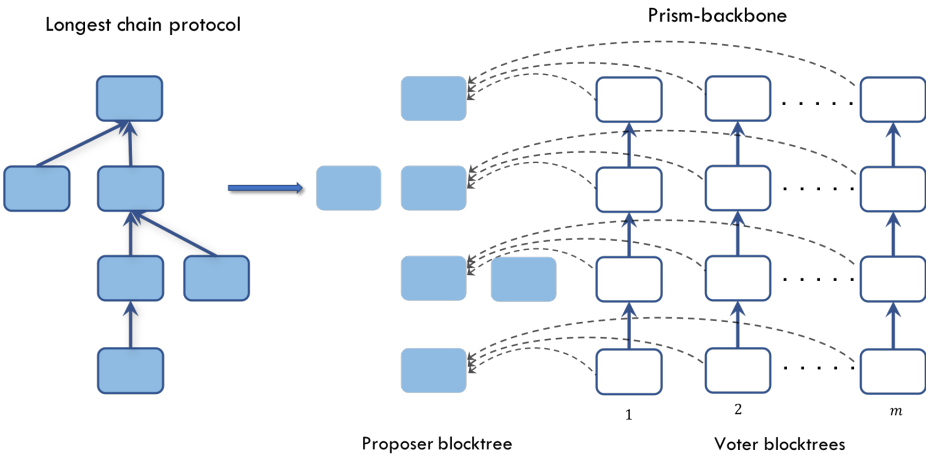


Fig. 9: Prism : Separating proposer and voter roles.

5.2 Prism

5.2.1 Prism: backbone

We begin by describing Prism's backbone, or blockchain architecture; this architecture specifies how blocks relate to each other, and which blocks find a place in the final ledger. We describe how individual blocks are packed with transactions in Section 5.2.2. Each block in Bitcoin acts as both a proposer and a voter, and this couples their proposing and voting functionalities. As a result, the security requirements of the proposer role upper bounds the mining rate, which in turn

upper bounds the voting rate. In the spirit of deconstructing the blockchain, we decouple these roles as illustrated in Figure 9. The backbone of Prism has two types of blocks: proposer and voter blocks. The role of the proposer block is to propose an extension to the transaction ledger. The voter blocks elect a leader block by voting among the proposer blocks at the same level. The sequence of leader blocks on each level determine the ledger. The voter blocks are mined on many independent blocktrees, each mined independently at a low mining rate. The voter blocktrees follow the longest chain protocol to provide security to the leader election procedure which in turn provides security to the transaction ledger. We now state the Prism backbone protocol from a node's local view:

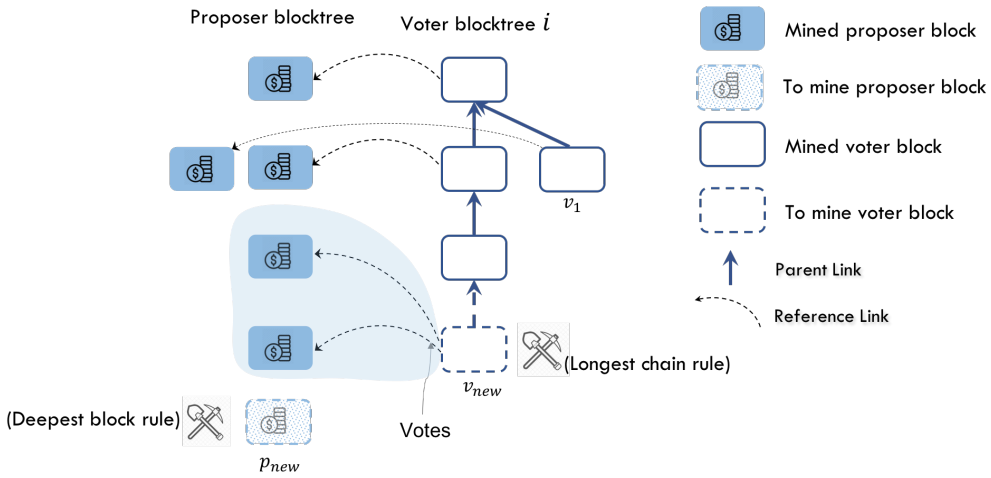


Fig. 10: Prism: Honest users mine a proposer block p_{new} at a level one deeper than the current deepest level—in this example, p_{new} has level 5. The voter block v_{new} is mined on the longest chain. It votes (via reference links) to all proposer block on level $\{3, 4\}$ because its ancestors have votes only till level 2. Since v_1 is not part of the main chain, it's vote will not be taken into account for leader block election.

- *Proposer blocks:* Proposer blocks are mined on a proposer blocktree as shown in Figure 9, using the longest-chain rule. The *level* of a proposer block is defined as the length of its path from the genesis block. Each proposer block includes a reference link to an existing proposer block to certify its level.
- *Voter blocks:* Voter blocks are mined independently on m separate voter trees, as shown in Figure 9. Each of these blocktrees has its own genesis block and nodes mine on the longest chain. Each voter block votes one or more proposer blocks using reference links.
- *Vote Interpretation:* Each voter blocktree votes only on one proposer block at each level in the proposer blocktree. The vote of the voter blocktree is

decided by the vote cast by the earliest voter block along its main chain. Thus the proposer blocks on each level has m votes in total. A voter block voting on multiple proposer blocks at the same level is invalid.

- *Voting rule:* The ancestor blocks of a voter block are all the blocks on its path to the genesis block. A voter block has to vote on a proposer block on all the levels which have not been voted by its ancestors voter blocks.
- *Leader blocks:* The proposer block that receives the most votes on each level is the) leader block for that level. The sequence of leader blocks across the levels is called the *leader sequence*.
- *Sortition:* A block is mined *before* knowing whether it will become a proposer block or a voter block. In case it becomes a voter block, the miner will not know a priori which voter tree it will be part of. This is enforced by using a sortition scheme, similar to the sortition described earlier in Prism 1.0 between core and transaction blocks, except now the hash range is divided into $m + 1$ instead of 2 intervals. This division is adjusted to ensure that the proposer tree has proposer rate f_p and each of the m voter trees have block mining rate f_v, with a total mining rate $f = f_p + mf_v$. By the security property of the hash function, a miner cannot know which range it will land in. This ensures that the adversarial power is uniformly distributed across the different voter trees and hence we assume the adversarial hash power is β in each of the voter trees as well as the proposer chain.
- *Choice of parameters:* Our protocol can operate with general settings of the parameters, but for good performance we set some specific numbers here. We set the block size $B = 1$ transaction, which as we discussed earlier is a good choice both for latency and for throughput. Under the assumption that $CD \gg 1$, the network delay $\Delta = D$, the smallest possible. To minimize latency, we want to maximize the vote generation rate, i.e. we set $f = C$, the capacity limit. The mining rate $\bar{f}_v := f_v D$ on each voting tree is chosen such that each voting tree is secure under the longest chain rule and according to (16) it should satisfy

$$\bar{f}_v < \frac{1}{1-\beta} \log \frac{1-\beta}{\beta}. \tag{22}$$

We also set the proposer and voter mining rates to be the same, i.e. $f_p = f_v$. This is not necessary but simplifies the notation in the sequel. This implies that

$$\begin{aligned} m &= \frac{CD}{\bar{f}_v} - 1 \\ &\geq \frac{(1-\beta)}{\log(\frac{1-\beta}{\beta})} \cdot CD - 1 \end{aligned} \tag{23}$$

i.e. the number of voting trees is at least proportional to the bandwidth-delay product CD. This number is expected to be very large, which is a key advantage of the protocol. The only degree of freedom left is the choice of $\bar{f}_v$. We will return to this issue in Section 5.5 when we discussed the fast confirmation latency of Prism.

5.2.2 Prism: transaction structure

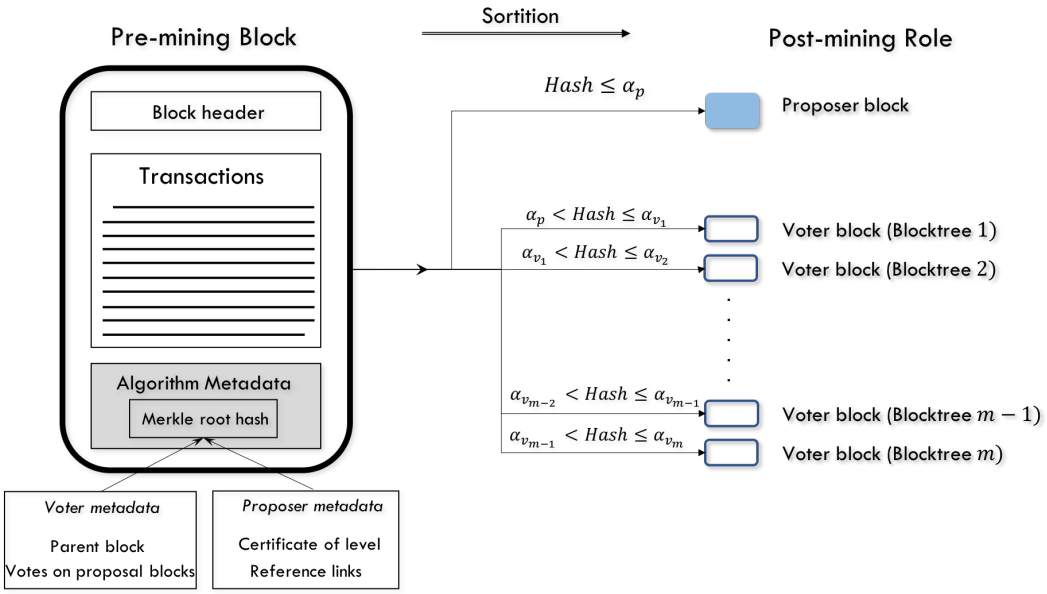


Fig. 11: Summary of the block structure and the sortition procedure.

Having presented the Prism backbone protocol, we now proceed to describe how transactions are embedded into this backbone structure. We also give more details on the content of the blocks. In Prism, the structure of the block has to be fixed prior to determining whether the block will be a proposer-block or a voter-block; therefore both blocks will have the same fundamental structure.

Block contents: Any block needs to contain the following data items:

1. *Hash of Voter / Proposal Metadata* The block includes the hash of voter metadata as well as the hash of proposal metadata. Once it is known which type of block it becomes, then that particular metadata is attached to the block.
2. *Transactions*: Each block contains transactions that are not in the current ledger, and furthermore are not included in any of the referred blocks. The honest nodes utilize transaction scheduling given in Section 4.4 to choose a random subset of transactions.
3. *Nonce*: The nonce is a string discovered during PoW mining that seals the block; a valid nonce ensures that the hash value of the block (concatenated with the nonce) is less than a predetermined threshold. Our sortition mechanism uses the value of the hash to decide what type of block it becomes. In particular, we produce a sortition as follows:

- Hash $< \alpha_p \Rightarrow$ Block is a proposer.
- $(i-1)\alpha_v + \alpha_p <$ Hash $< i\alpha_v + \alpha_p \Rightarrow$ Block belongs to voter blocktree i .
- The proposer PoW rate f_p will be proportional to α_p, and PoW rate on any voter blocktree f_v is proportional to α_v.

Voter Block Metadata: The voter block meta-data needs to contain two items: votes on the proposal blocks as well as where the parent block on the voter blocktree where it needs to be attached.

1. *Votes*: The votes are of the form (ℓ, p_ℓ) for $\ell \in \{\ell_{\min}, \ell_{\max}\}$ where p_ℓ is a hash of a proposer-block on level ℓ. The honest strategy is to vote on the block on level ℓ that it heard about the earliest. Also, for honest nodes $\ell_{\max}$ is the highest level that the node knows of, and $\ell_{\min}$ is the smallest level for which some blocktree has not yet voted.
2. *Parent link*: A voter block specifies one parent in each voter blocktree, $b_i, i = 1, 2, ..., m$. Honest nodes specify b_i as the leaf node in the longest chain of blocktree i. For efficiency, instead of storing all the m potential parents in the block, these potential parents are specified in a Merkle tree and only the root of the Merkle tree is specified in the block. If a block ultimately ends up in voter blocktree i, then it provides a proof of membership of b_i in the Merkle tree and is attached to voter block b_i.

Proposal Block Metadata: A proposal block needs to contain two metadata items, described as follows.

1. *Certificate of level*: A block that wants to be proposed for level ℓ contains a hash of a block in level $\ell - 1$.
2. *Reference links*: A proposal block p contains a list of reference links $\mathcal{R}(p)$ to other blocks. The honest strategy is to include a reference link to each proposal and voter block that is a leaf in the DAG. Here, the directed acyclic graph (DAG) is defined on the set of nodes equal to all the proposer and voter blocks. The edges include reference links from the proposer blocks to the voter blocks as well as the links from each voter block to its parent.

5.2.3 Generating the ledger

Given a sequence of proposer-blocks, $p_1, ..., p_\ell$, the ledger is defined as follows (our ledger construction procedure is similar to the one in Conflux [16]). Each proposer-block p_i defines an epoch; in that epoch is included all the blocks referenced from that proposer block p_i, as well as all other blocks reachable from p_i but not included in the previous epochs. In each epoch the list of blocks is sorted topologically (according to the DAG), and ties are broken deterministically based on the content of the block. The ledger comprises the list of blocks ordered by epoch. Since the transactions in the reference blocks may have been mined independently, there may be redundant transactions or double-spends in the ledger of transactions. Any end-user can create a sanitized version of this ledger by keeping only the first time a given transaction output is spent. We

note that this approach decouples transaction validation from mining, unlike in Bitcoin, where nodes only include valid transactions with respect to the current ledger.

