

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. Transaction throughput, confirmation latency and confirmation reliability are fundamental performance measures of any blockchain system in addition to its security. In a decentralized setting, these measures are limited by two underlying physical network attributes: communication capacity and speed-of-light propagation delay. 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.

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** [21]. Honest nodes mine blocks on top of each other by solving Proof-of-Work (PoW) cryptographic puzzles and by following a longest chain protocol they can come to a consensus on a transaction ledger that is hard to be reversed by an adversary. Solving the puzzle effectively involves randomly trying a hash inequality until success. Since Bitcoin’s invention, much work has been done on the analysis and design of blockchain protocols; 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. Our focus is on Proof-of-Work payment systems, although we believe similar design principles have applications to Proof-of-Stake as well as smart contracts systems.

^{*} Email: vbagaria@stanford.edu, ksreeram@uw.edu, dntse@stanford.edu, gfanti@andrew.cmu.edu, pramodv@illinois.edu

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.)

For example, Bitcoin is secure up to the adversary having 50% of the hashing power of the network nodes ($\beta = 0.5$), has throughput λ of the order of several transactions per seconds and confirmation latency of the order of tens of minutes to hours. In Bitcoin, there is also 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 [21] 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 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 physical 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 while the confirmation latency τ is at least D . What is less obvious is that these two parameters also limit the achievable confirmation *reliability*. 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 transaction during this period; i.e. votes are communicated at rate C/B and $C/B\tau$ votes are accumulated over time 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. In particular, if the desired confirmation latency τ is of the order of the propagation delay D , then we have:

$$\varepsilon = \exp\{-O(CD)\}, \quad (1)$$

i.e.

$$\log \frac{1}{\varepsilon} = O(CD).$$

The quantity CD is analogous to the key notion of *bandwidth-delay product* in networking (see eg. [14]); it is the number of “in-flight” transactions in the network.

To connect this to existing blockchain systems, 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, and the bandwidth-delay product is $CD = 5000$, giving an extremely small limit on the confirmation error probability (c.f. (1)). 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 an error probability $\varepsilon = 10^{-3}$. Ethereum has $\lambda \approx 15$ transactions per second and $\tau \approx 3$ minutes for a comparable error probability [5].

1.3 Main contribution

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

1. **security:** Prism is secure up to an adversary power of 50%, i.e. for any $\beta < 0.5$ and for any adversarial action, it can achieve an eventual total ordering of the transactions, with consistency and liveness guarantees.

2. **throughput:** For any adversarial action, Prism can achieve a throughput

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

3. **latency:** For any $\beta < 0.5$ and for any adversarial action, Prism can achieve an expected latency

$$\mathbb{E}[\tau] < \frac{16D}{(1 - 2\beta)^3} \quad \text{seconds}$$

on confirming a *list* of possible transaction ledgers such that with probability $\exp\{-\Omega(CD)\}$ one of the ledgers will be part of the eventual totally ordered ledger (Figure 1). Moreover, honest transactions (without double spends) will eventually appear in all ledgers in the list and can be confirmed with expected latency

$$\mathbb{E}[\tau] < \frac{22D}{(1 - 2\beta)^3} \quad \text{seconds.}$$

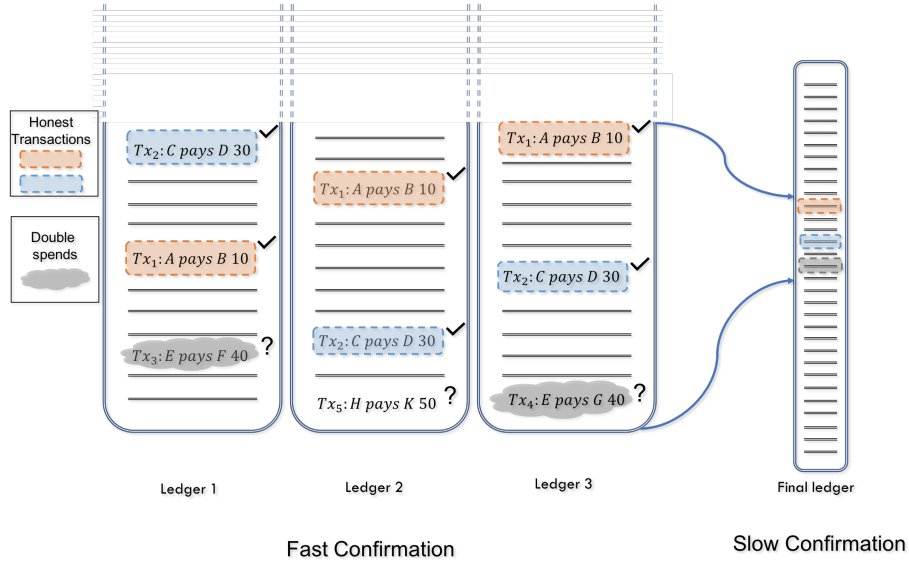


Fig. 1. List ledger confirmation. An example where one can fast-confirm that the final ledger is one of three possibilities. Honest transactions that appear in all three ledgers can be fast-confirmed. Double spends cannot appear in all ledgers and are therefore not fast-confirmed, although one of them will be slow-confirmed.

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.
- For honest transactions, **Prism** *simultaneously* achieves a latency proportional to the speed-of-light propagation delay and the smallest possible confirmation error probability, exponentially small in the bandwidth-delay product CD . Thus, there is essentially *no* tradeoff between confirmation latency and confirmation reliability for honest transactions. Put another way, for any desired security parameter $\log \frac{1}{\varepsilon} \ll CD$, the confirmation latency for honest transactions is *independent* of $\log 1/\varepsilon$ (Figure 2).
- For a total ordering of all transactions (including double spends), on the other hand, there is a tradeoff between latency and the security parameter, similar to that of Bitcoin: i.e. latency is proportional to $\log 1/\varepsilon$ (Figure 2).

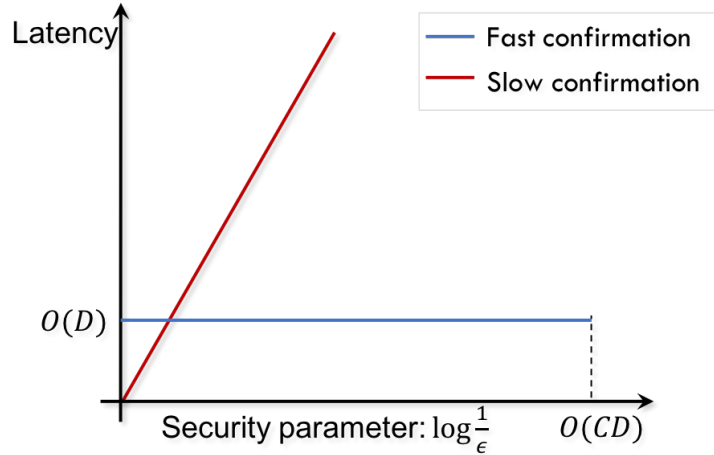


Fig. 2. Latency versus security parameter for fast and slow confirmation. For fast confirmation, latency is independent of security parameter value up to order CD . For slow confirmation, latency is proportional to security parameter.

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 [34]. Thus, Bitcoin's performance is security-limited, not communication-limited, and far away from the physical limits.

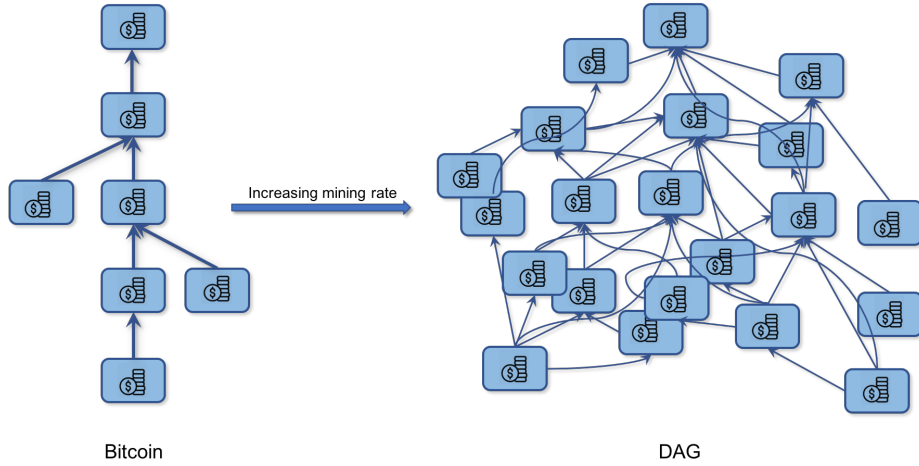


Fig. 3. The DAG approach to increasing the mining rate.

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 3). Examples of such works are GHOST [33], Inclusive [18], Spectre [32], Phantom[31] and Conflux [19]. 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.

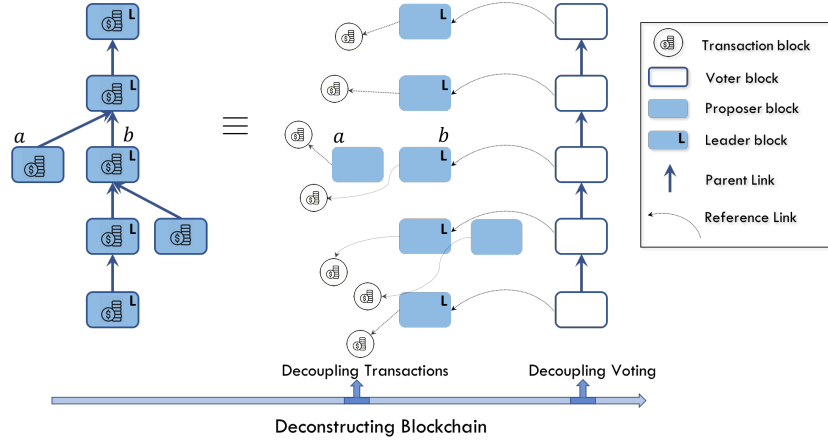


Fig. 4. Deconstructing the blockchain into transaction blocks, partially ordered proposal blocks arranged by level, and voter blocks organized in a voter tree. The main chain is selected through voter blocks, which vote among the proposal blocks at each level to select a leader block. For example, at level 3, block *b* is elected the leader over block *a*.

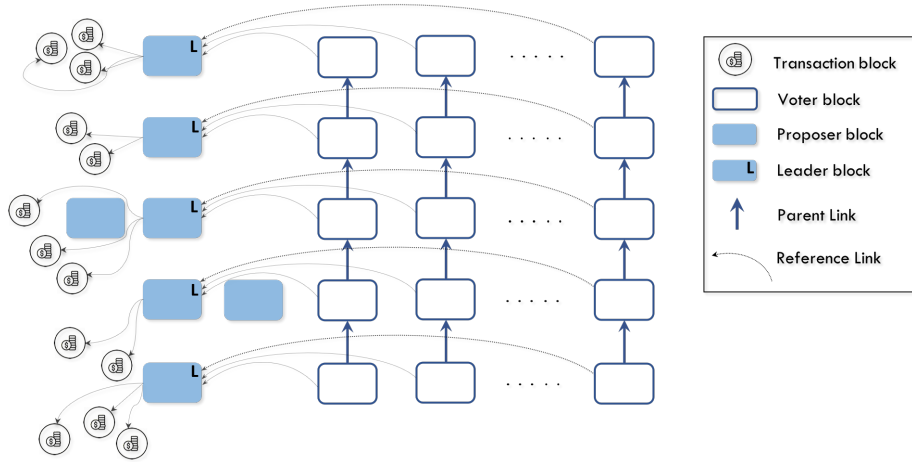


Fig. 5. Prism. Throughput, latency and reliability are scaled to the physical limits by increasing the number of transaction blocks and the number of parallel voting chains per proposal block.

In this work, we take a different approach. We start by *deconstructing* the basic blockchain structure into its atomic functionalities (Figure 4). 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. 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 block votes among the proposer blocks at the same level to select a leader block among them. The elected leader blocks can then bring in the transactions to form the final ledger. The 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 5). 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 [12], which is also used in **Fruitchains** [25] 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 than 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 [21] and is also implicit in the formal security analysis of **Bitcoin** in [12] (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. The averaging is achieved over time, and the price to pay is long latency, the latency increasing 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 casted 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 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 at extremely 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 that the final leader block at a given level must be one of a list of proposer blocks. 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. Together with liveness of honest transactions, we show that this “list decoding” capability is sufficient for fast confirmation of all honest transactions¹. If one block stands out in terms of number of votes cast on the different blocks, then the list narrows to this single block, which can then be declared as the leader block. In the worst case, when the votes are tied between two or more proposer blocks (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 a global consensus of a unique leader block at a given level, and 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 [12] in a synchronous

¹ 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.

round-by-round model and subsequently extended in [24] to an asynchronous model with an upper bound on block network delay. In particular, [12] 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 consistency 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 [24]). The technically most challenging part of the analysis is on fast latency confirmation, where we show that: 1) common prefix property of the vote trees guarantee *vote consistency*, so that a large fraction of the votes will not be reversed; 2) chain quality of the main chains of the vote trees guarantee *vote liveness*, so that a large fraction of the vote trees will have honest votes on the proposer blocks at each level.

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 [12] 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 [33], Inclusive [18], Spectre [32], Phantom [31], and Conflux [19]. The earliest of these schemes, GHOST, handles forking through a fork-choice rule that builds on the heaviest subtree [33]. 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 [33] that **GHOST** is secured against a 50% purely private attack, it turns out that **GHOST** is vulnerable to a public-private balancing attack [22], 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 [18], 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 DAG idea to achieve low confirmation time by interpreting the reference links as votes to compare between pairs of blocks [32]. 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 [31]. 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 [19] 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 [19]. 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 [9] 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 [25] 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 [10,29,23]. 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 [26]. 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 [27]. *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 $f p_n$ 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. As a consequence, the honest nodes mine blocks with Poisson process rate $\sum_{v \in \mathcal{H}} f p_v = (1 - \beta)f$ and the adversarial nodes mine blocks with Poisson process rate $\sum_{v \in \mathcal{N}/\mathcal{H}} f p_v = \beta f$. Without loss of generality we can assume a single adversarial node with $1 - \beta$ 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 [12]. Each round in this model corresponds to Δ seconds in the continuous-time model above. In the r th 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 r th round is a uniquely honest round, i.e. a round in which only one block

is mined by the honest nodes. Note that $Y[r]$ has a Bernoulli distribution with parameter $(1 - \beta)f\Delta e^{-(1-\beta)f\Delta}$.