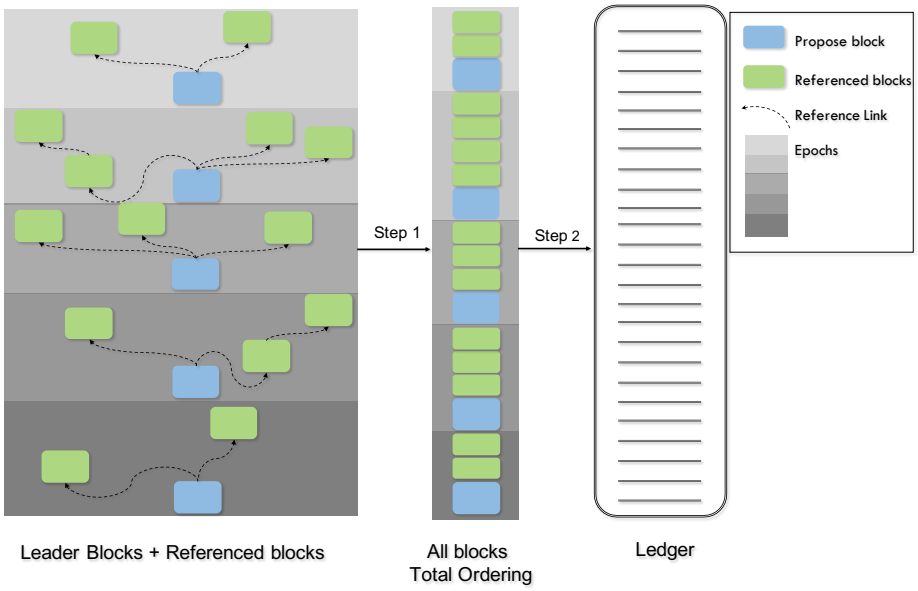


Fig. 12: Prism: Generating the ledger. The proposer blocks for a given proposer block sequence are highlighted in blue, and the referenced blocks are shown in green. Each shade of grey corresponds to an epoch. In Step 1, all the blocks are incorporated, and in Step 2, they are expanded out to give a list of transactions.

5.3 Prism: model

We provide a formal model of Prism based on a refinement of the round-by-round synchronous model in Section 3.

Let $H_i[r]$ and $Z_i[r]$ be the number of voter blocks mined by the honest nodes and by the adversarial nodes in round r on the i-th voting tree respectively, where $i = 1, 2, .., m$. Note that by the sortition process, $H_i[r], Z_i[r]$ are Poisson random variables with means $(1-\beta)f_v\Delta$ and $\beta f_v\Delta$ respectively, and are independent, and independent across trees and across rounds. Similarly, $H^p[r], Z^p[r]$ are the numbers of proposer blocks mined by the honest nodes and by the adversarial nodes in round r respectively, they are also Poisson, with means $(1-\beta)f_p\Delta$ and $\beta f_p\Delta$ respectively. They are independent, and independent of all the other random variables. We will define $X_i[r]$ ($X^p[r]$), which is 1 if $H_i[r] \geq 1$ ($H^p[r] \geq 1$) and zero otherwise. and define $Y_i[r]$ ($Y^p[r]$), which is 1 if $H_i[r] = 1$ ($H^p[r] = 1$) and zero otherwise. We denote $H_i[r_1, r_2] := \sum_{r=r_1+1}^{r_2} H_i[r]$, similarly for Z_i, X_i and Y_i. The interval $[r_1, r_2]$ denotes rounds $\{r_1+1, r_1+2, \cdots, r_2-1, r_2\}$.

The adversary decides to release blocks (either kept in private or just mined) in each tree (either the proposer tree or one of the voter trees) after observing all the blocks mined by the honest nodes in all trees in that round. It can also decide which proposal block each honest voter votes on (but it cannot remove the vote or violate protocol, e.g., by changing the proposal block level of the vote.) The adversary is powerful as it can observe what is happening in *all* the trees to make a decision on its action in any individual tree. In particular, this adversarial power means that the evolution of the trees are *correlated* even though the mining processes on the different trees are independent because of the sortition procedure. This fact makes the analysis of Prism more subtle, as we need to prove some kind of law of large numbers across the voter trees, but can no longer assume independence.

As in our basic model (which follows [10]), all the nodes have a common view of the (public) trees at the end of each round.

5.4 Total transaction ordering at optimal throughput

In this subsection, we show that Prism can achieve total transaction ordering for any $\beta < 0.5$. Following the framework of [10], we do so by first establishing two backbone properties: common-prefix and quality of a certain *leader sequence* of proposer blocks, analogous to the longest chain under Bitcoin.

The blockchain runs for $r_{\max}$ rounds, which we assume to be polynomial in m i.e, $r_{\max} = \text{poly}(m)$. 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 voting 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,

$$p^*_\ell(r) := \underset{p \in \mathcal{P}_\ell(r)}{\text{argmax}}\, V_p(r),$$

where tie-breaking is done in a hash-dependent way.

A *proposer sequence* up to level ℓ at round r is given by $[p_1, p_2, \cdots, p_\ell]$, where $p_j \in \mathcal{P}_j(r)$. The *leader sequence* up to level ℓ at round r, denoted by $\texttt{LedSeq}_\ell(r)$, is a proposer sequence with $p_j = p^*_j(r)$, in other words

$$\texttt{LedSeq}_\ell(r) := [p^*_1(r), p^*_2(r), \cdots, p^*_\ell(r)]. \tag{24}$$

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

The leader block $p^*_\ell(r)$ for a fixed level ℓ can change with round r due to the adversary displacing some of the votes from their voter chains. However as r increases, changing $p^*_\ell(r)$ is harder as the votes are embedded deeper in their respective voter chains. The theorem below characterizes this phenomenon.

Theorem 1 (Leader sequence common-prefix property). *Suppose* $\beta < 0.5$. *For a fixed level* ℓ, *we have*

$$\texttt{LedSeq}_\ell(r) = \texttt{LedSeq}_\ell(r_{\max}) \quad \forall r \geq R_\ell + r(\varepsilon) \tag{25}$$

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 C. □

Theorem 1 is similar in spirit to Theorem 15 of [10], which establishes the common-prefix property of the longest chain under the Bitcoin backbone protocol. Hence, the leader sequence in Prism plays the same role as the longest chain in Bitcoin. Note however that the leader sequence, unlike the longest chain, is not determined by parent-link relationships between the leader blocks. Rather, each leader block is individually determined by the (many) votes from the voter chains.

The common-prefix property of Bitcoin's longest chain guarantees consistency of the ledger produced by the protocol. Similarly, the common-prefix property of the leader sequence guarantees consistency of the ledger produced by Prism . Ledger liveness of Bitcoin , on the other hand, is guaranteed by the chain-quality property. The proposer block mining policy (Section 5.2.1) is to mine each proposer block at the highest level available, with a reference link to a parent block that certifies the new block's level. If we define a tree with proposer blocks and these reference links as the edges, then the users are in fact mining over the longest proposer chain. Therefore, intuitively, the chain-quality guarantees of Theorem 16 in [10] should hold for the leader sequence, resulting in the liveness of the Prism ledger. This result is formalized below.

Theorem 2 (Liveness). *Assume* $\beta < 0.5$. *Once a transaction enters into a mined block, w.p* $1-\varepsilon$ *it will eventually be pointed to by a permanent leader sequence block after a finite expected latency*

Proof. See Appendix C. □

Together, Theorem 1 and Theorem 2 guarantee that Prism achieves a consistent and live total ordering of all transactions, but requiring a confirmation latency of order $\log \frac{m}{\varepsilon}$ for a confirmation error probability of ε. Just like the longest chain in the core tree of Prism 1.0, the leader sequence blocks of Prism orders all the transactions in the transaction blocks they refer to. In conjunction with transaction scheduling, Prism, just like Prism 1.0, achieves a worst-case optimal throughput of $(1-\beta)C$ transactions per second.

While being able to achieve a total ordering of transactions at optimal throughput is an important property of a consensus protocol, this goal was already accomplished in the simpler Prism 1.0, using the longest chain protocol on the core tree. The use of a more sophisticated voting structure in Prism is to meet a more ambitious goal: fast confirmation latency near physical limit. We turn to this goal in the next subsection.

5.5 Fast confirmation of ledger list and honest transactions

5.5.1 An example

Let us start with an example to get a feel for why we can confirm with latency much shorter than Bitcoin latencies.

Suppose $CD = 5000$, $D = 0.2$ seconds and $\bar{f}_v = 0.1$ (corresponding to a tolerable $\beta = 0.49$), so we have $m \approx 5000/0.1 = 50,000$ votes at each level and votes are mined at rate $1 - e^{-\bar{f}_v} = 1 - e^{-0.1} \approx 0.1$ votes per round per voter chain. Two proposer blocks are mined from genesis at round 1 and appear in public at level 1. At the next round, on average, $0.1 \cdot 50,000 = 5000$ votes are generated to vote on these two proposer blocks. At the round after that, only the voter chains that have not voted in the last round can generate new votes, and on the average $0.1 \cdot (50000 - 5000) = 4500$ votes will be generated. The total number of chains that have not voted after r rounds is:

$$m(1 - 0.1)^r,$$

decreasing exponentially with r. After 20 rounds, or 4 seconds, about 6000 chains have not voted. That means at least one of the two proposer blocks has at least $(50,000 - 6000)/2 = 22,000$ votes.

At this point:

1. If the adversary later presents a proposer block that it has mined in private at this level, then it can gather at most 6000 votes and therefore not sufficient to displace both these two public blocks and become a leader block. Thus, no private attack is possible, and we are ensured that anytime in the future one of the two proposer blocks already in public will be a leader block.
2. If one of the public proposer blocks has significantly more votes than the other block, by much more than 6000, then we can already confirm that the current leader block will remain the leader forever, because there are not enough new votes to change the ordering.

Interestingly, when these events occur, an observer observing the public blockchain knows that it occurs. Moreover, we know that the first event will definitely occur after r rounds, where r is the smallest number of rounds such that

$$m(1 - 0.1)^r < \frac{m - m(1 - 0.1)^r}{2},$$

i.e. $r = 12$ rounds.

The above analysis gives some evidence that fast confirmation is possible, but the analysis is simplistic, due to three reasons:

1. $1 - e^{-\bar{f}_v}$ is the growth rate of each voter chain if every node follows the protocol. However, some fraction of the voter blocks on the chains may belong to adversarial nodes who decide not to vote on any pro poser block; in fact, this fraction may be greater than β due to selfish mining [9,26,20]. Thus, the number of outstanding votes calculated above may be on under-estimation.

However, we do know that the longest chain's quality is non-zero (Theorem 16 of [10]). Hence, the qualitative behavior of the voting dynamics remain the same but the voting rate has to be reduced to account for adversarial behavior.

2. The above analysis assumes that votes that have already been cast cannot be reversed. That is not true because the adversary can grow private chains to reverse some of the votes. However, because the adversary power is limited, the fraction of such votes that can be reversed is also limited. Moreover, as we wait longer, the fraction of votes that can be reversed in the future also gets smaller because the votes get embedded deeper in their respective chains. This needs to be accounted for, but again the qualitative picture from the simplistic analysis remains unchanged: after waiting for a finite number of rounds, one can be sure that the eternal leader block will be one of a list of current public proposer blocks.
3. The simplistic analysis assumes the total number of votes that are mined at each round is *deterministic*, at the mean value. In reality, the actual number of votes mined at each round is random, fluctuating around the mean value. However, due to a law of large numbers effect, which we will formally show, the fluctuations will be very small, since there are large number of voting chains. This justifies a deterministic view of the dynamics of the voting process.

5.5.2 Fast list confirmation

We convert the intuition from the above example to a formal rule for fast confirming a *list* of proposer blocks, which then allows the confirmation of a list of proposer sequences. The idea is to have *confidence intervals* around the number of votes cast on each proposer block. Figure 13 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 $[\underline{V}_n(r), \overline{V}_n(r)]$ for the votes on proposer block p_n bounds the maximum number of votes the block can lose or gain from votes not yet cast and from the adversary reversing the votes already cast. In the running there is also potentially a private hidden block, with an upper bound on the maximum number of votes it can accumulate in the future. We can fast confirm a list 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: Let $\mathcal{P}_\ell(r) = \{p_1, p_2 ...\}$ be the set of proposer blocks at level ℓ at round r. Let $U(r)$ be the number of voter blocktrees which have not voted for any proposer block in $\mathcal{P}_\ell(r)$. Let $V_n^d(r)$ be the number of votes at depth d or greater for proposer block p_n at round r. Let $V_{-n}^d(r)$ be the the number of votes at depth d or greater for a proposer blocks in the subset $\mathcal{P}_\ell(r) - \{p_n\}$. Define:

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

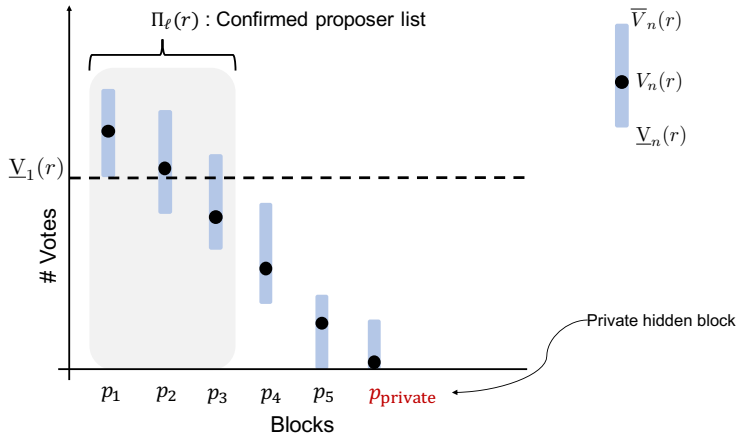


Fig. 13: In the above example, public proposer block p_1 has the largest lower confidence bound, which is larger than the upper confidence bound of the private block. So list confirmation is possible and the list confirmed is $\Pi_\ell(r) = \{p_1, p_2, p_3\}$.

and

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

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

$$\underline{V}_{\text{private}}(r) := 0,$$

$$\overline{V}_{\text{private}}(r) := m - \sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r).$$

Proposer list confirmation policy: If

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

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

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

The following theorem shows that one can confirm proposer lists up to level ℓ with an expected latency independent of ε; moreover the final leader sequence is contained in the product of the confirmed lists.

Theorem 3 (List common-prefix property). *Suppose $\beta < 0.5$. Suppose the first proposer block at level ℓ appears at round R_ℓ. Then w.p. $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{16 \log m}}$,*

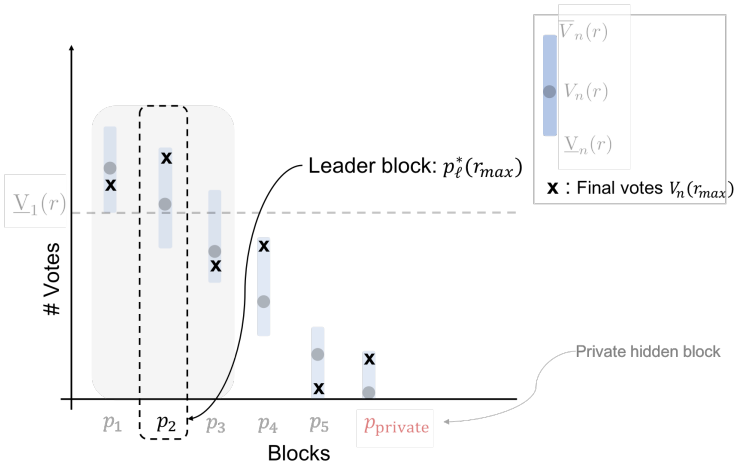


Fig. 14: A possible scenario by the final round.

we can confirm proposer lists $\Pi_1(r), \Pi_2(r), \ldots, \Pi_\ell(r)$ *for all rounds* $r \geq R_\ell + R_\ell^{conf}$*, where*

$$\mathbb{E}[R_\ell^{conf}] \leq \frac{2808}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)} + \frac{256}{(1-2\beta)^6 \bar{f}_v m^2}. \tag{26}$$

Moreover, w.p. $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{16 \log m}}$*,*

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

Proof. See Appendix D. □

Let us express the latency bound (26) in terms of physical parameters. If we set the voting rate $\bar{f}_v$ equal to the largest possible given the security constraint (22):

$$\bar{f}_v = \frac{1}{1-\beta} \log \frac{1-\beta}{\beta},$$

then according to (23), we have

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

With this choice of parameters, and in the regime where the bandwidth-delay product CD is large so that the second term in (26) can be neglected, the expected latency for list confirmation is bounded by

$$c_1(\beta) D \quad \text{seconds},$$

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. This is the constant part of the latency versus security parameter tradeoff of Prism in Figure 1.

Since CD is very large in typical networks, a confirmation error probability exponentially small in CD 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 (22) and increase the number of voting chains. More specifically, we set

$$\bar{f}_v = \frac{CD}{\log \frac{1}{\varepsilon}}, \tag{27}$$

resulting in

$$m = \log \frac{1}{\varepsilon} - 1 \approx \log \frac{1}{\varepsilon}$$

yielding the desired security parameter. Indeed, the above equation for $\bar{f}_v$ is valid only if the security condition for $\bar{f}_v$ is satisfied, i.e., when $\varepsilon < e^{-\frac{CD(1-\beta)}{\log \frac{1-\beta}{\beta}}}$.

Again neglecting the second term in (26), the corresponding latency bound is

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

where

$$c_2(\beta) := \frac{2808}{(1-2\beta)^3} \log \frac{50}{(1-2\beta)}.$$

This is the linearly increasing part of the tradeoff curve for Prism in Figure 1, with slope inversely proportional to the network capacity C.

Some comments:

- We have shown that latency and confirmation reliability can be traded off by choosing different values of $\bar{f}_v$ and m. But these are protocol parameters. We believe that one can achieve a similar tradeoff by changing the *confirmation rule* while fixing these protocol parameters. This would allow the recipient of a transaction to choose the level of security guarantee that they require and wait accordingly. A detailed analysis of this adaptive confirmation rule is left for future work.
- While the latency bounds exhibit the correct qualitative behavior, the constants involved are rather large. This is due to two reasons. First, our proofs are optimized for clarity rather than yielding the best constants. In particular, we structure the proofs to mirror as close as possible the backbone protocol framework of [10]. Second, in our analysis, we give full power to the adversary in choosing which proposer blocks the honest voter blocks vote on. Thus the bounds need to account for the worst case, where the number

of votes on the proposer blocks are very close. With a less crude model, one can improve the bounds considerably. We expect the actual latency to be much smaller than our bounds, but this conjecture is best validated by experiments rather than more theory.

5.5.3 Fast confirmation of honest transactions

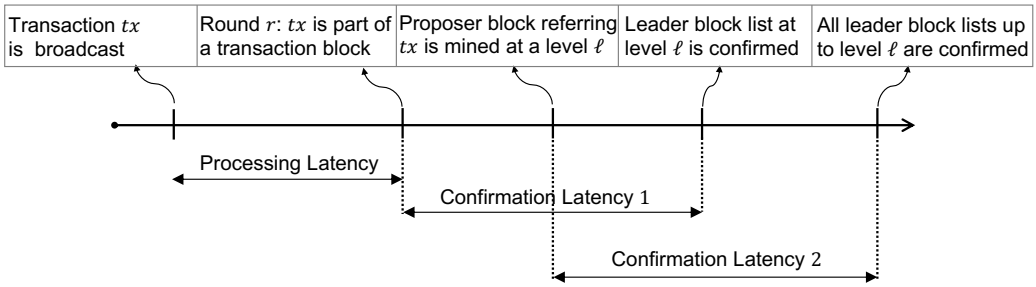


Fig. 15: Components of the latency: a) Processing latency is addressed in Section 4.5, b) Confirmation latency 1 is analyzed in Theorem 4, and c) Confirmation latency 2 is analyzed in Theorem 3.

In the previous subsection we have shown that one can fast confirm a list of proposer block sequences which is guaranteed to contain the prefix of the final totally ordered leader sequence. As discussed in Section 5.2.3, 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 the transaction is guaranteed to be in the final total ordered sanitized ledger, and the transaction can be fast confirmed. (See Figure 5.) 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 will add any honest transactions that have not already appeared in at least one of the ledgers in the list. Due to the positive chain-quality of the leader sequence (Theorem 2, this event of "uniquely honest" level eventually occurs. The latency of confirming honest transactions is therefore bounded by the sum of the latency of list confirmation in Theorem 3 plus the latency of waiting for this event to occur(Figure 15. The latter is given by the following theorem.

Theorem 4 (List-liveness). *Assume $\beta < 0.5$. If a 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

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

Proof. Appendix E. □

Figure 15 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 3 and 4. Repeating the analysis in the previous subsection, we see that this latency is bounded by:

$$\max\{a_1(\beta)D, \frac{a_2(\beta)}{C} \log \frac{1}{\varepsilon}\} \quad \text{seconds,}$$

where

$$a_1(\beta) := \frac{5400(1-\beta)}{(1-2\beta)^3 \log \frac{1-\beta}{\beta}} \log \frac{50}{(1-2\beta)} \tag{28}$$

$$a_2(\beta) := \frac{5400}{(1-2\beta)^3} \log \frac{50}{(1-2\beta)}. \tag{29}$$

6 Discussions

6.1 Prism: incentives

Our discussion on Prism has mostly focussed on honest users and adversarial behavior. Here we briefly discuss rational behavior, and the accompanying reward structure that incentivizes rational users to participate in the system without deviating from the proposed protocol. There are straightforward approaches to add a reward structure to Prism. Each block, whether a voter block or a proposal block, that finds its place in the ledger is assigned a block reward. To allocate transaction fees, we follow the method proposed in Fruitchains [22]. The transaction fees are distributed among the past Q blocks, where Q is a design parameter. In Prism, all blocks eventually find a place in the ledger, and thus the proportion of blocks contributed by a miner to the ledger is proportional to the hash rate of the miner. For large values of Q, our design ensures that incentives are fairly distributed and there is no gain in pursuing selfish-mining type attacks [26].

6.2 Prism: smart contracts

Most of our discussion on Prism has focused on transactions. However, we point out here that Prism is not restricted to processing transactions and can be extended to process complex smart contracts. Smart contracts are pieces of code which are executed based on the current state of the ledger. Importantly, they can depend on the *history of the ledger*, including on the timing of various events

recorded on the ledger. While many of the basic blockchain protocols such as longest-chain consensus or GHOST protocol can accommodate smart contracts, newer schemes such as Spectre and Avalanche are specific to transactions and do not confirm smart contracts. We note that Prism is naturally able to confirm the output and final-state of every smart contract at the ε-dependent latency since we get total order. We also note that this is the behavior desired in hybrid algorithms like Phantom+ Spectre .

We note that Prism has an additional attractive property for smart contracts - the ability to confirm several smart contracts at a short latency (proportional to propagation delay). Since Prism is able to confirm a list of ledgers within a short latency, this can be exploited to confirm some smart contracts. If a smart contract will execute to the same final state and output in all the ledgers in this list, then this output and final state can be confirmed for the smart contract even before confirming a unique ledger. We recall that Prism guarantees short confirmation time for honest transactions. Analogous to the notion of honest transactions, we can define a notion of *uncontested smart contracts*, where there is no alternate view of how the events happened in any of the blocks. Such uncontested smart contracts can then be shown to be confirmed within a short ε-independent latency proportional to the propagation delay - thus enhancing the scope and utility of Prism beyond payment systems.

6.3 Prism: Proof-of-Stake

In this paper we have described Prism in the proof-of-work (PoW) setting that scales the throughput by three orders of magnitude over Bitcoin . Despite this significant increase, PoW is nevertheless energy inefficient (Bitcoin consumes as much energy as medium sized countries [6]) and a leading alternative is the so-called proof-of-stake (PoS) paradigm. PoS restricts involvement in the consensus protocol to nodes who deposit a requisite amount of stake, or currency, into the system. This stake is used as a security deposit in case the nodes misbehave – for instance, by trying to unduly influence the outcome of consensus. PoS is appealing for several reasons, including the fact that it can be much more energy-efficient than PoW and also because it can be more incentive-compatible.

There are two key issues associated with designing a PoS version of Prism. First, a cryptographically secure source of randomness, that is distributed and verifiable, is needed to replace the source of randomness currently used in Prism – this includes the various mining steps, transaction scheduling and sortition operations. Second, PoS does not have the conservation of work that is implicit in PoW and this allows adversaries to "mine" at no cost in parallel and only report the outcomes that can be successfully verified – this exposes new security vulnerabilities (popularly known as the "grinding" [1] and "nothing at stake" attacks [17,12]) and a PoS design of Prism will have to contend with this attack. Both these obstacles can be successfully surmounted and will be the topic of a forthcoming paper [4].

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, and the Center for Science of Information (CSoI), an NSF Science and Technology Center (CCF-0939370) 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.

References

1. Ethereum Wiki proof of stake faqs: Grinding attacks. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs.
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. Vivek Bagaria, Giulia Fanti, Sreeram Kannan, David Tse, and Pramod Viswanath. Prism++: a throughput-latency-security-incentive optimal proof of stake blockchain algorithm. In *Working paper*, 2018.
5. Vitalik Buterin. On slow and fast block times, 2015. https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/.
6. Alex de Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.
7. C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.
8. Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.
9. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.
10. 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.
11. 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.
12. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.
13. Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. bloxroute: A scalable trustless blockchain distribution network whitepaper.
14. Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

15. 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.
16. 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.
17. Wenting Li, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 297–315. Springer, 2017.
18. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
19. 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.
20. Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.
21. Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.
22. Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017.
23. Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
24. Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2018.
25. Peter R Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.
26. Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
27. Y Sompolinsky and A Zohar. Phantom: A scalable blockdag protocol, 2018.
28. Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.
29. 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.
30. Statoshi. Bandwidth usage. https://statoshi.info/dashboard/db/bandwidth-usage.
31. TierNolan. Decoupling transactions and pow. Bitcoin Forum. https://bitcointalk.org/index.php?topic=179598.0.