The location of the $H[r]$ honest blocks in the block data structure after the r th round is protocol-dependent. In Bitcoin, for example, all honest blocks are 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 [12], 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:

$$\Delta = \frac{B}{C} + D, \quad (2)$$

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 [33], based on empirical data in [8], as well in [28]. 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:

$$fB < C. \quad (3)$$

For a given block size, (3) 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.

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* $\beta < 0.5$.

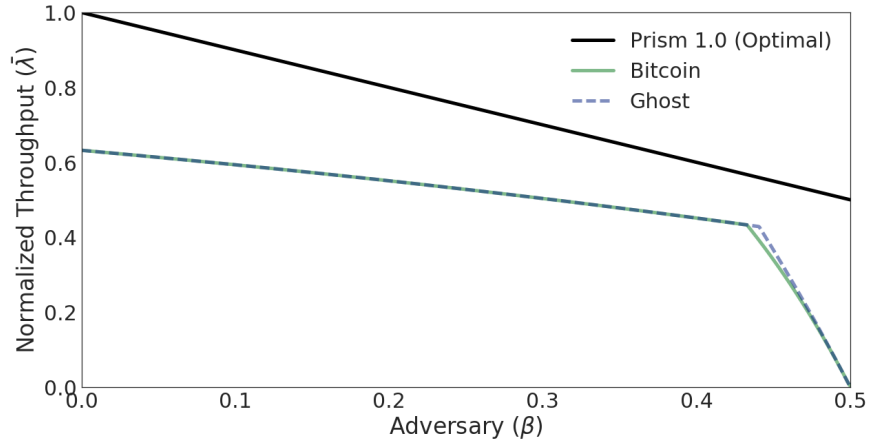


Fig. 6. Throughput efficiency versus β tradeoff of baseline protocols and Prism 1.0 . The tradeoffs for the baseline protocols are upper bounds, while that for Prism 1.0 is exact.

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 [21], and formally by [12] in the synchronous model, followed by the analysis of [24] in the asynchronous model. These works and others (e.g., [33, 16]) 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.

$$1 - e^{-(1-\beta)f\Delta} \text{ blocks per round,} \quad (4)$$

Notice that (4) 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, [12] shows that the main chain must grow faster in expectation than any adversarial chain, which can grow at rates up to $\beta f \Delta$ in expectation. Hence we have the following (necessary) condition for security:

$$1 - e^{-(1-\beta)f\Delta} > \beta f \Delta. \quad (5)$$

Equation (5) 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:

$$1 - e^{-(1-\beta)\bar{f}} = \beta \bar{f},$$

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

$$\lambda_{\text{BTC}} \leq \beta \bar{f}_{\text{BTC}}(\beta) B / \Delta. \quad (6)$$

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. (3)). Substituting into (4) 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:

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

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 $\beta \rightarrow 0.5$. This is a manifestation of the (well-known) fact that to get a

high degree of security, i.e. to tolerate β close to 0.5, the mining rate of Bitcoin must be small, resulting in a low throughput. 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 [33] 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.

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

(cf. (4)). 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 (8) for Bitcoin:

$$\bar{\lambda}_{\text{GHOST}} \leq 1 - e^{\beta-1}. \quad (8)$$

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:

$$\mathbb{E}[|H_1[r] - H_2[r]|] > \beta f \Delta, \quad (9)$$

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

$$\bar{\lambda}_{\text{GHOST}} \leq \beta \bar{f}_{\text{GHOST}}(\beta), \quad (10)$$

where $\bar{f}_{\text{GHOST}}(\beta)$ is the value of $f\Delta$ such that (9) is satisfied with equality instead of inequality.

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

$$\bar{\lambda}_{\text{GHOST}} \leq \min \{ \beta \bar{f}_{\text{GHOST}}(\beta), 1 - e^{\beta-1} \}. \quad (11)$$

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 [33], 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 5), 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.

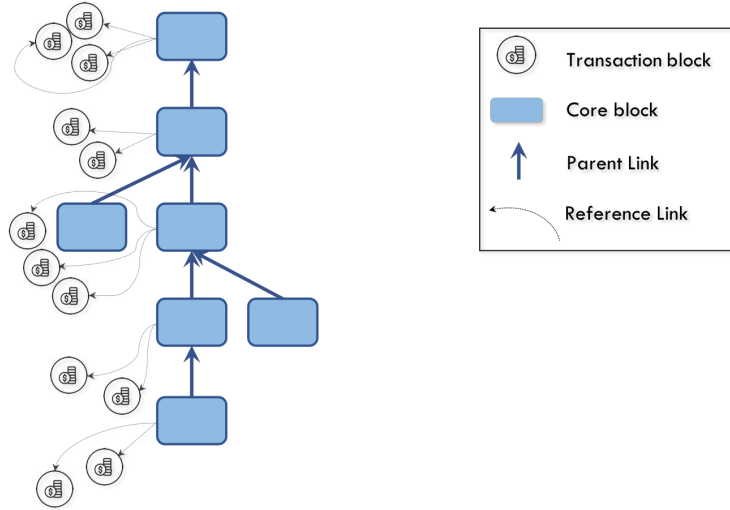


Fig. 7. Prism 1.0. Decoupling the transaction blocks from the core blocks in the Bitcoin blockchain.

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 fact.
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.

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. Given these blocks, the ledger is produced by including transactions from the referred transaction blocks as well as the transactions in each core block in order in the main-chain. If a transaction output is spent in multiple transactions, only the first transaction in the ledger is counted.

In the context of the round-by-round synchronous model, the $H[r]$ honest blocks mined in the r th round are now split into $H^c[r] \sim \text{Pois}((1-\beta)f_c\Delta)$ honest core blocks and $H^t[r] \sim \text{Pois}((1-\beta)f_t\Delta)$ honest transaction blocks, where $f_c + f_t = f$. Similarly, the $Z[r]$ adversarial blocks mined in the r th round are split into $Z^c[r] \sim \text{Pois}(\beta f_c\Delta)$ adversarial core blocks and $Z^t[r] \sim \text{Pois}(\beta f_t\Delta)$ 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, [12] gives one such sufficient condition, obtained by requiring that the rate of arrival of uniquely honest rounds exceeds the rate of work of the adversary:

$$f_c\Delta < \frac{1}{1-\beta} \ln \frac{1-\beta}{\beta} \quad (12)$$

Under this condition, [12] showed that the longest chain satisfies the common-prefix property as well as has positive chain quality. Similar to the argument in

Conflux, 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 at 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 (4), 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. (8)), i.e.

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

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 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

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

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

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

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,

$$\tau_p = \frac{qB}{\lambda} \text{ seconds.} \quad (14)$$

On the other hand, from (13), we see that the throughput λ , at the capacity limit, is given by

$$\lambda = q \left[1 - e^{-(1-\beta)C\Delta/(Bq)} \right] \frac{B}{\Delta} \text{ transactions per second} \quad (15)$$

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 (14) and (13) and obtain:

$$\bar{\lambda} = \frac{1 - \beta}{\bar{\tau}_p \ln \left(\frac{1}{1 - \frac{1}{\bar{\tau}_p}} \right)} \quad 1 < \bar{\tau}_p < \infty, \quad (16)$$

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 , (16) 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, and 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.

4.6 Discussions

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

1. Unlike Conflux, which tries to separate links into two types: main-chain links and reference links, in Prism 1.0 we separate **blocks** into two types: core blocks, which go into the core blockchain and transaction blocks, which are referenced by the core blocks. As a result, the Conflux's security is limited by GHOST, and because Conflux is not done in conjunction with transaction scheduling (unlike Prism 1.0), its throughput- β tradeoff is exactly the same as that of GHOST shown in Figure 6.

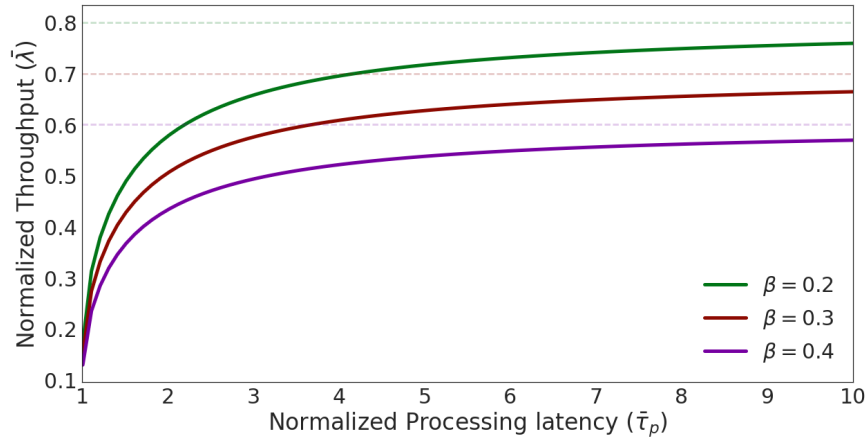


Fig. 8. Tradeoff between $\bar{\lambda}$ and $\bar{\tau}_p$ for different values of β . Throughput is normalized as a fraction of the network capacity, and the processing latency is normalized as a multiple of the speed-of-light propagation delay.

2. Prism 1.0 can be viewed as similar to BitcoinNG 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 [25] 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 fruit-set is stored directly inside the block in Fruitchains but we only store references (hash-pointers) to transaction blocks in the core blocks in order to optimize for throughput. 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 [12, 25] 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 [17]. 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 somewhat reminiscent of an approach called Subchains [28] or weak blocks [?, ?]. 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 [18] 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 [12]), 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 [12], 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 and honest transactions 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. More precisely, it can be shown that at a mining rate of f , it takes on average (Corollary F1):

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

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 (5) 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

$$\mathbb{E}[\tau] > \frac{\Delta(1 - \beta)^2}{(1 - 2\beta)^3} \log \frac{1}{\varepsilon} \quad \text{seconds.} \quad (17)$$

This lower bound is far from the physical limit D , the speed-of-limit propagation delay, 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 (6) 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 lower bound is proportional to $\log \frac{1}{\varepsilon}$, and hence is dependent on the confirmation reliability requirement.

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 make the confirmation latency ε -independent.

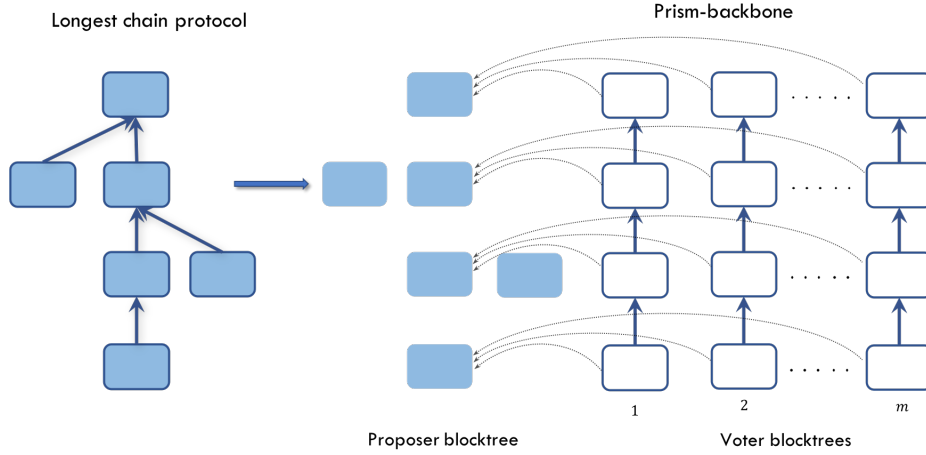


Fig. 9. Prism : Separating proposer and voter roles.

5.2 Prism

5.2.1 Prism: Backbone

We begin by describing Prism's backbone, or blockchain architecture, in which we will explain how blocks relate to each other, and which blocks will find a place in the final ledger. We will 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:

- *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 the level of the proposer block.
- *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

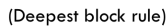


Fig. 10. Prism: Honest users mine a proposer block p_{new} at a level one deeper than the current deepest level—in this example, p_{new} has level 5. The voter block v_{new} is mined on the longest chain. It votes (via reference links) to all proposer block on level $\{3, 4\}$ because its ancestors have 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.

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 caste 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 for multiple blocks at the same proposer 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 (current) *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 $\bar{f}_v \triangleq f_v D$ on each voting tree is chosen such that each voting tree is secure under the longest chain rule and according to (12) it should satisfy

$$\bar{f}_v < \frac{1}{1-\beta} \ln \frac{1-\beta}{\beta}.$$

For notational simplicity, we choose $\bar{f}_v = \frac{(1-2\beta)}{2}$, which is less than $\frac{1}{1-\beta} \ln \frac{1-\beta}{\beta}$ and set $f_p = f_v$. This implies that

$$m = \frac{2CD}{1-2\beta} - 1, \quad (18)$$

i.e. the number of voting trees is proportional to the bandwidth-delay product CD . This number is expected to be very large, which is a key advantage of our protocol.