Appendices

A An attack on GHOST

This attack is similar to the balancing attack in [19]. 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. These groups are mining 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, $H_1[r], H_2[r] \sim \mathsf{Poiss}(1-\beta)f\Delta/2)$ 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 $\mathbb{E}[(H_2[r] - H_1[r])^+]$. So a necessary condition for this attack to not be able to continue indefinitely with non-zero probability is

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

   or equivalently:

$$\mathbb{E}[|H_2[r] - H_1[r]|] > \beta f\Delta.$$

B Bitcoin backbone properties revisited

[10] 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 3. In addition, we will use the following notation from [10]: if $\mathcal{C}$ is a chain of blocks, then $\mathcal{C}^{\lceil k}$ is the k-deep prefix of $\mathcal{C}$, i.e. the chain of blocks of $\mathcal{C}$ with the last k blocks removed. Additionally, given two chains $\mathcal{C}$ and $\mathcal{C}'$, we say that $\mathcal{C} \preceq \mathcal{C}'$ if $\mathcal{C}$ is a *prefix* of chain $\mathcal{C}'$.

Definition 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 [10] 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 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 $\mathcal{C}$ at round r are mined by the adversary.*

The chain-quality property in [10] is parameterized by two parameters μ and k, however, the property defined here is parameterized by three parameters μ, k and r.

Definition 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. Let $r' = \frac{k}{2\bar{f}}$. Define the following events:

$$
\begin{aligned}
\mathtt{E}_1\left[r-r', r\right] &:= \bigcap_{a,b\geq 0} \left\{ Y[r-r'-a, r+b] - Z[r-r'-a, r+b] > \frac{(1-2\beta)k}{8} \right\} \\
\mathtt{E}_2\left[r-r', r\right] &:= \left\{ X\left[r-r', r\right] + Z\left[r-r', r\right] < k \right\} \\
\mathtt{E}_3\left[r-r', r\right] &:= \left\{ X\left[r-r', r\right] > \frac{k}{6} \right\} \\
\mathtt{E}\left[r-r', r\right] &:= \mathtt{E}_1[r-r', r] \cap \mathtt{E}_2[r-r', r] \cap \mathtt{E}_3[r-r', r].
\end{aligned} \tag{30}
$$

As defined in Section 3, $X[r-r',r]$ 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 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 $\mathtt{E}_1[r-r',r]$ 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 $\mathtt{E}_2[r-r',r]$ 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 $\mathtt{E}_2[r-r',r]$ 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 [10]:

Lemma 1 (Lemma 6 [10]). *Suppose the k-th block, b, of a longest chain $\mathcal{C}$ was mined by a honest node in a uniquely successful round. Then the k-th block of a longest chain $\mathcal{C}'$, at possibly a different round, is either b or has been mined by the adversary.*

Lemma 2 (Lemma 7 [10]). *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 3. *Under the event $\mathtt{E}[r-r',r]$, the last k consecutive blocks of the longest chain $\mathcal{C}$ at round r are mined in at least r' consecutive rounds.*

Proof. By definition we know that $\mathtt{E}_2[r-r',r] \supseteq \mathtt{E}[r-r',r]$. Event $\mathtt{E}_2[r-r',r]$ 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 $\mathcal{C}$ was mined on or before round $r-r'$. □

The chain-growth lemma stated below is the localized version of Theorem 13 from [10] and the proof is similar.

Lemma 4 (Chain-growth). *Under event $\mathtt{E}[r-r',r]$, where $r' = \frac{k}{2f}$, the longest chain grows by at least $\frac{k}{6}$ blocks in the interval $[r-r',r]$.*

Proof. From Lemma 2, we know that the main chain grows by at least $X[r-r',r]$ in the interval $[r-r',r]$. Since $\mathtt{E}_3[r-r',r] \supseteq \mathtt{E}[r-r',r]$ implies $X[r-r',r] > \frac{k}{6}$ and this completes the proof. □

We modify the proofs of Lemma 14 and Theorem 15 of [10] by localizing it to a particular round in order to prove our common-prefix property.

Lemma 5 (Common-prefix). *Under the event $\mathtt{E}[r_1-r',r_1]$, where $r' = \frac{k}{2f}$, the k-deep common-prefix property holds at round r_1.*

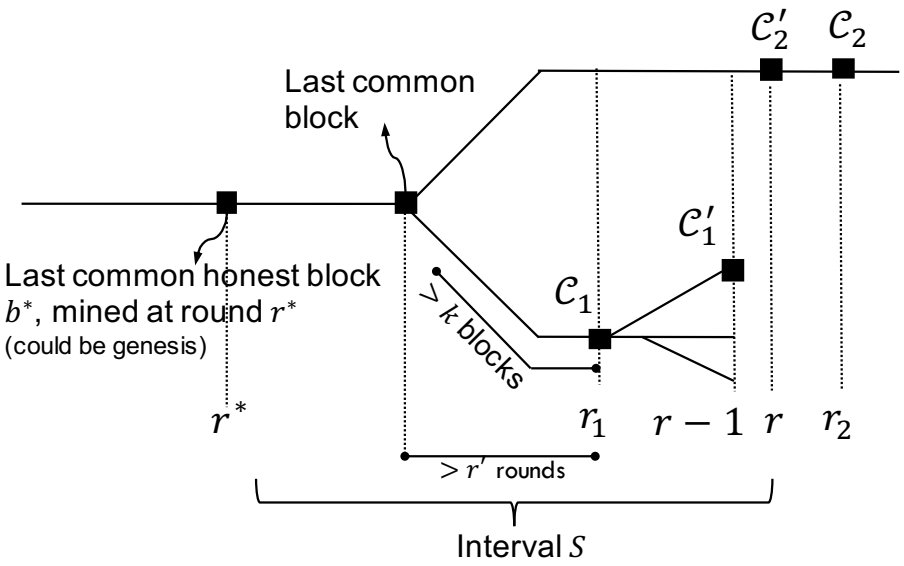


Fig. 16: Round r is the first round that the k-deep prefix of the longest chain is changed. (This is a slight modification of Figure 3 from [10].)

Proof. Consider a longest chain $\mathcal{C}_1$ in the current round r_1 and a longest chain $\mathcal{C}_2$ in a future round r_2, which violates the common-prefix property, i.e., $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$. Let r be the smallest round $r_1 \leq r \leq r_2$ such that there is a longest chain $\mathcal{C}_2'$ such that $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2'$. If $r = r_1$, define $\mathcal{C}_1' = \mathcal{C}_1$; otherwise, define $\mathcal{C}_1'$ to be a longest chain at round $r-1$. Note that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_1'$. 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 16 for an illustration. Consider the last block b^* on the common prefix of $\mathcal{C}_1'$ and $\mathcal{C}_2'$ 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

$$Z[r^*, r] \geq Y[r^*, r]. \tag{31}$$

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 $\text{len}(\mathcal{C}_1') \geq \max J$, because the honest node that mined the chain corresponding to $\max J$ position will broadcast it. Since $\mathcal{C}_2'$ is adopted at round r, it should be at least as long as $\mathcal{C}_1'$, i.e., $\text{len}(\mathcal{C}_2') \geq \text{len}(\mathcal{C}_1')$. 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 $\mathcal{C}_1'$ or in $\mathcal{C}_2'$ 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 $\mathcal{C}_1'$ and $\mathcal{C}_2'$ it is adversarial by the definition of r^*; if not, the argument follows from Lemma 1.

We assume the event $\mathtt{E}[r_1 - r', r_1]$ occurs and under $\mathtt{E}_2[r_1 - r', r_1] \supseteq \mathtt{E}[r_1 - r', r_1]$, from Lemma 3, the k-deep block of the chain $\mathcal{C}_1$ 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 (31). □

We again modify the proof of Theorem 16 of [10] by localizing it to a particular round in order to prove our chain-quality property.

Lemma 6 (Chain-quality). *Under the event $\mathtt{E}[r - r', r]$, where $r' = \frac{k}{2f}$, the (μ, k)-chain quality property holds at round r for $\mu = \frac{7+2\beta}{8}$.*

Proof. Let $\mathcal{C}$ be the longest chain at round r and denote the last k blocks in the chain $\mathcal{C}$ by $\mathcal{C}[-k] := [b_k, b_{k-1}, \cdots, b_2, b_1]$. Now define $N \geq k$ as the the least number of consecutive blocks $\mathcal{C}[-N] := [b_N, b_{N-1}, \cdots, b_2, b_1]$ 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 2, we have $N - 1 \geq X[r^*, r]$. Putting the last two statements together, we have

$$Z[r^*, r] > X[r^*, r] - (1 - \mu)k. \tag{32}$$

We assume the event $\mathtt{E}[r - r', r] = \mathtt{E}_1[r - r', r] \cap \mathtt{E}_2[r - r', r] \cap \mathtt{E}_3[r - r', r]$ occurs. Under $\mathtt{E}_2[r - r', r]$, from Lemma 3, the k-deep block of the chain $\mathcal{C}$, 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 (32) for $\mu = \frac{7+2\beta}{8}$. Therefore in the interval $[r^*, r]$, at least $(1 - \mu)k$ blocks on $\mathcal{C}[-N + 1]$ were mined by honest users. These blocks must be in $\mathcal{C}[-k]$ by definition of N. □

Since the common-prefix, chain-quality and chain-growth properties are all proved assuming the event $\mathtt{E}[r-r',r]$ 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 7. *Let* $\bar{f} \leq \frac{\log(2-2\beta)}{1-\beta}$.[5] *For any* r, $\mathbb{P}\left(\boldsymbol{E}^c\left[r-r',r\right]\right) \leq 4e^{-\gamma \bar{f} r'}$, *where* $r' = \frac{k}{2\bar{f}}$ *and* $\gamma = \frac{1}{36}(1-2\beta)^2$.

Proof. The event $\mathtt{E}^c\left[r-r',r\right]$ is a union of three events. We will upper bound the probability each of these events separately and then use union bound.

Lemma 8. *For any* r, $\mathbb{P}\left(\boldsymbol{E}_1^c\left[r-r',r\right]\right) \leq 2e^{-\frac{(1-2\beta)^2 \bar{f} r'}{36}}$. *Here* $r' = \frac{k}{2\bar{f}}$.

Proof. Let us restate the event $\mathtt{E}_1\left[r-r',r\right]$ by substituting $k = 2r'\bar{f}$:

$$\mathtt{E}_1\left[r-r',r\right] := \bigcap_{a,b\geq 0}\left\{Y[r-r'-a,r+b] - Z[r-r'-a,r+b] > \frac{(1-2\beta)\bar{f}r'}{4}\right\}.$$

Observe that the random variable $Y\left[r-r'-a,r+b\right] - Z\left[r-r'-a,r+b\right]$ can be interpreted the position of a 1-d random walk (starting at the origin) after $r'+a+b$ steps. Here $Y\left[r-r'-a,r+b\right]$, $Z\left[r-r'-a,r+b\right]$ are the number of steps taken in right and left direction respectively. The value of $\bar{f}$ is chosen s.t the random variables $Y\left[r-r'-a,r+b\right] \sim \mathsf{Bin}(r'+a+b,\frac{\bar{f}}{2})$ and as seen before $Z\left[r-r'-a,r+b\right] \sim \mathsf{Poiss}((r'+a+b)\bar{f}\beta)$; the random walk has $\frac{\bar{f}(1-2\beta)}{2}$ positive bias per step. In this random walk analogy, event $\mathtt{E}_1\left[r-r',r\right]$ 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 $\mathtt{D}\left[r-r',r\right] = \left\{Y\left[r-r',r\right] - Z\left[r-r',r\right] < \frac{1}{4}(1-2\beta)\bar{f}r'\right\}$. 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 $\frac{1}{4}(1-2\beta)\bar{f}r'$ after r' steps. We upper bound the probability of the event $\mathtt{D}\left[r-r',r\right]$:

$$\begin{aligned}
\mathbb{P}\big(\mathtt{D}\left[r-r',r\right]\big) =& \mathbb{P}\Big(Y\left[r-r',r\right] - Z\left[r-r',r\right] < \frac{1}{3}(1-2\beta)\bar{f}r'\Big) \\
=& \mathbb{P}\Big(Y\left[r-r',r\right] - Z\left[r-r',r\right] - \frac{1}{2}(1-2\beta)\bar{f}r' < -\frac{1}{6}(1-2\beta)\bar{f}r'\Big) \\
\overset{(a)}{\leq}& e^{-\gamma_1 \bar{f} r'}. \qquad (33)
\end{aligned}$$

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 $\mathtt{D}\left[r-r',r\right]$ to calculate the probability of

[5] We will assume this constraint in all our results without stating it explicitly.

the event $\mathtt{E}_1^c\left[r-r',r\right]$:

$$\begin{aligned}\mathbb{P}\big(\mathtt{E}_1^c\left[r-r',r\right]\big) &= \mathbb{P}\big(\mathtt{E}_1^c\left[r-r',r\right]\cap \mathtt{D}\left[r-r',r\right]\big)+\mathbb{P}\big(\mathtt{E}_1^c\left[r-r',r\right]\ \cap\ \mathtt{D}^c\left[r-r',r\right]\big)\\ &\le \mathbb{P}\big(\mathtt{D}\left[r-r',r\right]\big)+\mathbb{P}\big(\mathtt{E}_1^c\left[r-r',r\right]\ \big|\ D^c\left[r-r',r\right]\big)\\ &\overset{(a)}{\le} e^{-\gamma_1\bar{f}r'}+e^{-\gamma_2\bar{f}r'}\\ &\le 2e^{-\gamma\bar{f}r'}. \end{aligned} \tag{34}$$

In our random walk analogy, the event $\left\{\mathtt{E}_1^c\left[r-r',r\right]\ \bigcap\ \mathtt{D}^c\left[r-r',r\right]\right\}$ corresponds to a positive biased random walk $Y\left[r-r',r\right]-Z\left[r-r',r\right]$ 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 29 and using this lemma we obtain inequality (a) with $\gamma=\gamma_2=\frac{1}{36}(1-2\beta)^2$. □

Lemma 9. *For any* r, $\mathbb{P}\left(\boldsymbol{E}_2^c\left[r-r',r\right]\right)\le e^{-\bar{f}r'}$. *Here* $k=r'\bar{f}$.