5.2.2 Prism: Transaction Structure

Having presented the Prism backbone protocol, we now proceed to describe how the 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:
 - $\text{Hash} < \alpha_p \Rightarrow \text{Block is a proposer.}$

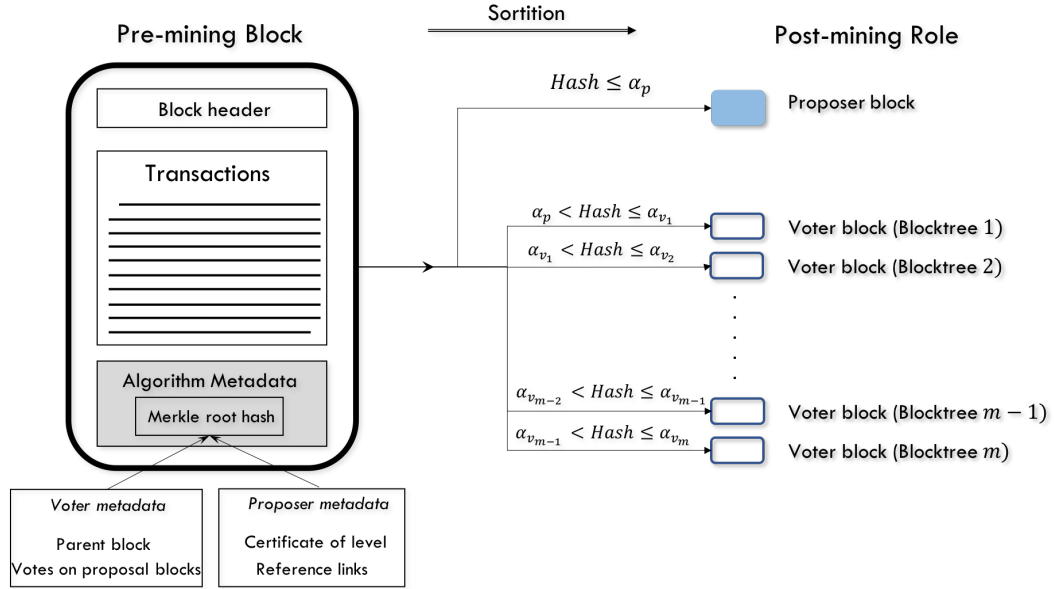


Fig. 11. Summary of the block structure and the sortition procedure.

- $(i-1)\alpha_v + \alpha_p < \text{Hash} < i\alpha_v + \alpha_p \Rightarrow$ 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 ℓ_{\max} is the highest level that the node knows of, and ℓ_{\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, $b_i, i = 1, 2, \dots, m$. 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 refer to all proposal and voter blocks, which are leaves 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, \dots, p_ℓ , the ledger is defined as follows (our ledger construction procedure is similar to the one in Conflux [19]). 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-spend 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 put in valid transactions with respect to the current ledger.

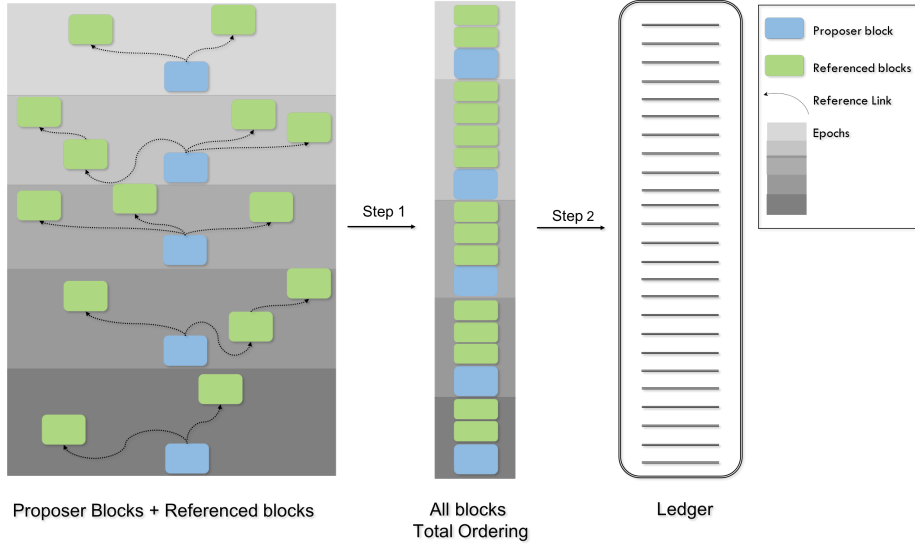


Fig. 12. Prism: Generating the ledger. The proposer blocks for a given proposer block sequence are highlighted in blue, and the referenced blocks are shown in green. Each shade of grey corresponds to an epoch. In step 1, all the blocks are incorporated and . In step 2, they are expanded out to give a 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 $i = 1, 2, \dots, m$. 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 also define $Y_i[r]$ ($Y^p[r]$), which is 1 if $H_i[r] = 1$ ($H^p[r] = 1$) and zero otherwise. We denote $Y_i[r_1 : r_2] := \sum_{r=r_1+1}^{r_2} Y_i[r]$, similarly for Z_i and H_i .

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 vote of the honest nodes go (but it cannot remove the vote, nor change 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 [12]), 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 will show that **Prism** can achieve total transaction ordering for any $\beta < 0.5$, but with an ε -dependent latency. Following the framework of [12], we will 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. 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 block 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.,

$$p_\ell^*(r) \triangleq \operatorname{argmax}_{p \in \mathcal{P}_\ell(r)} V_p(r),$$

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

A *proposer sequence* up to level ℓ at round r is given by $[p_1, p_2, \dots, p_\ell]$, where $p_j \in \mathcal{P}_j(r)$. The *leader sequence* up to level ℓ at round r , denoted by $\text{LedSeq}_\ell(r)$, is a proposer sequence with $p_j = p_j^*(r)$, in other words $\text{LedSeq}_\ell(r) \triangleq [p_1^*(r), p_2^*(r), \dots, p_\ell^*(r)]$. The leader sequence at the end of round r_{\max} , the end of the horizon, is the *final leader sequence*, $\text{LedSeq}_\ell(r_{\max})$.

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 become 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*

$$\text{LedSeq}_\ell(r) = \text{LedSeq}_\ell(r_{\max}) \quad \forall r \geq R_\ell + r(\varepsilon) \quad (19)$$

with probability $1 - \varepsilon$, where $r(\varepsilon) = \frac{12}{f_v(1-2\beta)^2} \log(\frac{3m}{\varepsilon})$, and R_ℓ is the round in which the first proposer block on level ℓ was mined. Thus, in terms of physical parameters, the latency is bounded by:

$$\frac{24D}{(1-2\beta)^3} \left[\log \frac{1}{\varepsilon} + \log \left(\frac{6CD}{1-2\beta} - 3 \right) \right] \quad \text{seconds.}$$

Proof. Lemma 20 in the Appendix shows that all the votes on proposer blocks up to level ℓ are permanent by round $R_\ell + r(\varepsilon)$ with probability $1 - \varepsilon$. Thus we have

$$p_{\ell'}^*(r) = p_{\ell'}^*(r_{\max}) \quad \forall \ell' \leq \ell \text{ and } r \geq R_\ell + r(\varepsilon) \quad (20)$$

w.p $1 - \varepsilon$. Since leader block at all levels $\leq \ell$ satisfy Equation (20), the main result of the theorem follows from the definition of the leader sequence. \square \square

Theorem 1 is the analog of Theorem 15 of [12], 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 supported by parent-link relationships between the leader blocks. Rather, each leader block is individually supported 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. Ledger liveness, on the other hand, is guaranteed by the chain quality property. This is Theorem 16 of [12]. The analogous result for Prism is given in the following theorem.

Theorem 2 (Leader sequence quality). *Suppose $\beta < 0.5$. At least $\frac{1-2\beta}{3}$ fraction of blocks in the final leader sequence, $\text{LedSeq}_{L_{\max}}(r_{\max})$, are mined by honest users with probability $1 - 2e^{-\frac{r_{\max}f_v(1-2\beta)^2}{200}}$. Here L_{\max} is the length of the final leader sequence.*

Proof. The total number of unique proposer blocks mined by the honest users is $Y^p[0 : r_{\max}]$ and the total number of proposer blocks mined by the adversary is $Z^p[0 : r_{\max}]$. Lemma 17 in the Appendix proves the following inequalities:

$$Y^p[0 : r_{\max}] > \frac{f_v r_{\max}(3 + 2\beta)}{8} \quad (21)$$

$$Z^p[0 : r_{\max}] < \frac{f_v r_{\max}(1 + 6\beta)}{8} \quad (22)$$

with probability $1 - 2e^{-\frac{r_{\max}f_v(1-2\beta)^2}{200}}$. In the worst case, all the adversary's proposer blocks will become leader blocks and the fraction of honest leader blocks is

$$\begin{aligned} \frac{Y^p[0 : r_{\max}] - Z^p[0 : r_{\max}]}{Y^p[0 : r_{\max}]} &> \frac{2 - 4\beta}{3 + 2\beta} \\ &> \frac{1 - 2\beta}{3}. \end{aligned}$$

The factor $1 - 2\beta$ appears in the chain quality property in bitcoin backbone paper [12] (Theorem 16). This factor also comes up in selfish mining results [10,29,23]. \square

Together, Theorem 1 and Theorem 2 guarantee that Prism achieves a total ordering of all transactions, with consistency and liveness properties, 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: a confirmation latency that is ε -independent. 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 some feel why and what we can confirm with latency much shorter than Bitcoin latencies.

Suppose $CD = 5000$, $D = 0.2$ seconds and $\beta = 0.4$, so we have $m = 2CD/(1-2\beta) = 50,000$ votes at each level and votes are mined at rate $1 - e^{-\bar{f}_v} = 1 - e^{-(1-2\beta)/2} \approx 0.1$ 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 the average, $\bar{f}_v m = 5000$ 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:

$$m(1 - \bar{f}_v)^r,$$

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

$$m(1 - \bar{f}_v)^r < \frac{m - m(1 - \bar{f}_v)^r}{2},$$

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 proposer block; in fact, this fraction may be greater than β due to selfish mining [10,29,23]. Thus, the number of outstanding votes calculated above may be on under-estimation. However, we do know that the longest chain quality is non-zero (Theorem 16 of [12]). 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 there the votes get deeper in their respective chain. 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 number 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 $[\underline{V}_n(r), \bar{V}_n(r)]$ for the votes on proposer block p_n bounds the maximum number of votes the block can lose or gain from 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.

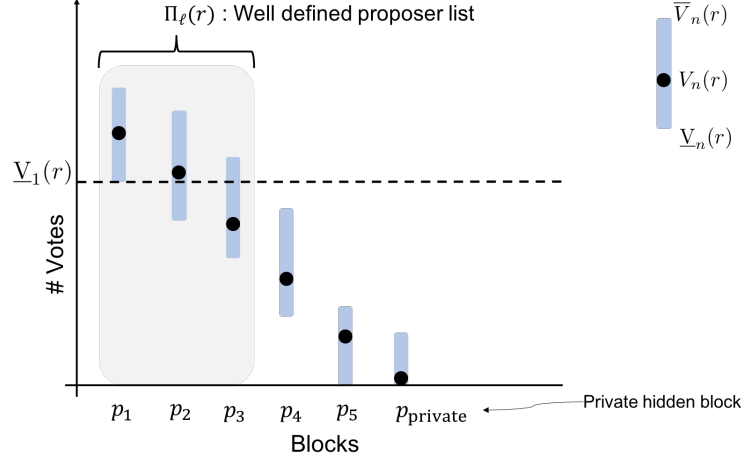


Fig. 13. In the above example, Public proposer block p_1 has the largest lower confidence bound, which is larger than the upper confidence bound of the private block. So list confirmation is possible and the list confirmed is $\Pi_\ell(r) = \{p_1, p_2, p_3\}$.

More formally: Let $\mathcal{P}_\ell(r) = \{p_1, p_2, \dots\}$ be the set of proposer blocks at level ℓ at round r . Let θ be the fraction of 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 number of votes at depth d or greater for a proposer blocks in the subset $\mathcal{P}_\ell(r) - \{p_n\}$. Define:

$$\delta_d \triangleq \left(\frac{1}{4\bar{f}_v d} \vee \frac{1 - 2\beta}{24 \log m} \right)$$

and

$$\underline{V}_n(r) \triangleq \max_{d \geq 0} (V_n^d(r) - \delta_d m)^+,$$

$$\bar{V}_n(r) \triangleq V_n(r) + \left(V_{-n}(r) - \max_{d \geq 0} (V_{-n}^d(r) - \delta_d m)^+ \right) + \theta,$$

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

$$\bar{V}_{\text{private}}(r) \triangleq \min_{d \geq 0} \delta_d m + \theta.$$

Proposer list confirmation policy: If

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

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

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

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.

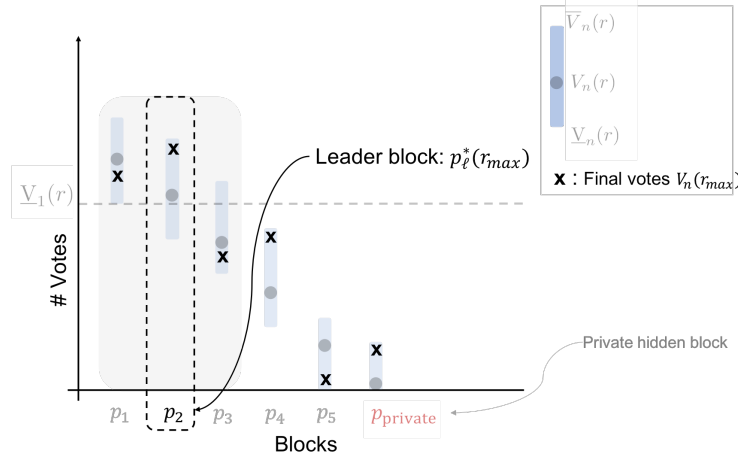


Fig. 14. A possible scenario after the final round.