Proof. Let us restate the event $\mathtt{E}_2\left[r-r',r\right]$ by substituting $k=2r'\bar{f}$:

$$\mathtt{E}_2\left[r-r',r\right]:=\left\{X\left[r-r',r\right]+Z\left[r-r',r\right]<2\bar{f}r'\right\}.$$

As defined in Section 3, the total number of block mined by the honest users in interval $[r-r',r]$ is $H[r-r',r]\sim\mathsf{Poiss}((1-\beta)\bar{f},r')$ and we have $H[r-r',r]\ge X[r-r',r]$. Using this we have

$$\begin{aligned}\mathbb{P}\left(\mathtt{E}_2^c\left[r-r',r\right]\right) &= \mathbb{P}\left(X\left[r-r',r\right]+Z\left[r-r',r\right]\ge 2\bar{f}r'\right)\\ &\le \mathbb{P}\left(H\left[r-r',r\right]+Z\left[r-r',r\right]\ge 2\bar{f}r'\right)\\ &= \mathbb{P}\left(\mathsf{Poiss}((1-\beta)\bar{f}r')+\mathsf{Poiss}(\beta\bar{f}r')\ge 2\bar{f}r'\right)\\ &= \mathbb{P}\left(\mathsf{Poiss}(\bar{f}r'))\ge 2\bar{f}r'\right)\\ &\le e^{-\bar{f}r'}.\end{aligned}$$

The last inequality follows from Chernoff bound[6]. □

Lemma 10. *For any* r, $\mathbb{P}\left(\boldsymbol{E}_3^c\left[r-r',r\right]\right)\le e^{-\frac{\bar{f}r'}{36}}$. *Here* $r'=\frac{k}{2\bar{f}}$.

Proof. Let us restate the event $\mathtt{E}_3\left[r-r',r\right]$ by substituting $k=2r'\bar{f}$:

$$\mathtt{E}_3\left[r-r',r\right]:=\left\{X\left[r-r',r\right]>\frac{\bar{f}r'}{3}\right\}.$$

[6] https://github.com/ccanonne/probabilitydistributiontoolbox/blob/master/poissonconcentration.pdf

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

$$\begin{aligned}
\mathbb{P}\left(\mathtt{E}_3^c\left[r-r', r\right]\right) &= \mathbb{P}\left(X\left[r-r', r\right] < \frac{\bar{f}r'}{3}\right) \\
&\leq \mathbb{P}\left(Y\left[r-r', r\right] < \frac{\bar{f}r'}{3}\right) \\
&= \mathbb{P}\left(\mathsf{Bin}(r', \frac{\bar{f}}{2}) < \frac{\bar{f}r'}{3}\right) \\
&\leq e^{-\frac{\bar{f}r'}{36}}.
\end{aligned}$$

The last inequality also follows from Chernoff bound. □

Combining Lemmas 8, 9 and 10, we obtain

$$\begin{aligned}
\mathbb{P}\left(\mathtt{E}^c\left[r-r', r\right]\right) &\leq \mathbb{P}\left(\mathtt{E}_1^c\left[r-r', r\right]\right) + \mathbb{P}\left(\mathtt{E}_2^c\left[r-r', r\right]\right) + \mathbb{P}\left(\mathtt{E}_3^c\left[r-r', r\right]\right) \\
&\leq 2e^{-\frac{(1-2\beta)^2\bar{f}r'}{36}} + e^{-\bar{f}r'} + e^{-\frac{\bar{f}r'}{36}} \\
&\leq 4e^{-\frac{(1-2\beta)^2\bar{f}r'}{36}}.
\end{aligned}$$

□

C Total ordering for Prism: proofs of Theorems 1 and 2

In Appendix B, we proved three chain properties – chain-growth, common-prefix and chain-quality – for the Bitcoin backbone under events defined in Equation (30). 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:

$$\begin{aligned}
\mathtt{E}_{1,j}\left[r-r', r\right] &:= \bigcap_{a,b\geq 0}\left\{Y_j[r-r'-a, r+b] - Z_j[r-r'-a, r+b] > \frac{(1-2\beta)k}{8}\right\} \\
\mathtt{E}_{2,j}\left[r-r', r\right] &:= \left\{X_j\left[r-r', r\right] + Z_j\left[r-r', r\right] < k\right\} \\
\mathtt{E}_{3,j}\left[r-r', r\right] &:= \left\{X_j\left[r-r', r\right] > \frac{k}{6}\right\} \\
\mathtt{E}_j\left[r-r', r\right] &:= \mathtt{E}_{1,j}[r-r', r] \cap \mathtt{E}_{2,j}[r-r', r] \cap \mathtt{E}_{3,j}[r-r', r].
\end{aligned} \tag{35}$$

Note the similarity between the above events and events defined in Equation (30). From definitions in Section 5.3, $X_j\left[r-r', r\right]$ and $Y_j\left[r-r', r\right]$ are the number successful and uniquely successful rounds respectively in the interval $[r-r', r]$ on the blocktree j. Along the same lines, $Z_j\left[r-r', r\right]$ is the number of voter blocks mined by the adversary on the blocktree j in the interval $[r-r', r]$. Events

$\mathtt{E}_{1,j}[r-r',r]$, $\mathtt{E}_{2,j}[r-r',r]$ and $\mathtt{E}_{3,j}[r-r',r]$ have corresponding interpretation of the events $\mathtt{E}_1[r-r',r]$, $\mathtt{E}_2[r-r',r]$ and $\mathtt{E}_3[r-r',r]$.

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

$$\mathtt{E}_j(r') := \bigcap_{\tilde{r}\geq r'} \bigcap_{0\leq r\leq r_{\max}} \mathtt{E}_j[r-\tilde{r},r]\,. \tag{36}$$

Lemma 11. *For any j, $\mathbb{P}\left(\mathcal{E}_j^c(r')\right) \leq 4r_{\max}^2 e^{-\gamma \bar{f}_v r'}$, where $\gamma = \frac{1}{36}(1-2\beta)^2$.*

Proof. Use Lemma 7 and apply union bound. □

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 (24).

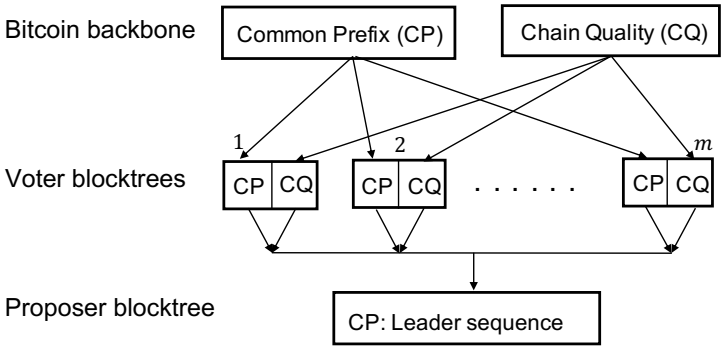


Fig. 17: Dependencies of properties required to prove the common-prefix property of the leader sequence.

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 17.

Lemma 12 (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,*

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

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 (36) and common-prefix Lemma 5 we know for under the event $\mathtt{E}_j(r')$, 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 11 we know that $\mathbb{P}\left(\mathtt{E}_j^c(r')\right) \leq \frac{\varepsilon_k}{m}$. 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 $1-\varepsilon_k$. 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 5.2.2, 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 $1-\varepsilon_k$. □

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 13. *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 4 under the event $\mathtt{E}_j(r_k)$, 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 6 under the second event $\mathtt{E}_j\left(\frac{k_1}{2\bar{f}_v}\right)$, 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

$$\begin{aligned}
\mathbb{P}\left(\mathtt{E}_j^c(r_k)\bigcup\mathtt{E}_j^c\left(\frac{k_1}{2\bar{f}_v}\right)\right) &\leq \mathbb{P}\left(\mathtt{E}_j^c(r_k)\right) + \mathbb{P}\left(\mathtt{E}_j^c\left(\frac{k_1}{2\bar{f}_v}\right)\right) \\
&\overset{(a)}{\leq} \mathbb{P}\left(\mathtt{E}_j^c(r_k)\right) + \mathbb{P}\left(\mathtt{E}_j^c\left(\frac{r_k}{6}\right)\right) \\
&\overset{(b)}{\leq} 2\mathbb{P}\left(\mathtt{E}_j^c\left(\frac{r_k}{6}\right)\right) \\
&\overset{(c)}{\leq} \frac{\varepsilon'_k}{m}. \qquad (37)
\end{aligned}$$

From Lemma 11, we see that as r' decreases, $\mathbb{P}\left(\mathtt{E}_j^c(r')\right)$ 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 11. Now applying union bound on Equation (37) over m blocktree gives us the required result. □

Proof of Theorem 1:

Proof. From Lemma 13 we know that by round $R_\ell + r(\varepsilon)$, all the voter blocktrees will have a k-deep honest voter blocks wp at least $1 - \frac{\varepsilon}{2}$[7] for $k \geq \frac{2}{\gamma} \log \frac{8mr_{\max}^2}{\varepsilon}$. Now applying Lemma 12 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 $1 - \varepsilon$. □

Worst Case vs Average Case: The confirmation policy in Lemma 13 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 1. 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 4 (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.*

Let us define the following events on the proposer blocktree:

$$\begin{aligned}
\mathtt{E}_1^p\left[r - r', r\right] &:= \bigcap_{a,b \geq 0} \left\{ Y^p[r - r' - a, r + b] - Z^p[r - r' - a, r + b] > \frac{(1-2\beta)k}{8} \right\} \\
\mathtt{E}_2^p\left[r - r', r\right] &:= \left\{ X^p\left[r - r', r\right] + Z^p\left[r - r', r\right] < k \right\} \\
\mathtt{E}_3^p\left[r - r', r\right] &:= \left\{ X^p\left[r - r', r\right] > \frac{k}{6} \right\} \\
\mathtt{E}^p\left[r - r', r\right] &:= \mathtt{E}_1^p[r - r', r] \cap \mathtt{E}_2^p[r - r', r] \cap \mathtt{E}_3^p[r - r', r].
\end{aligned} \tag{38}$$

From definitions in Section 5.3, $X^p\left[r - r', r\right]$ and $Y^p\left[r - r', r\right]$ are the number of users in successful and uniquely successful rounds respectively in the interval $[r - r', r]$ on proposer blocktree, and $Z^p\left[r - r', r\right]$ 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 (30) and (35).

Lemma 14 (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* $1 - 4r_{\max}^2 e^{-(1-2\beta)^2 k/72}$.

[7] $8mr_{\max}^2 e^{-\frac{\gamma \bar{f}_v}{8} \frac{64}{\gamma \bar{f}_v (1-2\beta)} \log \frac{8mr_{\max}^2}{\varepsilon}} = 8mr_{\max}^2 e^{-\frac{8}{1-2\beta} \log \frac{8mr_{\max}^2}{\varepsilon}} < \frac{\varepsilon}{2}$.

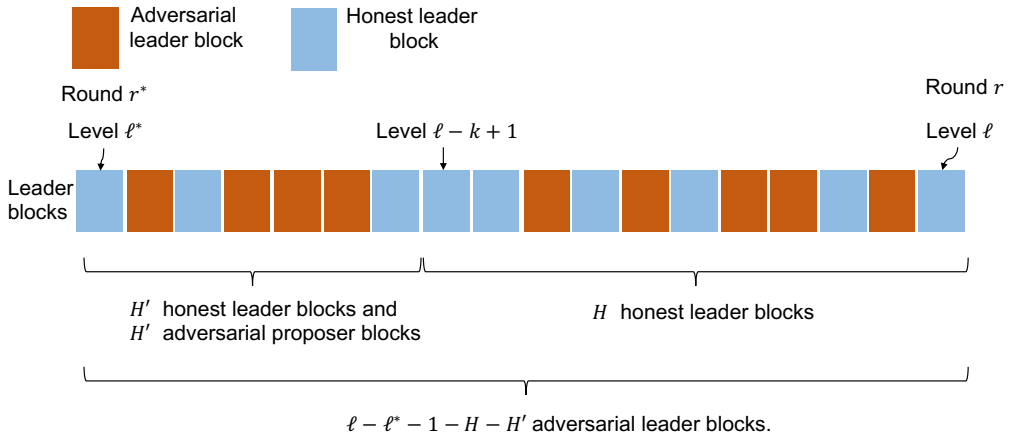


Fig. 18: The leader blocks in levels $[\ell^*, \ell]$.

Proof. Unlike the longest chain in Bitcoin, the leader sequence in Prism does not form a chain. Therefore, one cannot directly use Lemma 6 and we need to adapt its proof to prove the required property here.

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 $[\ell - k, \ell] := \{\ell - k + 1, \cdots, \ell\}$ on the leader sequence $\texttt{LedSeq}_\ell(r)$ and define:

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

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

$$H < (1 - \mu)k. \tag{39}$$

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

$$\begin{aligned} Z^p[r^*, r] &\geq (\ell - \ell^* - H - H' - 1) + H' \\ &\geq \ell - \ell^* - 1 - H \\ \text{(From Equation (39))} &> \ell - \ell^* - 1 - (1-\mu)k. \end{aligned} \tag{40}$$

Refer Figure 18 for an illustration. From the chain growth Lemma 4, we know that $\ell - \ell^* - 1 \geq X^p[r^*, r]$ and combining this with Equation (40) gives us

$$Z^p[r^*, r] > X^p[r^*, r] - (1-\mu)k. \tag{41}$$

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

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

The first proposer block on the level ℓ is mined before round r. Under the event $\mathtt{E}_2^p[r - r', r] \supseteq E^p(r')$, from Lemma 3, 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 (41) 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 11, we know that the event $\mathtt{E}^{\,p}(r')$ 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. □

The leader sequence quality defined in 4 is parameterized by two parameters r and k, whereas its counterpart definition of chain quality in [10], is parameterized only by a single parameter k. Even though our definition of 'quality' is a weaker, we show that it is suffices to ensure liveness.

Proof of Theorem 2:

Proof. Let $k := \frac{2048}{(1-2\beta)^3} \log(\frac{32mr_{\max}}{\varepsilon})$ and $k_1 := \frac{8k}{1-2\beta}$. Using Lemma 14 we know that w.p at least $1 - \varepsilon/4$, the last k_1 leader blocks have at least k honest leader blocks. From Lemma 3 and 11, w.p at least $1 - \varepsilon/4$, the deepest of these k honest leader block was proposed before the round $r - \frac{k}{2\bar{f}_v}$. Here $\frac{k}{2\bar{f}_v} = \frac{1024}{(1-2\beta)^3 \bar{f}_v} \log(\frac{32mr_{\max}}{\varepsilon})$ and now using Theorem 1, this deepest honest leader block is permanent w.p $1 - \varepsilon/4$. Therefore, the honest transaction will be permanently added to the blockchain after k_1 proposer blocks are mined. Using chain growth Lemma 4, we know that the k_1 proposer blocks will be mined in

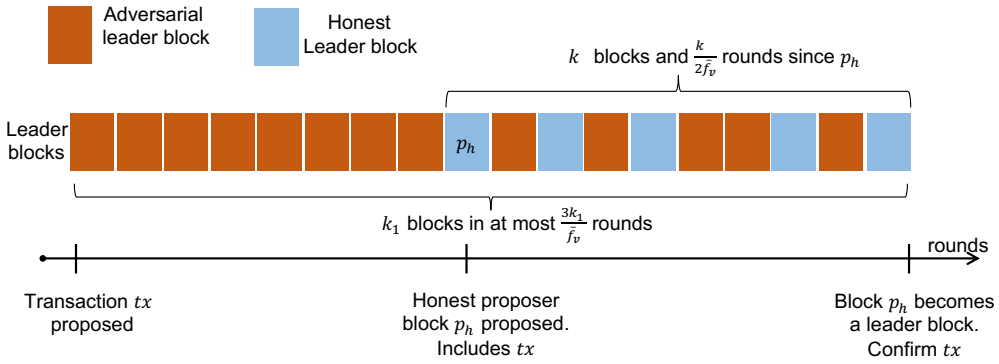


Fig. 19

no more than $\frac{3k_1}{\bar{f}_v}$ rounds w.p $1-\varepsilon/4$. Therefore, w.p $1-\varepsilon$, the transaction will be part of the permanent leader sequence in

$$\frac{3k_1}{\bar{f}_v} = \frac{3\times 2^{14}}{(1-2\beta)^3\bar{f}_v}\log\frac{32mr_{\max}}{\varepsilon} \text{ rounds.} \tag{42}$$

Refer Figure 19 for an illustration. Note that the constants in Equation (42) have not be optimized for the sake of readability. The scaling w.r.t $1-2\beta$, $\bar{f}_v$ and $\log\frac{1}{\varepsilon}$ is the main take away. □

D Fast list confirmation for Prism: Proof of Theorem 3

D.1 Voter chain properties

In Appendix C, we proved the common-prefix and the leader-sequence-quality properties by requiring the typical event defined in Equation (36) 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:

$$\begin{aligned}
\gamma &:= \frac{1}{36}(1-2\beta)^2 \\
c_1 &:= \frac{1-2\beta}{16} \\
r_{\min} &:= \frac{2\log\frac{200}{\gamma c_1}}{\gamma \bar{f}_v} \\
k_{\min} &:= \frac{4}{\gamma}\log\frac{200}{\gamma c_1} \\
\rho_{r'} &:= \max\left(\frac{c_1}{1+4\bar{f}_v r'}, \frac{(1-2\beta)c_1}{1+32\log m}\right) \\
\delta_k &:= \max\left(\frac{c_1}{1+2k}, \frac{(1-2\beta)c_1}{1+32\log m}\right) \\
\varepsilon_m &:= r_{\max}^2 e^{-\frac{(1-2\beta)c_1 m}{2+64\log m}}.
\end{aligned} \tag{43}$$

Lemma 15 (Macroscopic Typicality). *The macroscopic typical event* $\mathcal{T}$ *defined below occurs with probability* $1-\varepsilon_m$.

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

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

Proof. For a fixed r, r', the indicator random variables $\mathbf{1}\big(\mathtt{E}_j^c[r-r', r]\big)$ are identical and independent $\forall j \in [m]$. The mean of the random variable $\mathbf{1}\big(\mathtt{E}_j^c[r-r', r]\big)$ is μ, and it is at most $4e^{-\gamma\bar{f}_v r'}$ by Lemma 7. Using Chernoff bound[8] for Bernoulli random variables, $\forall a \geq 0$, we have

$$\begin{aligned}
&\mathbb{P}\Big\{\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}\big(\mathtt{E}_j^c[r-r', r]\big) \geq \mu + a\Big\} \leq e^{-\frac{ma^2}{a+2\mu}} \\
\overset{(1)}{\Rightarrow}&\mathbb{P}\Big\{\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}\big(\mathtt{E}_j^c[r-r', r]\big) \geq 4e^{-\gamma\bar{f}_v r'} + a\Big\} \leq e^{-\frac{ma^2}{a+4e^{-\gamma\bar{f}_v r'}}} \\
\overset{(2)}{\Rightarrow}&\mathbb{P}\Big\{\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}\big(\mathtt{E}_j^c[r-r', r]\big) \geq (\delta+1)4e^{-\gamma\bar{f}_v r'}\Big\} \leq e^{-4me^{-\gamma\bar{f}_v r'}\frac{\delta^2}{\delta+2}}.
\end{aligned} \tag{44}$$

[8] 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 $r' \geq r_{\min}$[9], 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 (44), for all values of $r' \geq r_{\min}$, we get

$$\mathbb{P}(\mathtt{T}^c[r-r',r]) = \mathbb{P}\Big\{\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}\big(\mathtt{E}_j^c[r-r',r]\big) \geq \rho_{r'}\Big\} \leq e^{-m\rho_r \frac{\delta^2}{(\delta+1)(\delta+2)}}$$
$$\overset{(a)}{\leq} e^{-m\rho_r/2}$$
$$\overset{(b)}{\leq} e^{-\frac{(1-2\beta)c_1 m}{2+64\log m}}.$$

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 $\mathtt{T}^c$ is a union of at most $r_{\max}^2$ $\mathtt{T}^c[r-r',r]$ events. Using union bound we prove that the event $\mathtt{T}^c$ 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. □

Lemma 16 (Macroscopic Chain-growth). *Under the event $\mathtt{T}$, 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 15, we know that under the event $\mathtt{T}[r-r',r] \supseteq T$,

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

Applying Lemma 4 on events $\mathtt{E}_j[r-r',r]$ for $j \in [m]$ gives us the required result. □

Lemma 17 (Macroscopic Common-prefix). *Under the event $\mathtt{T}$, 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 15, we know that under the event $\mathtt{T}[r-r',r] \supseteq T$,

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

Applying Lemma 5 on events $\mathtt{E}_j[r-r',r]$ for $j \in [m]$ gives us the required result. □

Lemma 18 (Macroscopic Chain-quality). *Under the event $\mathtt{T}$, 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.*

[9] The value of $r_{\min}$ was precisely chosen to satisfy this inequality.

Proof. From the typicality Lemma 15, we know that under the event $\mathtt{T}[r-r', r] \supseteq T$,

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

Applying Lemma 4 on events $\mathtt{E}_j[r-r', r]$ for $j \in [m]$ gives us the required result. □

In Appendix C, we used microscopic properties of each voter chain to obtain the common-prefix and the leader sequence quality properties for the block-tree. 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.

D.2 Fast list confirmation policy

We repeat the definitions from Section 5.5.2. $\mathcal{P}_\ell(r) = \{p_1, p_2 ...\}$ 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 19. *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*

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

where

$$\underline{V}_n(r) := \max_{k \geq k_{\min}} (V_n^k(r) - \delta_k m)_+, \tag{45}$$

$$\overline{V}_n(r) := V_n(r) + \left(V_{-n}(r) - \max_{k \geq k_{\min}} (V_{-n}^k(r) - \delta_k m)_+\right) + U_\ell(r). \tag{46}$$

Proof. From the typicality Lemma 15, 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 $r' = \frac{k}{2\bar{f}_v}$. Under the event T, from Lemma 17, 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, $\underline{V}_{-n}(r) := \max_{k \geq k_{\min}} (V_{-n}^d(r) - \delta_k m)_+$ 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 $V_n(r_f)$. □

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

$$V_{\text{private}}(r_f) \leq \overline{V}_{\text{private}}(r) := m - \sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r) \qquad \forall r_f \geq r, \tag{47}$$

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

$$\Pi_\ell(r) := \{p_i : \overline{V}_i(r) > \max_n \underline{V}_n(r)\}. \tag{48}$$

Figures 13 and 14 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 20. *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

$$V_i(r_f) \overset{(a)}{\leq} \overline{V}_i(r) \overset{(b)}{<} \underline{V}_1(r) \overset{(c)}{\leq} V_1(r_f) \qquad \forall r_f \geq r. \tag{49}$$

The inequality (b) is by definition of $\Pi_{\ell'}(r)$, and the inequalities (a) and (c) are due to confidence intervals from Lemma 19. Equation (49) 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. □

Lemma 20 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.

D.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 $\bar{f}_p = \bar{f}_v$. Define $\Delta_0 := \frac{12 r_{\min}}{1-2\beta}$.

Lemma 21. *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 $k \geq \frac{(1-2\beta)\bar{f}_v \Delta_r}{24}$ and is also greater than $k_{\min}$.*

Proof. From the typicality Lemma 15, we know that the event T occurs w.p $1-\varepsilon_m$. Using the chain-growth Lemma 16 under the event T, we know that by round r, $1-\rho_{\Delta_r}$ fraction of the voter blocktree's main chain grows by $k_1 \geq \frac{\Delta_r \bar{f}_v}{3}$ voter blocks. Let $r' = \frac{k_1}{2\bar{f}_v}$. Next, using chain-quality Lemma 18 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 $k \geq \frac{(1-2\beta)k_1}{8} \geq \frac{(1-2\beta)\bar{f}_v \Delta_r}{24}$. 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. □

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 $c_2 := \frac{16}{\bar{f}_v(1-2\beta)^3}$.

Lemma 22. *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*

$$\begin{aligned} &\textit{Case 1.}\ \ N_\ell(R_\ell + \Delta_r) + 1 < \frac{c_1}{\rho_{\Delta_r}}, \\ \textit{Or}\quad &\textit{Case 2.}\ \ \Delta_r = c_2 m. \end{aligned} \tag{50}$$

Proof. Let us first consider Case 1. All the events here are $1-\varepsilon_m$ probability events. From Lemma 21, 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 $k \geq \frac{(1-2\beta)\bar{f}_v \Delta_r}{24}$. This implies $\sum_{p_n \in \mathcal{P}_\ell(r)} V_n^k(r) \geq m(1-4\rho_{\Delta_r})$ and from Lemma 19, we have

$$\sum_{p_n \in \mathcal{P}_\ell(r)} \underline{\mathrm{V}}_n(r) \geq m(1-4\rho_{\Delta_r}-\delta_k), \tag{51}$$

where the constant δ_k satisfies $\delta_k \leq \frac{12\rho_{\Delta_r}}{(1-2\beta)}$. Without loss of generality we assume $\underline{\mathrm{V}}_1(r) \geq \underline{\mathrm{V}}_i(r)\ \forall p_i \in \mathcal{P}_\ell(r)$, and therefore from Equation (51), we have

$$\underline{\mathrm{V}}_1(r) \geq \frac{m}{N_\ell(r)}\left(1 - \frac{16\rho_{\Delta_r}}{1-2\beta}\right)^{10}. \tag{52}$$

[10] Note that this inequality is extremely weak.

On the other hand, the upper bound on the votes on a private proposer block, p_{private}, by Equation (47) is :

$$\begin{aligned}\overline{V}_{\text{private}}(r) &< m - \sum_{p_n \in \mathcal{P}_\ell(r)} \underline{V}_n(r) \\ &\overset{(a)}{<} \frac{(1-2\beta)\rho_{\Delta_r}}{16},\end{aligned} \tag{53}$$

where the inequality (a) follows from from Equation (51). From Equations (52) and (53), it is easy to see that

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

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 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 $r(\varepsilon_m) = c_2 m$, the upper bound on the number of votes on private block, $\overline{V}_{\text{private}}(r) = 0$ and $\underline{V}_1(R_\ell + k) \geq 1 > \overline{V}_{\text{private}}(R_\ell + k)$ w.p $1 - \varepsilon_m$.