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 $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$, we can confirm proposer lists $\Pi_1(r), \Pi_2(r), \dots, \Pi_\ell(r)$ for all rounds $r \geq R_\ell + R_\ell^{\text{conf}}$, where

$$\mathbb{E}[R_\ell^{\text{conf}}] \leq \frac{8}{(1-2\beta)^2 f_v} + \frac{288}{(1-2\beta)^3 m}.$$

Moreover, w.p $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$,

$$\text{LedSeq}_\ell(r_{\max}) \in \Pi_1(r) \times \Pi_2(r) \times \dots \times \Pi_\ell(r), \quad \forall r \geq R_\ell + R_\ell^{\text{conf}}.$$

Thus, for $CD \gg 1$, the average latency is bounded by:

$$\frac{16D}{(1-2\beta)^3} \quad \text{seconds.}$$

Proof. Lemma 10 in the Appendix shows that in round $R_\ell + R_\ell^{\text{conf}}$, proposer list for all levels up to level ℓ can be confirmed where $\mathbb{E}[R_\ell^{\text{conf}}] \leq \frac{16}{(1-2\beta)^3} + \frac{288}{(1-2\beta)^3 m}$ for $f_v = \frac{1-2\beta}{2}$. Applying this in Lemma 8 in the Appendix, we get

$$\text{LedSeq}_\ell(r_{\max}) \in \mathcal{L}_\ell(r) \quad \forall r \geq R_\ell + R_\ell^{\text{conf}}.$$

□

□

5.5.3 Fast confirmation of honest transactions

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 1.) 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. The latter is given by the following theorem.

Theorem 4. *Assume $\beta < 0.5$. If a honest transaction without double spends is mined in a transaction block in round r , then wp $1 - r_{\max}^2 e^{-\frac{m}{48 \log m}}$ it will appear in all of the ledgers corresponding to proposer block sequences after an expected latency no more than*

$$\frac{3}{(1 - 2\beta)^2 \bar{f}_v} \quad \text{rounds}$$

i.e.

$$\frac{6D}{(1 - 2\beta)^3} \quad \text{seconds.}$$

Proof. Theorem 2 shows that the leader block sequence quality is positive using the fact that honest users mine more proposer blocks than the adversary. This ensures liveness for honest transactions. In order to give latency guarantees, we need to prove that these honest leader blocks are frequent. We will consider a *Good event* and show that this event occurs in finite number of expected rounds.

Good event: After round r , consider the event E when 1) only one proposer block p is mined by the honest users at level ℓ at round R_ℓ . 2) No further proposal blocks are mined at level ℓ in the next k_{good} rounds by the adversary and by round $R_\ell + k_{\text{good}}$ block p *permanently* obtains more than $1/2$ the fraction of votes wp $1 - r_{\max}^2 e^{-\frac{m}{48 \log m}}$.

If this event occurs, $\Pi_\ell(r') = \{p\} \forall r' > R_\ell + k_{\text{good}}$ wp $1 - r_{\max}^2 e^{-\frac{m}{48 \log m}}$, and tx will be present in all the ledgers in any future list of proposal sequences. We show that the number of rounds $R_\ell - r$ in expectation is at most $\frac{3}{(1-2\beta)^2 \bar{f}_v}$ in expectation in Appendix D. □

Combining Theorem 3 and 4, the expected latency for confirming a honest transaction is bounded by

$$\frac{22D}{(1 - 2\beta)^3} \text{ seconds.}$$

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. It is easy 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 [25]. 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 [29].

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 [7]) 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 [20,15]) 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 Technology Research Foundation, the Army Research Office under grant W911NF-18-1-0332-(73198-NS), and the National Science Foundation under grants 1705007 and 1651236 for supporting their research program on blockchain technologies. We thank Applied Protocol Research Inc. for support and for providing a conducive environment that fostered this collaborative research. We also thank Andrew Miller for his 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. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. 2018.
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. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.
7. Alex de Vries. Bitcoin’s growing energy problem. *Joule*, 2(5):801–805, 2018.
8. C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.
9. Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoinng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.
10. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.
11. Lei Fan and Hong-Sheng Zhou. A scalable proof-of-stake blockchain in the open se ing. 2018.
12. 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.
13. Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.
14. 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.
15. 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.
16. Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. bloxroute: A scalable trustless blockchain distribution network whitepaper.
17. Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolsky, 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.
18. Yoad Lewenberg, Yonatan Sompolsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.
19. 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.
20. 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.
21. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

22. 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.
23. 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.
24. 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.
25. Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017.
26. 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.
27. 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.
28. Peter R Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.
29. Ayelet Sapirshtein, Yonatan Sompolsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
30. Zhan Shi. Galton–watson trees. In *Branching Random Walks*, pages 11–17. Springer, 2015.
31. Y Sompolsky and A Zohar. Phantom: A scalable blockdag protocol, 2018.
32. Yonatan Sompolsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.
33. Yonatan Sompolsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.
34. Statoshi. Bandwidth usage. <https://statoshi.info/dashboard/db/bandwidth-usage>.

A Appendix: Attack on GHOST

This attack is similar to the balancing attack in [22]. 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.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 , $H_1[r], H_2[r] \sim \text{Poiss}((1 - \beta)f\Delta/2)$ honest blocks are mined in subtree 1 (below b_1) and subtree 2 (below b_2) respectively.
5. *Attack Strategy:*
 - If $H_1[r] = H_2[r]$, then the adversary does nothing.
 - If say $H_1[r]$ is larger, then adversary releases $H_1[r] - H_2[r]$ blocks that it has mined in subtree 2 (either in private or just mined in this round). Vice versa for the case when $H_2[r]$ is larger. This (re)balances the weight of the two subtrees and the honest work is again split in the next round.
6. *Analysis:* The expected number of blocks the adversary needs to release in subtree 1 per round is $\mathbb{E}[(H_2[r] - H_1[r])^+]$. 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 Appendix: Latency Formal Proofs

As alluded to earlier, we show that in a constant number of rounds after the first proposer block at level ℓ appears, our protocol produces a *list* of proposer block sequences which contain the final leader sequence upto level ℓ with high probability³. In this subsection, the constants are of the order $O(\frac{1}{(1-2\beta)^3})$. The

² Say the adversary mines b_1 and the honest nodes mine b_2

³ The probability being $1 - r_{\max}^2 e^{-\frac{m(1-2\beta)}{48 \log m}}$.

proof broadly has four sub-parts. Each of these sub-parts proves that a certain class of events occurs in a constant number of rounds with high probability:

1. *Consistency of Votes*: We first show among all the votes embedded at a depth, a significant fraction of them will remain forever. Thus the voting pattern cannot be modified significantly after constant number of rounds.
2. *Liveness of Votes*: We then show that after a constant number of rounds from when the first proposer block at level ℓ showed up, a significant fraction of the voter trees will have votes on this level embedded with a constant depth.
3. *List confirmation policy*: We define a proposer block list confirmation policy, which specifies when we confirm a list of proposal blocks at a given level. After this list is confirmed, we show that the (final) leader block at this level will be from this confirmed proposer-list whp.
4. *Proposer-list confirmation latency*: Using the consistency and liveness of the votes results from the first and second point, we show that the proposer list at a given level is confirmed in constant number of rounds in expectation. We then further proceed to show that *all* the proposer lists upto that level are confirmed in constant number of rounds in expectation. This result forms the basis for Theorem 3.

In the next subsection we define a set of typical events which will be used to prove our results. After that, the later four subsections prove each of these above four points.

B.1 Typical events

Along the lines of events defined in the bitcoin backbone paper [12], we define the following events:

$$E_j[r: r+k] = \{Y_j[r: r+k] > Z_j[r: r+k] + \frac{1}{8}\bar{f}_v(1-2\beta)k\} \quad (23)$$

$$F_j[r: r+k] = \bigcap_{a,b \geq 0} E_j[r-a: r+k+b] \quad (24)$$

In the above expression, $E_j[r: r+k]$ is an event where the honest users mined $\frac{1}{8}\bar{f}_v(1-2\beta)k$ unique blocks more than the total blocks mined by adversary in the interval $[r: r+k]$. Next, $F_j[r: r+k]$ is an event where the honest users mine more than $\frac{1}{8}\bar{f}_v(1-2\beta)k$ blocks than the adversary in *all* the intervals containing the interval $[r: r+k]$.

Lemma 1. *For all r, k , $\mathbb{P}(F_j[r: r+k]) > 1 - 2e^{-\gamma k}$, where $\gamma \geq \frac{1}{12}f_v(1-2\beta)^2$.*

Proof. Appendix C.1. □

Theorem 5. *The typical event T defined below occurs with high probability $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$.*

$$T_{r,r+k} \triangleq \left\{ \frac{1}{m} \sum_{j=1}^m \mathbf{1}(F_j[r:r+k]) \geq 1 - \left(\frac{1}{8\bar{f}_v k} \vee \frac{1-2\beta}{24 \log m} \right) \right\}$$

$$T \triangleq \bigcap_{0 \leq r, k \leq L} T_{r,r+k}.$$

For $r_{\max} = O(e^{o(m)})$, T is a high probability event.

Proof. For a fixed r, k , $\mathbf{1}(F_j^c([r:r+k]))$ are identical and independent indicator random variables $\forall j \in [m]$ with probability at most $2e^{-\gamma k}$ (Lemma 1). Using Chernoff bound⁴, we have

$$\mathbb{P}\left\{ \frac{1}{m} \sum_{j=1}^m \mathbf{1}(F_j^c[r:r+k]) \geq (1+\delta)2e^{-\gamma k} \right\} \leq e^{-\frac{\delta^2}{\delta+2} 2e^{-\gamma k} m}$$

For $\delta + 1 = e^{\gamma k} \left(\frac{1}{16\bar{f}_v k} \vee \frac{1-2\beta}{48 \log m} \right)$, where \vee denotes the maximum, we get

$$\mathbb{P}\left\{ \frac{1}{m} \sum_{j=1}^m \mathbf{1}(F_j^c[r:r+k]) \geq \left(\frac{1}{8\bar{f}_v k} \vee \frac{1-2\beta}{24 \log m} \right) \right\} \leq e^{-\frac{m}{48 \log m}}.$$

Since r, k can take at most r_{\max}^2 different values, the event T intersection of $r_{\max}^2 T_{r,r+k}$ events, and hence T occurs wp greater than $1 - L^2 e^{-\frac{m}{48 \log m}}$. Therefore, for large values m and $r_{\max} = O(e^{o(m)})$, T is a high probability event. \square

B.2 Consistency of Votes

Along the lines of [12], we show that at any given round, a fraction of d -deep votes on the voter main chains are permanent with high probability. i.e, they will not be reverted in future.

Theorem 6 (Consistency). *At any round r , among all the d -deep blocks on m voter blocktrees, $1 - \left(\frac{1}{4\bar{f}_v d} \vee \frac{1-2\beta}{12 \log m} \right)$ fraction of these blocks are permanent wp $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$.*

Proof. First, Lemma 2 proves that most of the blocks d -deep blocks in round r are mined before round $r - \frac{d}{2\bar{f}_v}$. Next, Lemma 3 proves that most of the blocks mined before round $r - \frac{d}{2\bar{f}_v}$ are permanent.

Let b_j denote the d -deep voter block on the main chain of voter blocktree j . Consider the fixed round $r' = r - \frac{d}{2\bar{f}_v}$.

⁴ <http://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf>

Lemma 2. *The fraction of blocks $\{b_j\}_{j=1}^m$ mined before round r' is greater than $1 - \left(\frac{1}{8f_v d} \vee \frac{1-2\beta}{24 \log m}\right)$ with probability $1 - e^{-\frac{(1-2\beta)m}{48 \log m}}$.*

Proof. Refer to Appendix subsection C.2. \square

Lemma 3. *If the event $F_j[r' : r]$ occurs then block b_j permanently remains in the main chain of voter blocktree j .*

Proof. Refer to Appendix subsection C.3. \square

From Lemma 2, at least $1 - \left(\frac{1}{8f_v d} \vee \frac{1-2\beta}{24 \log m}\right)$ fraction of the voting blocktree, voter block b_j is mined before round r' . Theorem 5 shows that $F_j[r' : r]$ occurs in at least $1 - \left(\frac{1}{8f_v d} \vee \frac{1-2\beta}{24 \log m}\right)$ fraction of voter blocktrees with high probability. This combined with Lemma 3, proves that $1 - \left(\frac{1}{4f_v d} \vee \frac{1-2\beta}{12 \log m}\right)$ fraction of d deep voter blocks are permanent. \square \square

B.3 Liveness of Votes

After the first proposal of block(s) at level ℓ , we show that after a constant number of rounds, a large fraction of the voter trees will permanently vote on a proposal block at level ℓ .

Theorem 7 (Liveness). *The first proposer block(s) at level ℓ is proposed in round R_ℓ . At round $R_\ell + k$, $1 - \left(\frac{1}{8f_v k} \vee \frac{1-2\beta}{24 \log m}\right)$ fraction of voter blocktree will have a **permanent** vote on a proposer block at level ℓ and these votes will be at least αk -deep w.p. $1 - r_{max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$. Here $\alpha = \frac{1}{8} \bar{f}_v (1 - 2\beta)$.*

Proof. First, Lemma 4 shows that by round $R_\ell + k$, a large fraction of voter blocktree's main chain has a αk deep vote on a proposer block at level ℓ . Next, Lemma 5 shows this a large fraction of these votes permanently remain in the main chain.

Lemma 4. *If event $F_j[R_\ell : R_\ell + k]$ occurs, then by round $R_\ell + k$, a honest voter block is at least αk -deep in voting chain j .*

Proof. Refer to Appendix subsection C.4. \square

Lemma 5. *If event $F_j[R_\ell : R_\ell + k]$ occurs, the fixed αk -deep voter block b in the main chain of voter blocktree j at round $R_\ell + k$ will permanently remain in the main chain.*

Proof. Refer to Appendix subsection C.5. \square

From typicality in Theorem 5, we know that event $F_j[R_\ell : R_\ell + k]$ occurs for $1 - \left(\frac{1}{8f_v d} \vee \frac{1-2\beta}{24 \log m}\right)$ fraction of voter blocktrees with probability $1 - e^{-\frac{(1-2\beta)m}{48 \log m}}$ and this along with Lemma 4 and 5 completes the proof. \square \square

B.4 Proposer-list confirmation policy

We repeat the definitions subsection 5.5.2 for the sake of readability. Let $\mathcal{P}_\ell(r) = \{p_1, p_2, \dots\}$ be the set of proposer blocks at level ℓ at round r . Let θ be the fraction of 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 number of votes at depth d or greater for a proposer blocks in the subset $\mathcal{P}_\ell(r) - \{p_n\}$. Let $\delta_d \triangleq \left(\frac{1}{4f_v d} \vee \frac{1-2\beta}{24 \log m}\right)$. The next lemma bounds the future number of votes on a block.

Lemma 6. *The number of votes in a future round $r' \geq r$, $V_p(r')$, satisfies*

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

wp $1 - r_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$, where

$$\underline{V}_n(r) \triangleq \max_{d \geq 0} \{(V_n^d(r) - \delta_d m)^+\},$$

$$\bar{V}_n(r) \triangleq V_n(r) + \left(V_{-n}(r) - \max_{d \geq 0} \{(V_{-n}^d(r) - \delta_d m)^+\} \right) + \theta.$$

Proof. Appendix C.6. □

Any private hidden block $p_{\text{private}} \notin \mathcal{P}_\ell(r)$, has zero votes in round r . From Theorem 6, we know that p_{private} can have at most $\min_{d \geq 0} \delta_d m + \theta$ votes in future round $r' \geq r$. Thus we have

$$V_{\text{private}}(r') \leq \bar{V}_{\text{private}}(r) \triangleq \min_{d \geq 0} \delta_d m \quad \forall r' \geq r.$$

Let us assume without loss of generality $\underline{V}_1(r) \geq \underline{V}_i(r)$ for list $\mathcal{P}_\ell(r) = \{p_1, p_2, \dots\}$.

Definition 1. *If $\underline{V}_1(r) > \bar{V}_{\text{private}}(r)$, then the level ℓ has a well defined proposer list $\Pi_\ell(r)$:*

$$\Pi_\ell(r) \triangleq \{p_i : \bar{V}_i(r) > \underline{V}_1(r)\}.$$

Proposer list confirmation policy: If the proposer list at level ℓ is well defined in round r , then we confirm the list of proposer blocks $\Pi_\ell(r)$.

Lemma 7. *If the proposer list at level ℓ is confirmed in round r , then the (final) leader block at level ℓ , $p_\ell^*(r_{\max})$, whp satisfies*

$$p_\ell^*(r_{\max}) \in \Pi_\ell(r),$$

Proof. Appendix C.7. □

Lemma 8. *If the proposer lists at all levels $\ell' \leq \ell$ are confirmed by round r , then the (final) leader sequence up to level ℓ , $\text{LedSeq}_\ell(r_{\max})$, whp satisfies*

$$\text{LedSeq}_\ell(r_{\max}) \in \Pi_1(r) \times \dots \times \Pi_\ell(r),$$

Proof. Appendix C.8. □

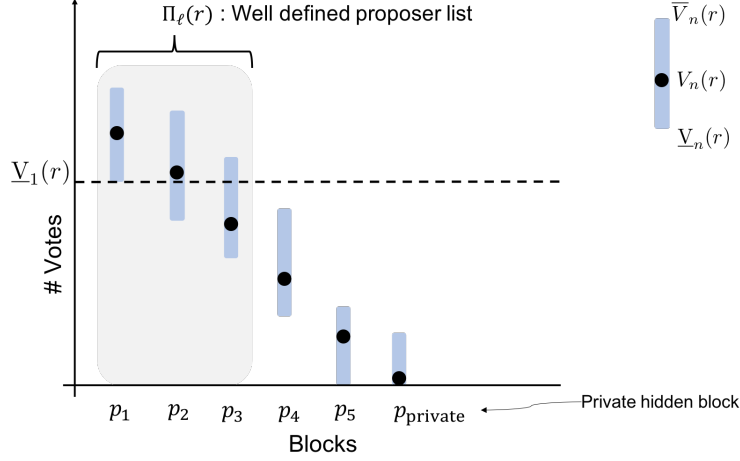


Fig. 15. In the above example, the proposer list is *well defined* at level ℓ and it is $\Pi_\ell(r) = \{p_1, p_2, p_3\}$.

B.5 Latency: Confirming a list of proposer blocks at level ℓ

Proposition 1. *The first proposer block at level ℓ is mined at round R_ℓ and let $r = R_\ell + k$. The proposer list at level ℓ can be confirmed if*

$$\text{Case 1. } |P_\ell(R_\ell + k)| + 1 < \frac{1}{\delta_k}, \quad (25)$$

$$\text{Or Case 2. } k = c_1 m$$

wp $e^{-\frac{(1-2\beta)m}{48 \log m}}$. Here $\delta_k \triangleq \left(\frac{1}{4f_v k} \vee \frac{1-2\beta}{24 \log m} \right)$ and $c_1 = \frac{12}{f_v(1-2\beta)}$.

Proof. By definition the ℓ' -th element in $\text{LedSeq}_\ell(r_{\max})$ is $L_{\ell'}(r_{\max})$ and thus the proof directly follows from Lemma 7. \square

We now use the above lemma to calculate the expected number of rounds to *well* define the proposer block list at level ℓ . The first proposer block at level ℓ is mined in round R_ℓ . The honest users do not mine new proposer blocks at level ℓ after round R_ℓ , however, the adversary can mine new blocks on on level ℓ after round R_ℓ . At a future round $R_\ell + k$, the number of of proposer blocks at level ℓ is $|P_\ell(R_\ell + k)|$. Let us define the stopping round for Case 1 (25):

$$K_\ell = \left\{ \min k \text{ s.t. } |P_\ell(R_\ell + k)| + 1 < \frac{1}{\delta_k} \right\}. \quad (26)$$

Lemma 9. *The proposer list at level ℓ can be confirmed at round $R_\ell + (K_\ell \wedge c_1 m)$ and we have*

$$\mathbb{E}[K_\ell \wedge c_1 m \log m] \leq \frac{6}{f_v(1-2\beta)} + \frac{12}{f_v(1-2\beta)m^2}.$$

Proof. Appendix C.9 □

Lemma 9 upper bounds the expected number of rounds to *well* define the proposer list at level ℓ .

B.6 Latency: Well defining a list of proposer blocks *up to* level ℓ

To form a list of proposal blocks sequences upto level ℓ , we have to confirm the proposer-block list for all levels $\ell' \leq \ell$ and we now analyze the expected number of rounds to confirm proposal block list up to level ℓ . Let us define $D_{\ell', \ell} \triangleq R_\ell - R_{\ell'}$ be the difference between the rounds when the first block at level ℓ and ℓ' was mined.

Proposition 2. *The first proposer block at level $\ell' (< \ell)$ is mined at round $R_{\ell'}$ and let $r = R_\ell + k$. All the proposer lists up to level ℓ' can be confirmed if*

$$\text{Case 1. } |P_\ell(R_\ell + k)| + 1 < \frac{1}{\delta_{k+D_{\ell', \ell}}}, \quad \forall \ell' \leq \ell \quad (27)$$

$$\text{Or Case 2. } k = \max_{\ell' \leq \ell} (c_1 m - D_{\ell', \ell})_+$$

wp $e^{-\frac{(1-2\beta)m}{48 \log m}}$. Here $\delta_{k+D_{\ell', \ell}} \triangleq \left(\frac{1}{4\bar{f}_v(k+D_{\ell', \ell})} \vee \frac{1-2\beta}{24 \log m} \right)$ and $c_1 = \frac{12}{\bar{f}_v(1-2\beta)}$.

Proof. This is a direct consequence of Propositions 9 and 13. □ □

Along the lines of Equations (26) and (47), Let us define the stopping round for Case 1 (27):

$$K_\ell^* = \left\{ \min k \text{ s.t. } \mathcal{P}_{\ell'}[R_{\ell'} : R_\ell + k] + 1 < \frac{1}{\delta_{k+D_{\ell', \ell}}} \quad \forall \ell' \leq \ell \right\}.$$

Lemma 10. *All the proposal lists up to level ℓ are well defined confirmed by round $R_\ell + R_\ell^{\text{conf}}$, where ℓ is given by*

$$\mathbb{E}[R_\ell^{\text{conf}}] = \left[K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell', \ell})_+ \right] \leq \frac{8}{\bar{f}_v(1-2\beta)^2} + \frac{144}{\bar{f}_v(1-2\beta)^2 m}.$$

Along with this

Proof. Appendix C.10 □

C Appendix: Proofs of Lemmas used in Appendix B

C.1 Proof of Lemma 1

In order to prove the result, let us define an event $D_j[r : r+k] = \{Y_j([r : r+k]) < Z_j([r : r+k]) + \frac{1}{2}\bar{f}_v(1-2\beta)k\}$ and upper bound its probability:

$$\begin{aligned} \mathbb{P}(D_j[r : r+k]) &= \mathbb{P}(Y_j([r : r+k]) - Z_j([r : r+k]) < \frac{1}{4}\bar{f}_v(1-2\beta)k) \\ &= \mathbb{P}\left(Y_j([r : r+k]) - Z_j([r : r+k]) - \frac{1}{2}\bar{f}_v(1-2\beta)k < -\frac{1}{4}\bar{f}_v(1-2\beta)k\right) \\ &\stackrel{(a)}{\leq} e^{-\gamma_1 k}. \end{aligned} \quad (28)$$

The value of γ_1 is $\frac{1}{12}\bar{f}_v(1-2\beta)$. The inequality (a) follows from Chernoff bounds. Here the random variables $Y_j([r: r+k]) \sim \text{Bin}(\bar{f}_v'(1-\beta), k)$ ⁵ and $Z_j([r: r+k]) \sim \text{Pois}(\bar{f}_v\beta k)$. The probability of the event $F_j[r: r+k]$ is given by:

$$\begin{aligned} F_j[r: r+k] &= 1 - \mathbb{P}(F_j^c[r: r+k]) \\ &= 1 - \mathbb{P}(D_j[r: r+k]) - \mathbb{P}(F_j^c[r: r+k] \mid D_j[r: r+k]) \\ &\geq 1 - e^{-\gamma_1 k} - e^{-\gamma_2 k} \\ &\geq 1 - 2e^{-\gamma k}. \end{aligned} \tag{29}$$

The values of $\gamma = \gamma_2 \geq \frac{1}{12}\bar{f}_v(1-2\beta)^2$ and is obtained by using $c_1 = \frac{1}{4}\bar{f}_v(1-2\beta)$ in Lemma 23.

C.2 Proof of Lemma 2

For voter blocktree j , let b_j denote the d -deep voter block on its main chain mined in round $r(b_j)$. Consider the following event for voter blocktree j :

$$G_j([r', r]) = \{Y_j([r': r]) + Z_j([r': r]) < d\}. \tag{30}$$

Event $G_j([r', r])$ implies that less than d voter blocks were mined rounds $[r': r] = \{r' + 1, r' + 2, \dots, r\}$ and this further implies that $r(b_j) < r'$ because block b_j is d -deep. The probability of this event is :

$$\begin{aligned} \mathbb{P}(G_j([r': r])) &= 1 - \mathbb{P}(G_j^c([r': r])) \\ &= 1 - \mathbb{P}(\text{Bin}(\bar{f}_v'(1-\beta), \frac{d}{2\bar{f}_v}) + \text{Pois}(\frac{\beta d}{2}) > d) \\ &= 1 - \mathbb{P}(\text{Pois}(\frac{d}{2}) > d) \\ &\stackrel{(a)}{\geq} 1 - e^{-\frac{d}{2}}. \end{aligned}$$

The inequality (a) is obtained using Chernoff bound⁶. We now consider indicator random variables $\mathbf{1}(G_j[r': r])$ for $j \in [m]$. These random variables are identical and independent random variables with probability $1 - e^{-\frac{d}{2}}$. Thus using Chernoff bound, similar to proof of Theorem 5, the fraction of voting chains for where $G_j([r', r])$ occurs is

$$\frac{1}{m} \sum_{j=1}^m \mathbf{1}(G_j[r': r]) \geq 1 - \frac{1}{8d} \stackrel{(a)}{\geq} 1 - \frac{1}{8\bar{f}_v d} \quad \text{wp} \quad 1 - e^{-\frac{m}{48 \log m}}.$$

Inequality (a) follows because our regime of interest is $\bar{f}_v < 1$.

⁵ $\bar{f}_v' = \bar{f}_v e^{-\bar{f}_v \Delta}$

⁶ <https://github.com/ccanonne/probabilitydistributiontoolbox/blob/master/poissonconcentration.pdf>

C.3 Proof of Lemma 3

We will prove this by contradiction. Refer Fig. 16 for a visual proof. Say at a future round $r + a$ ($a > 0$), there are two chains C_1, C_2 s.t $b_j \in C_1, \notin C_2$ and $\text{length}(C_2) = \text{length}(C_1)$.

Consider the latest block \tilde{b} on the common prefix of C_1 and C_2 that was mined by an honest party. Let \tilde{r} be the round on which \tilde{b} was mined⁷. It is important to note that $\tilde{r} \leq r(b_j) < r'$ because a) $b_j \notin C_2$, and b) $G_j([r', r])$ (Equation 30) occurs. Consider the set of rounds $[\tilde{r} : r]$. For each unique level ℓ , only one of the level ℓ blocks in C_1 and C_2 is mined by honest users and thus $Z_j([\tilde{r} : r + a]) \geq Y_j([\tilde{r} : r + a])$. However, since event $F_j([r' : r])$ occurred, we know that $Y_j([\tilde{r} : r + a]) > Z_j([\tilde{r} : r + a]) + \frac{1}{4}\bar{f}_v(1 - 2\beta)(r + a - \tilde{r})$, hence a contradiction. \square

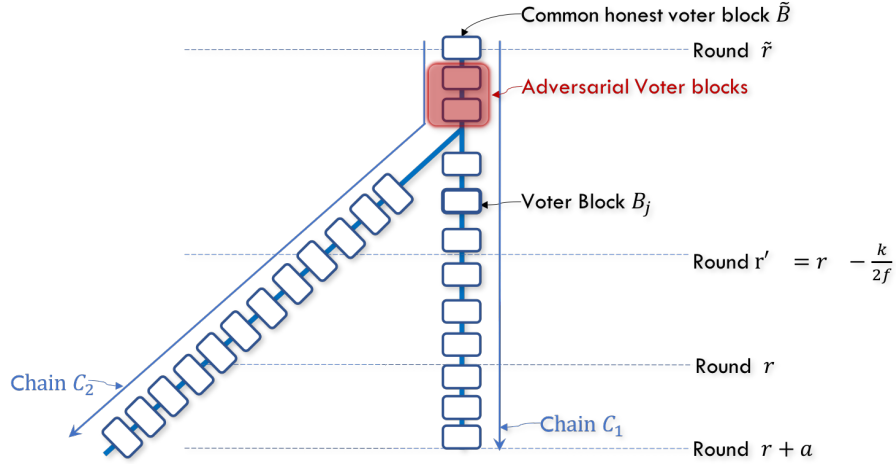


Fig. 16.