□

We now use the above Lemma 22 to calculate the expected number of rounds to confirm the proposer block list at level ℓ. For Case 1 (50) let us define the random variable:

$$R_\ell^{\text{stop}} := \min \Delta_r > \Delta_0 \; s.t \; N_\ell(R_\ell + \Delta_r) + 1 < \frac{c_1}{\rho_{\Delta_r}}. \tag{54}$$

Note that $R_\ell^{\text{stop}} = \infty$ if the inequality condition in Equation (54) is not satisfied for any Δ_r. From Lemma 22, the proposer list at level ℓ can be confirmed in $\min(R_\ell^{\text{stop}}, c_2 m)$ rounds and the next lemma calculates its expectation.

Lemma 23. *The proposer list at level ℓ can be confirmed by round $R_\ell + \min(R_\ell^{stop}, c_2 m)$ and we have*

$$\mathbb{E}[\min(R_\ell^{stop}, c_2 m)] \leq \frac{13}{(1-2\beta)} r_{\min} + \frac{48}{\bar{f}_v(1-2\beta)^3 m^3}.$$

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 3, we know that $H^p[R_\ell] \sim \mathsf{Poiss}((1-\beta)\bar{f}_v)$.

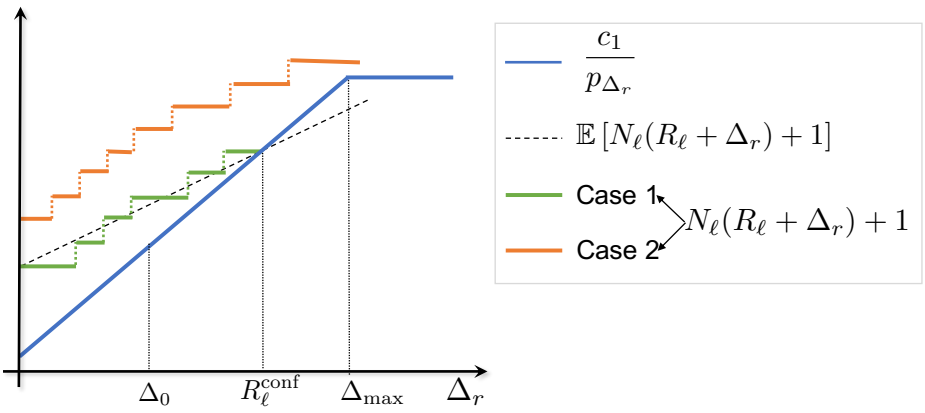


Fig. 20: Sample paths of rv $N_\ell(R_\ell+\Delta_r)+1$ falling under Case 1 and Case 2 in Lemma 22. The values of $\frac{c_1}{\rho_{\Delta_r}} = \min\left(1+4\bar{f}_v\Delta_r, 1+32\log m\right)$, $\Delta_0 = \frac{12}{1-2\beta}r_{\min}$, $\Delta_{\max} = 8\log m$.

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 F.1 that $W_\ell^p(R_\ell) \sim \mathsf{Geometric}(1-2\beta)$.
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 3, we know that $Z_\ell^p[R_\ell, R_\ell+\Delta_r] \sim \mathsf{Poiss}(\bar{f}_v\beta\Delta_r)$.

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 (54) should be constant which depends only on β. Two examples are illustrated in Figure 20. 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
$$\mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) = 1 \quad \forall \Delta_r \leq \Delta_0. \tag{55}$$
2. *Interval* $[\Delta_{max},\infty)$*:* For $\Delta_r \geq \Delta_{\max}$, we have
$$\begin{aligned}\{R_\ell^{\text{stop}} > \Delta_r\} &= \bigcap_{x\leq\Delta_r}\left\{N_\ell(R_\ell+x)+1 > \frac{c_1}{\rho_{\Delta_r}}\right\}\\ &\subseteq \bigcap_{x\leq\Delta_{\max}}\left\{N_\ell(R_\ell+\Delta_{\max})+1 > \frac{c_1}{\rho_{\Delta_{\max}}}\right\}\\ &= \{R_\ell^{\text{stop}} > \Delta_{\max}\}.\end{aligned} \tag{56}$$
This implies
$$\mathbb{P}\{R_\ell^{\text{stop}} > \Delta_r\} \leq \mathbb{P}\{R_\ell^{\text{stop}} > \Delta_{\max}\} \qquad \forall \Delta_{\max} \leq \Delta_r. \tag{57}$$

3. *Interval* (Δ_0, Δ_{max})*:* Using Equation (43), we have

$$\frac{c_1}{\rho_{\Delta_r}} = 1 + 4\bar{f}_v\Delta_r \quad \forall \Delta_r < \Delta_{\max}. \tag{58}$$

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

$$\begin{aligned}
\{R_\ell^{\text{stop}} > \Delta_r\} &= \bigcap_{x \le \Delta_r} \left\{ N_\ell(R_\ell + x) + 1 > \frac{c_1}{\rho_{\Delta_r}} \right\} \\
&\subseteq \left\{ N_\ell(R_\ell + \Delta_r) + 1 > \frac{c_1}{\rho_{\Delta_r}} \right\} \\
&\subseteq \left\{ H^p[R_\ell] + W_\ell^p(R_\ell) + Z_\ell^p[R_\ell, R_\ell + \Delta_r] + 1 > \frac{c_1}{\rho_{\Delta_r}} \right\} \\
&= \Bigg\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \\
&\qquad > \frac{c_1}{\rho_{\Delta_r}} - \left(1 + \mathbb{E}[H^p[R_\ell]] + \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]\right) \Bigg\} \\
&\overset{(a)}{=} \Bigg\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \\
&\qquad > 1 + 4\bar{f}_v\Delta_r - \left(1 + (1-\beta)\bar{f}_v + \beta\bar{f}_v\Delta_r\right) \Bigg\} \\
&\subseteq \Bigg\{ (H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]]) + W_\ell^p(R_\ell) + (Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]]) \\
&\qquad > \left(\bar{f}_v\Delta_r + \bar{f}_v\Delta_r + \bar{f}_v\Delta_r\right) \Bigg\}
\end{aligned}$$

$$\Rightarrow \{R_\ell^{\text{stop}} > \Delta_r\} \subseteq \mathtt{F}_1 \cup \mathtt{F}_2 \cup \mathtt{F}_3, \tag{59}$$

where the events are:

$$\begin{aligned}
\mathtt{F}_1 &:= \left\{ H^p[R_\ell] - \mathbb{E}[H^p[R_\ell]] \ge \bar{f}_v\Delta_r \right\} \\
\mathtt{F}_2 &:= \left\{ W_\ell^p(R_\ell) \ge \bar{f}_v\Delta_r \right\} \\
\mathtt{F}_3 &:= \left\{ Z_\ell^p[R_\ell, R_\ell + \Delta_r] - \mathbb{E}[Z_\ell^p[R_\ell, R_\ell + \Delta_r]] > \bar{f}_v\Delta_r \right\}.
\end{aligned}$$

The equality (a) follows from Equation (58). Using Chernoff bounds, we upper bound the probabilities of events the $\mathtt{F}_1$, $\mathtt{F}_2$ and $\mathtt{F}_3$:

$$\mathbb{P}(\mathtt{F}_1) \le e^{-\frac{\bar{f}_v\Delta_r}{2}} \tag{60}$$

$$\mathbb{P}(\mathtt{F}_2) \le (2\beta)^{\bar{f}_v\Delta_r} \le e^{-\frac{(1-2\beta)\bar{f}_v\Delta_r}{2}} \tag{61}$$

$$\mathbb{P}(\mathtt{F}_3) \le e^{-\frac{\bar{f}_v\Delta_r}{2}}. \tag{62}$$

From Equations (59), (60), (61) and (62), for $\Delta_0 < \Delta_r < \Delta_{\max}$, we have

$$\begin{aligned}
\mathbb{P}(\{R_\ell^{\text{stop}} > \Delta_r\}) &\le e^{-\frac{\bar{f}_v\Delta_r}{2}} + e^{-\frac{(1-2\beta)\bar{f}_v\Delta_r}{2}} + e^{-\frac{\bar{f}_v\Delta_r}{2}} \\
&\le 3e^{-\frac{(1-2\beta)\bar{f}_v\Delta_r}{2}}.
\end{aligned} \tag{63}$$

From Equations (55), (57) and (63), we have

$$\mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) \leq \begin{cases} 1 & \Delta_r \leq \Delta_0 \\ 3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}} & \Delta_0 < \Delta_r < \Delta_{\max} \\ 3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_{\max}}{2}} & \Delta_{\max} \leq \Delta_r \end{cases} \tag{64}$$

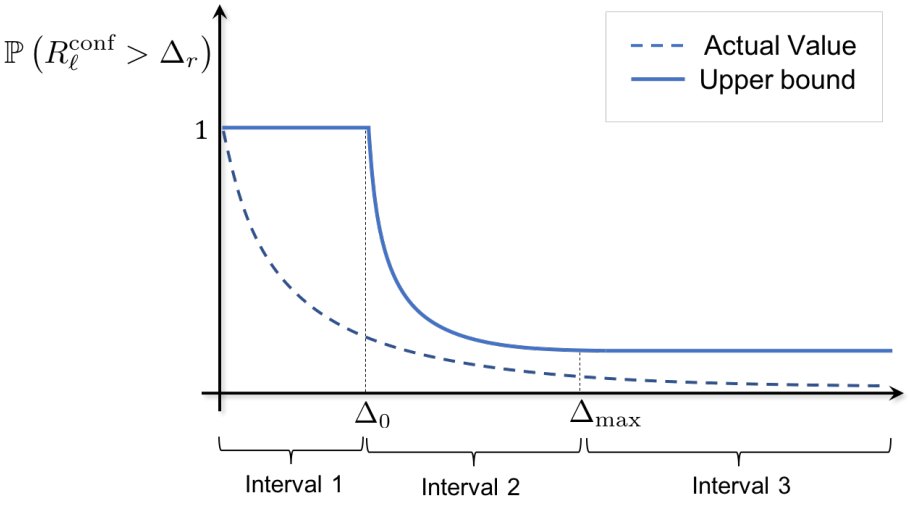


Fig. 21

Using the above expression, the expectation of $\max(R_\ell^{\text{stop}}, c_2 m)$ is given by

$$\begin{aligned}
\mathbb{E}[\min(R_\ell^{\text{stop}}, c_2 m)] &= \sum_{\Delta_r=0}^{\Delta_{\max}} \mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) + c_2 m \mathbb{P}(R_\ell^{\text{stop}} > \Delta_{\max}) \\
&\leq \Delta_0 + \sum_{\Delta_r=\Delta_0}^{\Delta_{\max}} \mathbb{P}(R_\ell^{\text{stop}} > \Delta_r) + 3c_2 m e^{-(1-2\beta) f_v \Delta_{\max}/2} \\
&\leq \Delta_0 + \sum_{\Delta_r=\Delta_0}^{\infty} \left(3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}}\right) + 3c_2 m e^{-4\log m} \\
&= \Delta_0 + \frac{6e^{-\frac{(1-2\beta)\bar{f}_v \Delta_0}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3c_2}{m^3} \\
&\leq \frac{12}{(1-2\beta)} r_{\min} + \frac{6}{(1-2\beta)\bar{f}_v} + \frac{3c_2}{m^3} \\
\mathbb{E}[\min(R_\ell^{\text{stop}}, c_2 m)] &\leq \frac{13}{(1-2\beta)} r_{\min} + \frac{48}{\bar{f}_v (1-2\beta)^3 m^3} \,.
\end{aligned} \tag{65}$$

□

Lemma 23 upper bounds the expected number of rounds to confirm the proposer block list at level ℓ. However, our goal in Lemma 20 is to conform proposer list for *all* the levels $\ell' \leq \ell$. From Lemma 22, we know that the proposer list at level ℓ' is confirmed by round $R_{\ell'} + \min(R_\ell^{\text{stop}}, c_2 m)$. Therefore, all the proposer list up to level ℓ are confirmed in the following number of rounds:

$$\begin{aligned} R_\ell^{\text{conf}} :=& \max_{\ell' \leq \ell} \left(R_{\ell'} + \min(R_{\ell'}^{\text{stop}}, c_2 m) - R_\ell \right) \\ =& \max_{\ell' \leq \ell} \left(\min(R_{\ell'}^{\text{stop}}, c_2 m) - D_{\ell',\ell} \right), \end{aligned} \tag{66}$$

where $D_{\ell',\ell} = R_\ell - R_{\ell'}$. Expression (66) 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 $\min(R_\ell^{\text{stop}}, c_2 m)$. We now calculate the expectation of the random variable in expression (66).