C.4 Proof of Lemma 4

Let the main chain of this blocktree j at round $R_\ell + k$ be denoted by $C = \{b_1, b_2, \dots\}$. Let block b_n be computed in round $r(b_n)$. Let block $b_{\tilde{\ell}} \in C$ be the last honest block computed before round R_ℓ and $R_{\tilde{\ell}}$ be the round in this block \tilde{b}_ℓ was mined. i.e.,

$$\ell \triangleq \{ \max n \text{ s.t } r(b_n) \leq R_\ell \text{ and } b_n \text{ is a honest block.} \}$$

$$\tilde{R}_\ell \triangleq r(\tilde{b}_\ell)$$

⁷ If no such block exists then $\tilde{r} = 0$ (voter genesis block).

Observation: All the blocks $B_\ell \in C$ mined in rounds between \tilde{R}_ℓ and R_ℓ are adversarial blocks by definition.

Proposition 1. *If $Y_j([\tilde{R}_\ell: R_\ell + k]) - Z_j([\tilde{R}_\ell: R_\ell + k]) \geq \alpha k$, then at round $R_\ell + k$, there is a vote at least αk -deep.*

Proof. If $Y_j([\tilde{R}_\ell: R_\ell + k]) - Z_j([\tilde{R}_\ell: R_\ell + k]) \geq \alpha k$, then the main chain of blocktree j has at least αk honest vote-blocks. From previous observation we know that there are no honest blocks on the main chain between rounds \tilde{R}_ℓ and R_ℓ , and thus all these αk honest vote-blocks on the main chain are mined after round R_ℓ . The earliest of these αk vote-blocks votes on a leader block at depth ℓ and is at least αk -deep. \square \square

As before, $Y_j([\tilde{R}_\ell: R_\ell + k])$, $Z_j([\tilde{R}_\ell: R_\ell + k])$ are the number of blocks mined between rounds \tilde{R}_ℓ and $R_\ell + k$ by honest and adversary respectively. Since event $F_j[R_\ell: R_\ell + k]$ occurs, the event $Y_j([\tilde{R}_\ell: R_\ell + k]) > Z_j([\tilde{R}_\ell: R_\ell + k]) \geq \frac{1}{8}\bar{f}_v(1 - 2\beta)$ also occurs. Using the above proposition we obtain the result for $\alpha \geq \frac{1}{8}\bar{f}_v(1 - 2\beta)$. \square

C.5 Proof of Lemma 5

We will prove this by contradiction. Refer Fig 17. Suppose until round $r - 1$, the main chain of voter blocktree j has block B . At at round $r(> R_\ell + k)$ there are two chains C_1, C_2 s.t $B \in C_1, \notin C_2$ and $\text{len}(C_2) > \text{len}(C_1)$.

Consider the last block \tilde{B} on the common prefix of C_1 and C_2 that was mined by an honest party and let \tilde{r} be the round on which \tilde{B} was mined (if no such block exists let $\tilde{r} = 0$). It is important to note $\tilde{r} \leq R_\ell$ because a) all the voter blocks on C_1 mined after round R_ℓ and before block B are private blocks, b) $B \notin C_2$. Consider the set of rounds between \tilde{r} and r , $[\tilde{r}: r]$. For each depth ℓ , only one of the depth ℓ blocks in C_1 and C_2 is mined by honest users and thus $Z([\tilde{r}: r]) \geq Y([\tilde{r}: r])$. However since event $F_j([R_\ell: R_\ell + k])$ occurred and $\tilde{r} < R_\ell$, $R_\ell + k < r$, we know that $Y_j([\tilde{r}: r]) > Z_j([\tilde{r}: r]) + \frac{1}{4}\bar{f}_v(1 - 2\beta)(\tilde{r} - r)$, hence a contradiction. \square

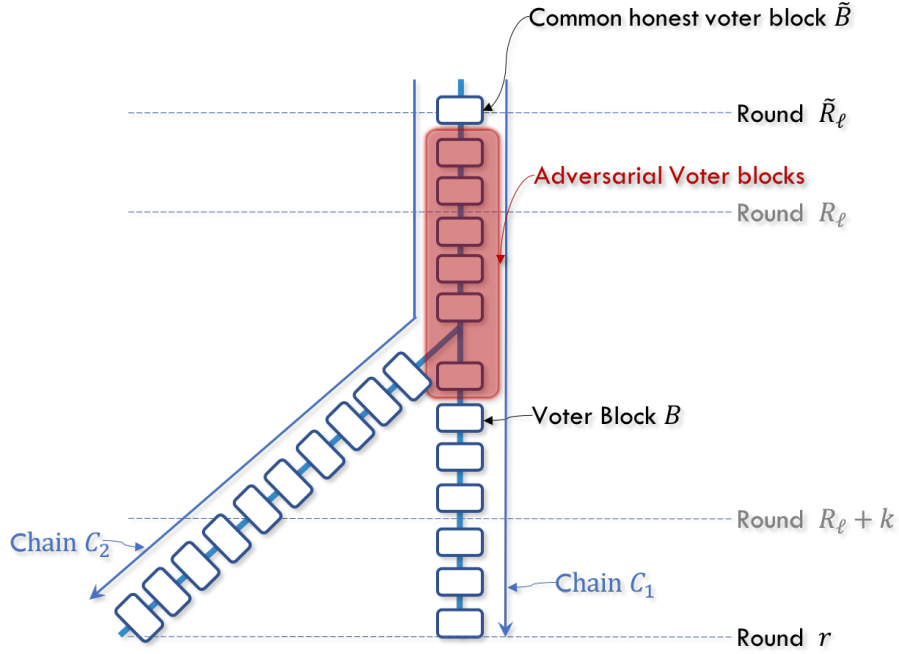


Fig. 17.

C.6 Proof of Lemma 6

Theorem 6 shows that $1 - \delta_d$ fraction of votes at depth d or greater are permanent whp⁸. Therefore $\underline{V}_n(r) \triangleq \max_{d \geq 0} \{(V_n^d(r) - \delta_d m)^+\}$ is a lower bound on $V_n(r')$ – the number of votes in a future round $r' > r$.

Following the same line of reasoning, $\underline{V}_{-n}(r) \triangleq \max_{d \geq 0} \{(V_{-n}^d(r) - \delta_d m)^+\}$ is a lower bound on $V_{-n}(r')$. Thus at most $(V_{-n}(r) - \underline{V}_{-n}(r)) + \theta$ can existing votes can be removed from blocks in $\mathcal{P}_\ell(r) - \{p_n\}$ and added to block p_n . Also the θ chains which have not yet voted could also vote on block p_n . These two combined gives us the upper bound on $V_n(r')$.

C.7 Proof of Lemma 7

We prove by contradiction. Say the final leader block is b_i and is not in set $\Pi_\ell(r)$. Since $b_i \notin \Pi_\ell(r)$, by definition of $\Pi_\ell(r)$ and Lemma 6, for any future round $r' \geq r$ we have

$$\bar{V}_i(r) < \underline{V}_1(r) \leq V_1(r'). \quad (31)$$

⁸ By high probability in this subsection, we mean probability at least $1 - R_{\max}^2 e^{-\frac{(1-2\beta)m}{48 \log m}}$.

Lemma 6 we gives us

$$V_i(r') \leq \bar{V}_i(r). \quad (32)$$

From Equations (31) and (32), we have $V_i(r') < V_1(r')$ and therefore b_i cannot be the leader block in any future rounds $r' \geq r$ which includes the round R_{\max} .

C.8 Proof of Lemma 8

By definition the ℓ' -th element of $\text{LedSeq}_\ell(r_{\max})$ is $p_{\ell'}^*(r_{\max})$. From Lemma 7 we have $p_{\ell'}^*(r_{\max}) \in \Pi_{\ell'}(r)$ for all levels $\ell' \leq \ell$ and thus the required result follows.

C.9 Proof of Lemma 9

Proof. The random variable $P_\ell(R_\ell + k)$ satisfies $P_\ell(R_\ell + k) \leq 2 + Z_\ell^p(R_\ell) + Z_\ell^p([R_\ell : R_\ell + k])$, where a) ‘2’ corresponds to the number of blocks mined by the honest users, b) $Z_\ell^p(R_\ell)$ denote the number of proposer blocks at level ℓ presented by the adversary at round R_ℓ , and c) let $1 + Z_\ell^p([R_\ell : R_\ell + k])$ denote the number of proposer blocks mined by the adversary at level ℓ from round $R_\ell + 1$ to round $R_\ell + k$. It is shown in Appendix G.1 that $Z_\ell^p(R_\ell) \sim \text{Geometric}(1 - 2\beta)$, and $Z_\ell^p([R_\ell : R_\ell + k]) \sim \text{Poiss}(\bar{f}_v \beta k)$. Therefore, $P_\ell(R_\ell + k)$ is random variable and it can increase with k .

It is easy to see that the tail event $\{K_\ell > k\} = \{P_\ell(R_\ell + k) + 1 > \frac{1}{\delta_k}\}$. For $k f_v < \frac{6 \log m}{1-2\beta}$, we have

$$\begin{aligned} \mathbb{P}(K_\ell > k) &= \mathbb{P}(P_\ell(R_\ell + k) + 1 > 4\bar{f}_v k) \\ &= \mathbb{P}(Z_\ell^p(R_\ell) + Z_\ell^p([R_\ell : R_\ell + k]) + 2 > 4\bar{f}_v k) \\ &= \mathbb{P}(Z_\ell^p([R_\ell : R_\ell + k]) > 4\bar{f}_v k - Z_\ell^p(R_\ell) - 2) \end{aligned} \quad (33)$$

Remark 1. Observe $\bar{f}_v k - 2 > 0 \ \forall k \geq \frac{2}{\bar{f}_v}$. Here let

$$k_1 = \frac{2}{\bar{f}_v} \quad (34)$$

Using the above remark, LHS of Equation 33 can be divided into three cases

$$\mathbb{P}(K_\ell > k) \leq \begin{cases} 1 & k \leq k_1 \\ \mathbb{P}(Z_\ell^p([R_\ell : R_\ell + k]) > 3k\bar{f}_v - Z_\ell^p(R_\ell)) & k_1 < k \leq \frac{6 \log m}{\bar{f}_v(1-2\beta)} \\ \mathbb{P}(Z_\ell^p([R_\ell : R_\ell + k]) > \frac{6 \log m}{1-2\beta} - 1 - Z_\ell^p(R_\ell)) & k > \frac{6 \log m}{\bar{f}_v(1-2\beta)} \end{cases} \quad (35)$$

Now let us upper bound $\mathbb{P}(K_\ell > k)$ as Case 2 in Equation (35):

$$\begin{aligned}
& \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) > 3k\bar{f}_v - Z_\ell^p(R_\ell)) \\
&= \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) - k\bar{f}_v > 2k\bar{f}_v - Z_\ell^p(R_\ell)) \\
&\leq \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) - k\bar{f}_v > 2k\bar{f}_v - Z_\ell^p(R_\ell) \mid Z_\ell^p(R_\ell) < k\bar{f}_v) + \mathbb{P}(Z_\ell^p(R_\ell) > k\bar{f}_v) \\
&\leq \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) - k\bar{f}_v > k\bar{f}_v \mid Z_\ell^p(R_\ell) < k\bar{f}_v) + \mathbb{P}(Z_\ell^p(R_\ell) > k\bar{f}_v) \\
&\leq \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) - k\bar{f}_v > k\bar{f}_v) + \mathbb{P}(Z_\ell^p(R_\ell) > k\bar{f}_v) \\
&\stackrel{(a)}{\leq} e^{-\frac{k\bar{f}_v}{2}} + e^{-k\bar{f}_v \frac{(1-2\beta)}{2}} \leq 2e^{-\frac{(1-2\beta)f_vk}{2}}. \tag{36}
\end{aligned}$$

The inequality (a) is due to Chernoff bound and cdf of a geometric random variable. Using Equation (36) we directly obtain upper bound $\mathbb{P}(K_\ell > k)$ as Case 3 in Equation (35):

$$\mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) > \frac{6 \log m}{1-2\beta} - 1 - Z_\ell^p(R_\ell)) \leq 2e^{-3 \log m} = \frac{2}{m^3}. \tag{37}$$

Using Equations (33), (35), (36), and (37), we obtain

$$\begin{aligned}
\mathbb{E}[K_\ell \wedge c_1 m] &= \int_{k=0}^{c_1 m} \mathbb{P}(K_\ell \geq k) + \mathbb{P}(K_\ell > c_1 m)(c_1 m) \\
&= \int_{k=0}^{k_1} 1 + \int_{k=k_1}^{c_1 m} \mathbb{P}(Z_\ell^p([R_\ell: R_\ell + k]) > 3k\bar{f}_v - Z_\ell^p(R_\ell)) \\
&\quad + \mathbb{P}(K_\ell > c_1 m)(c_1 m) \\
&\leq k_1 + \int_{k=k_1}^{\infty} 2e^{-\frac{(1-2\beta)f_vk}{2}} + \frac{c_1}{m^2} \\
&\leq k_1 + \frac{4}{\bar{f}_v(1-2\beta)} + \frac{c_1}{m^2} \\
&\leq \frac{2}{\bar{f}_v} + \frac{4}{\bar{f}_v(1-2\beta)} + \frac{c_1}{m^2} \\
&\leq \frac{6}{\bar{f}_v(1-2\beta)} + \frac{c_1}{m^2}. \tag{39}
\end{aligned}$$

□

□

C.9.1 Proof of Proposition 1

Let us first consider Case 1. From Theorem 7, we know that at round $r = R_\ell + k$, the lower bound on total number of permanent votes on blocks in set $\mathcal{P}_\ell(r)$ is at least $1 - \delta_k$. Therefore, the lower bound $\underline{V}_1(r) \geq \frac{m(1-\delta_k)}{|\mathcal{P}_\ell(r)|}$. Thus the upper bound on votes on a private block b_{private} is $\bar{V}_{\text{private}}(r) < m\delta_k$. It is easy to see

$$|P_\ell(R_\ell + k)| + 1 < \frac{1}{\delta_k} \implies \underline{V}_1(R_\ell + k) > \bar{V}_{\text{private}}(R_\ell + k).$$

Now let us focus on Case 2. From Lemma 20, we know that the all m votes are permanent wp $1 - \epsilon$ for

$$k = \frac{12}{\bar{f}_v(1-2\beta)^2} \log \frac{2m}{\epsilon}.$$

Substituting $\epsilon = e^{-\frac{(1-2\beta)m}{48 \log m}}$ in the above equation, we conclude that at round $r = R_\ell + c_1 m$, the upper bound on the number of votes on private block, $\bar{V}_{\text{private}}(r) = 0$ and thus $\underline{V}_1(R_\ell + k) \geq 1 > \bar{V}_{\text{private}}(R_\ell + k)$ whp.

C.10 Proof of Lemma 10

Proof. Recall, the first proposer block at level ℓ' is proposed at round $R_{\ell'}$. It is important to note that these blocks can be presented by honest users or the adversary. Let the honest users *first* propose proposer blocks on levels $\{L_1^h, L_2^h, \dots, L_i^h, \dots, L_n^h\}$, $L_i < \ell$. Here L_i 's are a random variables. The subscript h stands for 'honest'. The first proposer block at level L_i^h is produced in round $R_{L_i^h}$. Let $L_{n+1} = \ell$. If the adversary produces a block at level ℓ' , satisfying $L_i < \ell' < L_{i+1}$, then the monotonicity gives us the following constraint $R_{L_i} \leq R_{\ell'} \leq R_{L_{i+1}}$. Let $k_1 = \frac{2}{\bar{f}_v}$ and $\gamma_4 = \frac{\bar{f}_v(1-2\beta)}{2}$, then from Lemma 13 we have

$$\begin{aligned} & \mathbb{E}[K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell', \ell})_+ | \{R_{L_i}\}_{i=1}^n, \{L_i\}_{i=1}^n] \\ & \leq \sum_{\ell' \leq \ell} \left((k_1 - D_{\ell', \ell})_+ + \frac{2}{\gamma_4} e^{-\gamma_4 D_{\ell', \ell}} + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3} \right) \\ & \leq \sum_{\ell' \leq \ell} \left((k_1 - (R_\ell - R_{\ell'}))_+ + \frac{2}{\gamma_4} e^{-\gamma_4 (R_\ell - R_{\ell'})} + \frac{(c_1 m - (R_\ell - R_{\ell'}))_+}{m^3} \right) \\ & \leq \sum_{i \in [n]} \sum_{L_i < \ell' \leq L_{i+1}} \left((k_1 - (R_\ell - R_{\ell'}))_+ + \frac{2}{\gamma_4} e^{-\gamma_4 (R_\ell - R_{\ell'})} + \frac{(c_1 m - (R_\ell - R_{\ell'}))_+}{m^3} \right) \\ & \stackrel{(a)}{\leq} \sum_{i \in [n]} (L_{i+1} - L_i) \left((k_1 - (R_{L_{n+1}} - R_{L_{i+1}}))_+ + \frac{2}{\gamma_4} e^{-\gamma_4 (R_{L_{n+1}} - R_{L_{i+1}})} + \frac{(c_1 m - (R_{L_{n+1}} - R_{L_{i+1}}))_+}{m^3} \right). \end{aligned} \tag{40}$$

The inequality (a) is because $R_{\ell'} \leq R_{L_{i+1}}$.

Let G_j for j be iid random variables s.t $G_j \sim \text{Geometric}(\frac{1}{\bar{f}_v})$. Since the levels L_i and L_{i+1} are mined by the honest users, we have $R_{L_{i+1}} - R_{L_i} \geq \sum_{j=L_i}^{L_{i+1}} G_j$ and $R_{L_{n+1}} - R_{L_{i+1}} \geq \sum_{j=L_{i+1}}^{L_{n+1}} G_j$. Thus Equation (40) can be further upper bounded by

$$\mathbb{E}[K_\ell^* | \{G_j\}, \{L_i\}_{i=1}^n] \leq \sum_{i \in [n]} (L_{i+1} - L_i) \left((k_1 - \sum_{j=L_{i+1}}^{L_{n+1}} G_j)_+ + \frac{2}{\gamma_4} e^{-\gamma_4 (\sum_{j=i}^{j=n} G_j)} + \frac{(c_1 m - \sum_{j=L_{i+1}}^{L_{n+1}} G_j)_+}{m^3} \right)$$

Conditioning on L_i 's and taking expectation over G_j 's give us

$$\begin{aligned} \mathbb{E}[K_\ell^* | \{L_i\}_{i=1}^n] &\leq \sum_{i \in [n]} (L_{i+1} - L_i) \left(\left(k_1 - \frac{(L_{n+1} - L_{i+1})}{\bar{f}_v(1-\beta)} \right)_+ + \frac{2}{\gamma_4} \left(\frac{\bar{f}_v(1-\beta)}{\bar{f}_v(1-\beta) + \gamma_4} \right)^{\frac{(L_{n+1} - L_{i+1})}{\bar{f}_v(1-\beta)}} \right. \\ &\quad \left. + \frac{(c_1 m - \frac{(L_{n+1} - L_{i+1})}{\bar{f}_v(1-\beta)})_+}{m^3} \right) \end{aligned}$$

From Appendix G.1, we know that $(L_{i+1} - L_i - 1) \sim \text{Geometric}(1 - 2\beta)$ and thus on taking expectation over $\{L_i\}_{i=1}^n$, we obtain

$$\begin{aligned} \mathbb{E}[K_\ell^*] &\leq \left(\frac{2-3\beta}{1-2\beta} \right) \sum_{i \in [n]} \left(\left(k_1 - \frac{(n-i)(1-\beta)}{\bar{f}_v(1-2\beta)} \right)_+ + \frac{2}{\gamma_4} \left(\frac{\bar{f}_v(1-\beta)}{\bar{f}_v(1-\beta) + \gamma_4} \right)^{n-i} \frac{1-2\beta}{1-\beta} + \frac{(c_1 m - \frac{n-i}{\bar{f}_v(1-2\beta)})_+}{m^3} \right) \\ &\quad (41) \end{aligned}$$

$$\begin{aligned} &\leq \left(\frac{2-3\beta}{1-2\beta} \right) \left(\frac{k_1^2 \bar{f}_v(1-2\beta)}{2(1-\beta)} + \frac{2\bar{f}_v(1-2\beta)}{\gamma_4^2} + \frac{\bar{f}_v(1-2\beta)c_1^2}{m} \right) \\ &\leq \frac{8}{\bar{f}_v(1-2\beta)^3} + \frac{\bar{f}_v(1-2\beta)c_1^2}{m}. \end{aligned} \quad (42)$$

□

□

C.11 Lemma 11

Lemma 11. $\mathbb{E}[K_\ell^* | \{D_{\ell',\ell}\}_{\ell' \leq \ell}] \leq \sum_{\ell' \leq \ell} \left[\left(\frac{2}{\bar{f}_v} - D_{\ell',\ell} \right)_+ + \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell',\ell}} \right]$.

Here $D_{\ell',\ell} = R_\ell - R_{\ell'}$.

Lemma 12. The expected rounds to well define the propose list up to level ℓ is given by

$$\begin{aligned} &\mathbb{E} \left[K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell',\ell})_+ | \{D_{\ell',\ell}\}_{\ell' \leq \ell} \right] \\ &\leq \sum_{\ell' \leq \ell} \left[\left(\frac{2}{\bar{f}_v} - D_{\ell',\ell} \right)_+ + \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell',\ell}} \right] + \frac{(c_1 m - D_{\ell',\ell})_+}{m^3}, \end{aligned} \quad (43)$$

where $D_{\ell',\ell} = R_\ell - R_{\ell'}$.

Proof. Let us re-write K_ℓ^*

$$K_\ell^* = \{ \min k \text{ s.t. } P_{\ell'}([R_{\ell'} : R_\ell + k]) + 1 < 4\bar{f}_v(k + R_\ell - R_{\ell'}) \quad \forall \ell' \leq \ell \}.$$

Following the definition of K_ℓ^* , we have

$$\begin{aligned} &\mathbb{P}(K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell',\ell})_+ > k | \{D_{\ell',\ell}\}_{\ell' \leq \ell}) \\ &\leq \sum_{\ell' \leq \ell} \mathbb{P}(K_{\ell',\ell} \wedge (c_1 m - D_{\ell',\ell})_+ > k | D_{\ell',\ell}) \end{aligned} \quad (44)$$

This gives us that

$$\begin{aligned}
\mathbb{E}(K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell', \ell})_+ | \{D_{\ell', \ell}\}_{\ell' \leq \ell}) &= \int_{k=0}^{\infty} \mathbb{P}(K_\ell^* \wedge \max_{\ell' \leq \ell} (c_1 m - D_{\ell', \ell})_+ > k | D_{\ell', \ell}) \\
&\leq \sum_{\ell' \leq \ell} \int_{k=0}^{\infty} \mathbb{P}(K_{\ell', \ell} \wedge (c_1 m - D_{\ell', \ell})_+ > k | D_{\ell', \ell}) \\
&= \sum_{\ell' \leq \ell} \mathbb{E}(K_{\ell', \ell} \wedge (c_1 m - D_{\ell', \ell})_+ | D_{\ell', \ell}) \\
&\leq \sum_{\ell' \leq \ell} \left[\left(\frac{2}{\bar{f}_v} - D_{\ell', \ell} \right)_+ + \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell', \ell}} \right. \\
&\quad \left. + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3} + \right] \quad (45)
\end{aligned}$$

The last inequality follows from Lemma 13. \square \square

Lemma 11 upper bounds the latency to confirm a list of ledgers up to level ℓ . This lemma is derived for conditional values of $D_{\ell', \ell} \forall \ell' \leq \ell$. In Lemma 10, we take an expectation over $D_{\ell', \ell}$ to complete the latency bound.

C.12 Tools for proving Lemma 10

Proposition 3. *The first proposer block at level $\ell' (< \ell)$ is mined at round $R_{\ell'}$ and let $r = R_\ell + k$. The proposer list at level ℓ' can be confirmed if*

$$\text{Case 1. } |P_\ell(R_\ell + k)| + 1 < \frac{1}{\delta_{k+D_{\ell', \ell}}}, \quad (46)$$

$$\text{Or Case 2. } k = (c_1 m - D_{\ell', \ell})_+$$

$$\text{wp } e^{-\frac{(1-2\beta)m}{48 \log m}} \text{ where } \delta_{k+D_{\ell', \ell}} \triangleq \left(\frac{1}{4\bar{f}_v(k+D_{\ell', \ell})} \vee \frac{1-2\beta}{24 \log m} \right).$$

Proof. This follows the same logic of the proof of Proposition 9. \square

The analysis here is very similar to the analysis in the previous subsection. Similar to Equation (26), let us define the stopping round for Case 1 (46):

$$K_{\ell', \ell} = \{ \min k \text{ s.t. } \mathcal{P}_{\ell'}(R_\ell + k) + 1 < \frac{1}{\delta_{k+D_{\ell', \ell}}} \}. \quad (47)$$

Lemma 13. *The proposer list at level ℓ' is well defined at round $R_\ell + (K_\ell \wedge (c_1 m - D_{\ell', \ell})_+)$ and we have*

$$\begin{aligned}
\mathbb{E}[K_{\ell', \ell} \wedge (c_1 m - D_{\ell', \ell})_+ | D_{\ell', \ell}] &\leq \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell', \ell}} \\
&\quad + \left(\frac{2}{\bar{f}_v} - D_{\ell', \ell} \right)_+ + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3},
\end{aligned}$$

where $D_{\ell', \ell} = R_\ell - R_{\ell'}$.

Proof. Similar to notations in the proof of Lemma 9, $|P_{\ell'}([R_{\ell'} : R_{\ell} + k])|$ is the number of proposer blocks at level ℓ' mined from time $R_{\ell'}$ to $R_{\ell} + k$. Note that the starting round has changed to $R_{\ell'}$.

Here $|P_{\ell'}([R_{\ell'} : R_{\ell} + k])| \leq 2 + Z_{\ell'}^p(R_{\ell'}) + Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k])$, where 1 corresponds to the number of blocks mined by the honest users, $1 + Z_{\ell'}^p(R_{\ell'})$ is the number of proposer blocks at level ℓ presented by the adversary at round $R_{\ell'}$, and $Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k])$ is the number of proposer blocks at level ℓ from round $R_{\ell'}$ to round $R_{\ell} + k$. It is shown in Appendix G.1 that $Z_{\ell'}^p(R_{\ell'}) \sim \text{Geometric}(1 - 2\beta)$, and $Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) \sim \text{Pois}(\bar{f}_v \beta (k + R_{\ell} - R_{\ell'}))$.