Lemma 24. *All the proposer lists up to level ℓ will get confirmed in the following number of rounds in expectation:*

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

Proof. Let us define

$$\begin{aligned} F(\{D_{\ell',\ell}\}_{\ell' \leq \ell}) :=& \mathbb{E}\left[R_\ell^{\text{conf}} | \{D_{\ell',\ell}\}_{\ell' \leq \ell} \right] \\ =& \mathbb{E}\left[\max_{\ell' \leq \ell} \left(\min(R_{\ell'}^{\text{stop}}, c_2 m) - D_{\ell',\ell} \right) \Big| \{D_{\ell',\ell}\}_{\ell' \leq \ell} \right] \\ \leq& \Delta_0 + \mathbb{E}\left[\max_{\ell' \leq \ell} \left(\min(R_{\ell'}^{\text{stop}} - \Delta_0, c_2 m) - D_{\ell',\ell} \right) \Big| \{D_{\ell',\ell}\}_{\ell' \leq \ell} \right] \\ \leq& \Delta_0 + \sum_{\ell' \leq \ell} \mathbb{E}\left[\left(\min(R_{\ell'}^{\text{stop}} - \Delta_0, c_2 m) - D_{\ell',\ell} \right)_+ \Big| D_{\ell',\ell} \right]. \end{aligned} \tag{67}$$

We bound each term in the summation the Equation (67) similar to steps used to Equations (65).

$$\mathbb{E}\left[\left(\min(R^{\text{stop}}_{\ell'} - \Delta_0, c_2 m) - D_{\ell',\ell}\right)_+ \Big| \left\{D_{\ell',\ell}\right\}_{\ell' \le \ell}\right] \tag{68}$$

$$= \sum_{\Delta_r = D_{\ell',\ell}+\Delta_0}^{\Delta_{\max}} \mathbb{P}(R^{\text{stop}}_{\ell'} > \Delta_r | \left\{D_{\ell',\ell}\right\}_{\ell' \le \ell}) + (c_2 m - D_{\ell',\ell})_+ \mathbb{P}(R^{\text{stop}}_{\ell'} > \Delta_{\max} | \left\{D_{\ell',\ell}\right\}_{\ell' \le \ell})$$

$$\overset{(a)}{=} \sum_{\Delta_r = D_{\ell',\ell}+\Delta_0}^{\Delta_{\max}} \mathbb{P}(R^{\text{stop}}_{\ell'} > \Delta_r) + (c_2 m - D_{\ell',\ell})_+ \mathbb{P}(R^{\text{stop}}_{\ell'} > \Delta_{\max})$$

$$\le \sum_{\Delta_r = D_{\ell',\ell}}^{\infty} \left(3e^{-\frac{(1-2\beta)\bar{f}_v \Delta_r}{2}}\right) + 3(c_2 m - D_{\ell',\ell})_+ e^{-4\log m}$$

$$\le \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell',\ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell',\ell})_+}{m^4}. \tag{69}$$

The inequality (a) follows because the random variable $R^{\text{stop}}_{\ell'}$ 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 (69) in Equation (67) we get

$$F\left(\left\{D_{\ell',\ell}\right\}_{\ell' \le \ell}\right) \le \Delta_0 + \sum_{\ell' \le \ell} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell',\ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell',\ell})_+}{m^4}. \tag{70}$$

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 (70) 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 $\{L_1, L_2, \cdots, L_i, \cdots, L_n\}$ 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 $R_{L_i} \le R_{\ell'} \le R_{L_{i+1}}$. Let us use this in Equation

(70).

$$
\begin{aligned}
&F\left(\left\{D_{\ell',\ell}\right\}_{\ell'\le\ell}\right)\\
&\le \Delta_0 + \sum_{\ell'\le\ell} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell',\ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell',\ell})_+}{m^4}.\\
&\le \Delta_0 + \sum_{i\in[n]} \sum_{L_i<\ell'\le L_{i+1}} \frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{\ell',\ell}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{\ell',\ell})_+}{m^4}\\
&\overset{(a)}{\le} \Delta_0 + \sum_{i\in[n]} (L_{i+1}-L_i)\left(\frac{6e^{-\frac{(1-2\beta)\bar{f}_v D_{L_{n+1},L_{i+1}}}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - D_{L_{n+1},L_{i+1}})_+}{m^4}\right).
\end{aligned}
\tag{71}
$$

The inequality (a) follows because $R_{\ell'} \le R_{L_{i+1}}$. Let G_j be i.i.d random variables s.t $G_j \sim \mathsf{Geometric}(\bar{f}_v)$. Since the levels L_i and L_{i+1} are mined by the honest users, we have $D_{L_{i+1},L_i} \ge \sum_{j=L_i}^{L_{i+1}} G_j$ and $D_{L_{n+1},L_i} = \sum_{j=L_{i+1}}^{L_{n+1}} G_j$. Using this in Equation (71), we get

$$
F\left(\left\{D_{\ell',\ell}\right\}_{\ell'\le\ell}\right) \le \Delta_0 + \sum_{i\in[n]} (L_{i+1}-L_i)\left(\frac{6e^{-\frac{(1-2\beta)\bar{f}_v \sum_{j=L_{i+1}}^{L_{n+1}} G_j}{2}}}{(1-2\beta)\bar{f}_v} + \frac{3(c_2 m - \sum_{j=L_{i+1}}^{L_{n+1}} G_j)_+}{m^4}\right).
$$

We now take expectation over G_j's gives us

$$
\begin{aligned}
\mathbb{E}\left[F\left(\left\{D_{\ell',\ell}\right\}_{\ell'\le\ell}\right)\middle|\{L_i\}_{i=1}^n\right] \le \Delta_0 + \sum_{i\in[n]} (L_{i+1}-L_i)\Bigg(&\frac{6}{(1-2\beta)\bar{f}_v}\left(\frac{1}{1+(1-2\beta)}\right)^{\frac{2(L_{n+1}-L_{i+1})}{\bar{f}_v}}\\
&+ \frac{3(c_2 m - \frac{L_{n+1}-L_{i+1}}{\bar{f}_v})_+}{m^4}\Bigg).
\end{aligned}
$$

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

$$\begin{aligned}
\mathbb{E}\left[R_\ell^{\text{conf}}\right] &= \mathbb{E}\left[F\left(\left\{D_{\ell',\ell}\right\}_{\ell'\le\ell}\right)\right] \\
&\le \Delta_0 + \frac{1}{1-\beta}\sum_{i\in[n]}\left(\frac{6}{(1-2\beta)\bar{f}_v}\Big(\frac{1}{1+(1-2\beta)}\Big)^{\frac{(n-i)}{\bar{f}_v}} + \frac{(c_2 m - \frac{(n-i)}{\bar{f}_v})_+}{m^4}\right) \\
&\le \Delta_0 + \frac{1}{1-\beta}\sum_{i=0}^{\infty}\Big(\frac{6}{(1-2\beta)\bar{f}_v}\Big(\frac{1}{1+(1-2\beta)}\Big)^{\frac{i}{\bar{f}_v}} + \frac{(c_2 m - \frac{i}{\bar{f}_v})_+}{m^4}\Big) \\
&\le \frac{2}{(1-2\beta)} r_{\min} + 2\Big(\frac{6}{(1-2\beta)^2\bar{f}_v} + \frac{128}{(1-2\beta)^6\bar{f}_v m^2}\Big) \\
&\le \frac{13}{(1-2\beta)} r_{\min} + \frac{256}{(1-2\beta)^6\bar{f}_v m^2} \\
&\le \frac{2808}{(1-2\beta)^3\bar{f}_v}\log\frac{50}{(1-2\beta)} + \frac{256}{(1-2\beta)^6\bar{f}_v m^2}.
\end{aligned}$$

□

E Fast confirmation for honest transactions: proof of Theorem 4

This section uses ideas from the proof of Lemma 14. Let the transaction tx enters the system[11] in round r and let ℓ be the last level on the proposer blocktree which has proposer blocks at round r. Define

$$\ell^* := \max\left(\tilde{\ell} \le \ell \;\; s.t \quad \text{the honest users mined the first proposer block on level } \tilde{\ell}\right)$$

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

$$A_{r_f} = \left\{Y^p[r^*, r_f - \Delta_0] - Z^p[r^*, r_f] > 0\right\}. \tag{72}$$

Lemma 25. *If event A_{r_f} occurs, then the transactions tx is included in a block b which is proposed in round $r(b) \le r_f - \Delta_0$ and confirmed as a leader block by round r_f.*

[11] As a part of a transaction block.

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 $\tilde{l} \geq \ell^*$ 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 23, 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. □

Let us define the following random variable:

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

Lemma 26.

$$\mathbb{E}[R_f - r] \leq \frac{24(1-\beta)r_{\min}}{(1-2\beta)^2} \leq \frac{2592}{(1-2\beta)^3 \bar{f}_v} \log \frac{50}{(1-2\beta)}. \tag{73}$$

Proof. Consider the following random walk

$$W_{r_f} := Y^p[r^* + \Delta_0, r_f] - Z^p[r^*, r_f - \Delta_0]. \tag{74}$$

and a random variable $V \sim \mathsf{Bin}(\Delta_0, \bar{f}_v/2)$ 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

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

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

$$\mathbb{E}[R_f - r^*] = \Delta_0 + \frac{\Delta_0}{1-2\beta} = \frac{24(1-\beta)r_{\min}}{(1-2\beta)^2}. \tag{75}$$

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 (43) give us the required result. □

From Lemma 25 and 26, 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.

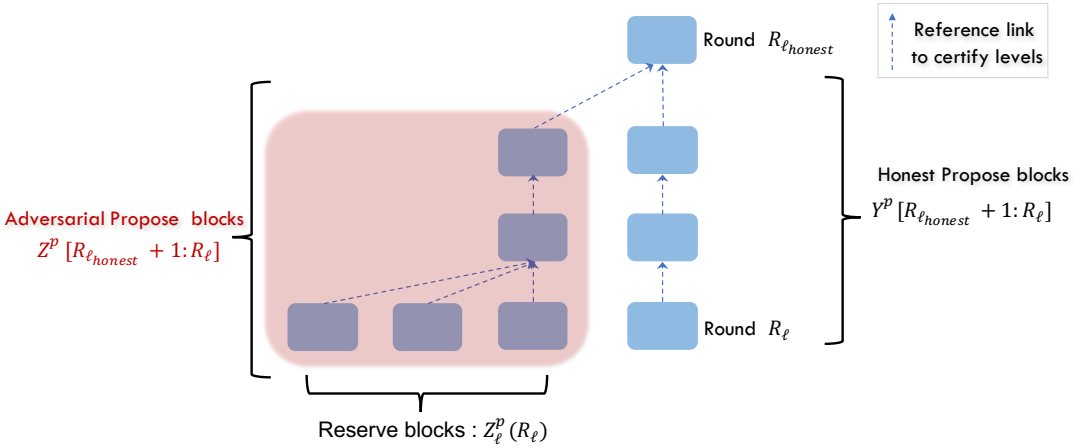


Fig. 22: Number of reserved blocks by the adversary on level ℓ in round $\mathcal{R}_\ell$.

F Others

F.1 Reserve proposer blocks by the adversary

Say the honest users mine the first proposer block at level ℓ in round R_ℓ. Let $W^p_\ell(R_\ell)$ denote that the number of hidden proposal blocks blocks on level ℓ by the adversary. In order to maximize $W^p_\ell(R_\ell)$, 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 22. The total number of reserve blocks is given by

$$W^p_\ell(R_\ell) = \max_{\ell_{honest}\le\ell} Z^p[R_{\ell_{honest}}+1, R_\ell] - Y^p[R_{\ell_{honest}}+1, R_\ell] + 1. \tag{76}$$

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

$$\begin{aligned}\mathbb{P}(W^p_\ell(R_\ell) > k) &= \mathbb{P}\Big(\max_{\ell_{honest}\le\ell} Z^p[R_{\ell_{honest}}+1, R_\ell] - Y^p[R_{\ell_{honest}}+1, R_\ell] \ge k\Big)\\ &= (2\beta)^k.\end{aligned}$$

Therefore $W^p_\ell(R_\ell) \sim \mathsf{Geometric}(1-2\beta)$.

F.2 Random walk proofs

Consider the following event from Equation (30)

$$\mathtt{E}_1\,[r-r', r] := \bigcap_{a,b\ge 0}\left\{Y[r-r'-a, r+b] - Z[r-r'-a, r+b] > \frac{(1-2\beta)k}{8}\right\},$$

for $r' = \frac{k}{2\bar{f}_v}$. 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 27. *If $W[r-r', r] > c_1 k$, for $c_2 < c_1$ we have*

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

Proof. Refer [2]. □

If the random walk is to the right of $c_1 k$ after r' steps, the above lemma calculates the probability of that the random walk remains to the right of $c_2 k$ in all future rounds.

Lemma 28. *If $W[r-r', r] > c_1 k$, for $c_2 < c_1$, then we have*

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

Proof. Refer [2]. □

The above lemma is mathematically characterizing the same event as Lemma 27.

Lemma 29. *If $W[r-r', r] > c_1 k$,then for $c_3 < c_1$, then we have*

$$\begin{aligned}\mathbb{P}\big(W[r-r'-b, r+a] \geq c_3 k \;\forall\; a > 0\big) &\geq 1 - 2(2\beta)^{(c_1-c_3)k/2} \\ &= 1 - 2e^{\log(2\beta)(c_1-c_3)k/2} \\ &\overset{(a)}{\geq} 1 - 2e^{-(1-2\beta)(c_1-c_3)k/2}.\end{aligned}$$

Proof. Using $c_2 = (c_1 - c_3)/2$ in the above two Lemmas 27 and 28, we get the required result. The inequality (a) uses $\log 2\beta < 2\beta - 1$ for $\beta > 0$. □