Starting round R_{ℓ} , the number of rounds required to fix the proposer list at level ℓ' is denoted by $K_{\ell', \ell}$. As seen previously, it must satisfy

$$K_{\ell', \ell} = \{ \min k \text{ s.t. } |P_{\ell'}([R_{\ell'} : R_{\ell} + k])| + 1 < 4\bar{f}_v(k + R_{\ell} - R_{\ell'}) \}.$$

Let us define $D_{\ell', \ell} \triangleq R_{\ell} - R_{\ell'}$. It is easy to see that the tail event $\{K_{\ell', \ell} > k\} = \{|P_{\ell'}([R_{\ell'} : R_{\ell} + k])| + 1 > e^{\gamma(k + D_{\ell', \ell})}\}$ and that gives us

$$\begin{aligned} \mathbb{P}(K_{\ell', \ell} > k) &= \mathbb{P}(|P_{\ell'}([R_{\ell'} : R_{\ell} + k])| + 1 > 4\bar{f}_v(k + D_{\ell', \ell})) \\ &= \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) + Z_{\ell'}^p(R_{\ell'}) + 2 > 4\bar{f}_v(k + D_{\ell', \ell})) \\ &= \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) > 4\bar{f}_v(k + D_{\ell', \ell}) - Z_{\ell'}^p(R_{\ell'}) - 2) \end{aligned} \quad (48)$$

Again using the Remark 1, similar to Equations (35), LHS of Equation 48 can be divided into three cases

$$\mathbb{P}(K_{\ell', \ell} > k) \leq \begin{cases} 1 & k + D_{\ell', \ell} \leq k_1 \\ \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) > 3(k + D_{\ell', \ell})\bar{f}_v - Z_{\ell'}^p(R_{\ell'})) & k_1 < k + D_{\ell', \ell} \leq \frac{6 \log m}{\bar{f}_v(1-2\beta)} \\ \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) > \frac{6 \log m}{1-2\beta} - Z_{\ell'}^p(R_{\ell'})) & \frac{6 \log m}{1-2\beta} < k + D_{\ell', \ell}. \end{cases} \quad (49)$$

Now let us upper bound $\mathbb{P}(K_{\ell', \ell} > k)$ as Case 2 in Equation (49):

$$\begin{aligned} &\mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) - (k + D_{\ell', \ell})\bar{f}_v > 2(k + D_{\ell', \ell})\bar{f}_v - Z_{\ell'}^p(R_{\ell'})) \\ &\leq \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) - (k + D_{\ell', \ell})\bar{f}_v > 2(k + D_{\ell', \ell})\bar{f}_v - Z_{\ell'}^p(R_{\ell'}) | Z_{\ell'}^p(R_{\ell'}) < (k + D_{\ell', \ell})\bar{f}_v) \\ &\quad + \mathbb{P}(Z_{\ell'}^p(R_{\ell'}) > (k + D_{\ell', \ell})\bar{f}_v) \\ &\leq \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) - (k + D_{\ell', \ell})\bar{f}_v > (k + D_{\ell', \ell})\bar{f}_v | Z_{\ell'}^p(R_{\ell'}) < (k + D_{\ell', \ell})\bar{f}_v) \\ &\quad + \mathbb{P}(Z_{\ell'}^p(R_{\ell'}) > (k + D_{\ell', \ell})\bar{f}_v) \\ &\leq \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_{\ell} + k]) - (k + D_{\ell', \ell})\bar{f}_v > (k + D_{\ell', \ell})\bar{f}_v) \\ &\quad + \mathbb{P}(Z_{\ell'}^p(R_{\ell'}) > (k + D_{\ell', \ell})\bar{f}_v) \\ &\leq e^{-(k + D_{\ell', \ell})\frac{\bar{f}_v}{2}} + e^{-(k + D_{\ell', \ell})\frac{\bar{f}_v(1-2\beta)}{2}} \leq 2e^{-(k + D_{\ell', \ell})\bar{f}_v(1-2\beta)}. \end{aligned} \quad (50)$$

Using Equation (50) we directly obtain upper bound $\mathbb{P}(K_\ell > k)$ as Case 3 in Equation (49):

$$\mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_\ell + k]) > \frac{6 \log m}{1 - 2\beta} - 1 - Z_{\ell'}^p(R_{\ell'})) \leq 2e^{-3 \log m} = \frac{2}{m^3}. \quad (51)$$

Using Equations (48), (49), (50), and (51), we obtain

$$\begin{aligned} \mathbb{E}[K_{\ell', \ell} \wedge c_1 m | D_{\ell', \ell}] &= \int_{k=0}^{c_1 m - D_{\ell', \ell}} \mathbb{P}(K_{\ell', \ell} \geq k | D_{\ell', \ell}) \\ &\quad + \mathbb{P}(K_{\ell', \ell} > c_1 m \log m - D_{\ell', \ell}) (c_1 m - D_{\ell', \ell})_+ \\ &= \int_{k=0}^{k_1 - D_{\ell', \ell}} 1 + \int_{k=k_1 - D_{\ell', \ell}}^{c_1 m - D_{\ell', \ell}} \mathbb{P}(Z_{\ell'}^p([R_{\ell'} : R_\ell + k]) > 3k \bar{f}_v - Z_{\ell'}^p(R_{\ell'})) \\ &\quad + \mathbb{P}(K_{\ell', \ell} > c_1 m \log m - D_{\ell', \ell}) (c_1 m - D_{\ell', \ell})_+ \\ &\leq (k_1 - D_{\ell', \ell})_+ + \int_{k=0}^{\infty} 2e^{-\frac{\bar{f}_v(1-2\beta)}{2}(k+D_{\ell', \ell})} + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3} \\ &\leq (k_1 - D_{\ell', \ell})_+ + \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell', \ell}} + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3} \\ &\leq \left(\frac{2}{\bar{f}_v} - D_{\ell', \ell}\right)_+ + \frac{4}{\bar{f}_v(1-2\beta)} e^{-\frac{\bar{f}_v(1-2\beta)}{2} D_{\ell', \ell}} + \frac{(c_1 m - D_{\ell', \ell})_+}{m^3} \end{aligned} \quad (52)$$

□

□

The first proposer block at level ℓ' was proposed $D_{\ell', \ell}$ rounds before the proposer block at level ℓ . Intuitively, the proposer list at level ℓ' should be *well* defined $D_{\ell', \ell}$ before the proposer list at level ℓ is *well* defined and Lemma precisely 13 characterizes that. Note that for $\ell' = \ell$, we have $D_{\ell', \ell} = 0$ and the result of Lemma 13 reduces to Lemma 9.

D Honest transactions: Proof of Theorem 4

At the current round r the honest users are mining on level ℓ_{current} over proposer block (say) p_1 and adversary is mining over blocks p_2 . Let \tilde{p} be the common honest ancestor of blocks p_1, p_2 and it (say) was proposed in round \tilde{R} , at level $\tilde{\ell}$ as shown in Fig 18

Remark 2. By definition of the ancestor block p_{ancestor} , the number of adversarial blocks between level $\tilde{\ell}$ and ℓ_{current} is at least equal to the number of honest blocks between level $\tilde{\ell}$ and ℓ_{current} .

Let $k_{\text{good}} = \frac{1}{2\bar{f}_v}$. We divide the *good event* into the following two events:

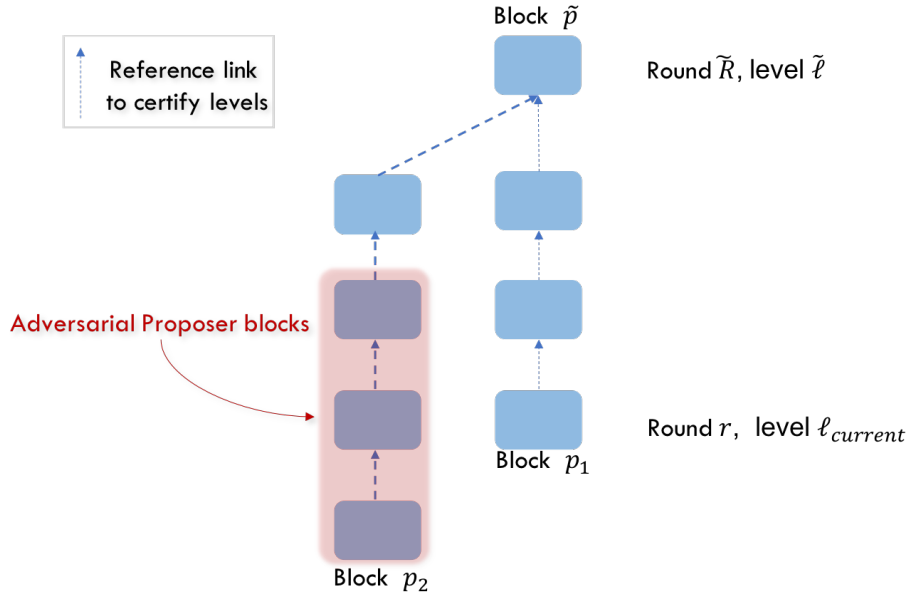


Fig. 18.

1. *Event 1:* At a future round $r+k$, the honest users will have a lead of $P_{\text{lead}} = f_v \beta k_{\text{good}}$ proposer blocks for *all* rounds $r+k+a$ ($a > 0$) over the adversary i.e ,

$$A_k = \{Z[\tilde{R} : r+k+a] < Y[\tilde{R} : r+k+a] - P_{\text{lead}} \forall a \geq 0\} \quad (54)$$

Mathematically $r+k$ is the last round in which random walk $Y[\tilde{R} : r+k+a] - Z[\tilde{R} : r+k+a]$ is on P_{lead} and then never goes below that in the future rounds.

2. *Event 2:* There is a round $R' > r_k$ s.t from round R' to round $R' + k_{\text{good}}$, the adversary mines less than P_{lead} blocks i.e,

$$F_{R'} = \{Z[R' : R' + k_{\text{good}}] < P_{\text{lead}}\}. \quad (55)$$

Lemma 14. Say events A_k occurs and then $F_{R'}$ occurs. Then the transactions tx is included in block B which is proposed in round $r(B) < R' + k_{\text{good}}$ w.p $1 - e^{-r_{\max} \frac{m}{48 \log m}}$.

Proof. Say event A_k occurs at round $r+k$. Then in any future round R' there is always at least $P_{\text{lead}} = f\beta k_{\text{good}} + 1$ levels which have only single honest proposer blocks (and no adversarial blocks). Now suppose event $F_{R'}$ occurs. This implies that in the next k_{good} rounds, the adversary produces at most P_{lead} blocks. Since $k_{\text{good}} = \frac{1}{2f_v}$, from Proposition 13 (substituting $|P_\ell(R_\ell + k)| = 1$ in the Proposition) at least one of above P_{lead} levels will have only a single

honest proposer block (say) p_h with no competitor block and since this proposer block gets more than $\frac{1}{2}$ of the votes by k_{good} rounds wp $1 - e^{-r_{\max} \frac{m}{48 \log m}}$ after its mined, $\Pi_\ell(r') = \{p_h\} \forall r' > R' + k_{\text{good}}$ and it be confirmed as a leader block in round $R' + k_{\text{good}}$. \square \square

Let event A_{K_1} occur in round $r + K_1$. Then *Event 2* occurs after K_2 rounds. Then $\mathbb{E}[R_B - R] = \mathbb{E}[K_1 + K_2]$. The next two lemma shows that $\mathbb{E}[K_1] \leq \frac{1}{(1-2\beta)f_v} + \frac{1}{(1-2\beta)^2 f_v}$, and $\mathbb{E}[K_2] \leq \frac{1}{f_v}$, thus giving us the required result

$$\begin{aligned} \mathbb{E}[K_1 + K_2] &\leq \frac{1}{(1-2\beta)f_v} + \frac{1}{(1-2\beta)^2 f_v} + \frac{1}{f_v} \\ &\leq \frac{3}{(1-2\beta)^2 f_v} \end{aligned} \quad (56)$$

Lemma 15. *Let Event A_{K_1} occur, then The expected number of rounds for Event 1 is K_1 which satisfies*

$$\mathbb{E}[K_1] = \frac{1}{(1-2\beta)f_v} + \frac{1}{(1-2\beta)^2 f_v}$$

Proof. Let us first define the following two rounds

$$\begin{aligned} K_{\text{first}} &= \{ \min k \text{ s.t. } Z[\tilde{R} : R + k] < Y[\tilde{R} : R + k] - P_{\text{lead}} \} \\ K_{\text{stop}} &= \{ \min k \geq K_{\text{first}} \text{ s.t. } Z[\tilde{R} : R + k + a] < Y[\tilde{R} : R + k + a] - P_{\text{lead}} \forall a \geq 0 \} \end{aligned}$$

The number of rounds in which the random walk $Y[\tilde{R} : R + k'] - Z[\tilde{R} : R + k']$ first time hits P_{lead} is $\frac{P_{\text{lead}}}{(1-2\beta)f_v}$ in expectation and thus

$$\mathbb{E}[K_{\text{first}}] = \frac{P_{\text{lead}}}{(1-2\beta)f_v} \leq \frac{1}{(1-2\beta)f_v}. \quad (57)$$

Next, the probability that the random walk never hits P_{lead} is $1-2\beta$. However, if it does hits P_{lead} again, in expectation it takes $\frac{1}{(1-2\beta)f_v}$ ⁹ rounds between the successive hits. Therefore $K_{\text{stop}} - K_{\text{first}} = \sum_{i=0}^N G_i$, where $N \sim \text{Geometric}(2\beta)$ and G_i 's are independent random variables with mean $\mathbb{E}[G_i] = \frac{1}{f_v(1-2\beta)}$. This implies that

$$\mathbb{E}[K_{\text{stop}} - K_{\text{first}}] = \mathbb{E}[N]\mathbb{E}[G_i] = \frac{1}{(1-2\beta)^2 f_v} \quad (58)$$

By definition we have $K_1 = K_{\text{stop}} - K_{\text{first}} + K_{\text{first}}$ and we get the required result from Equations (57) and (58). \square

⁹ The proof is in the link <https://math.stackexchange.com/questions/2423777/error-in-calculation-of-hitting-time-of-1-in-a-biased-random-walk-on-the-intege>

Lemma 16. Say event A_{r+K_1} occurred and event $F_{r+K_1+K_2}$ occurred. Then

$$\mathbb{E}[K_2] \leq \frac{1}{f_v} \quad (59)$$

Proof. Divide the rounds from $\{r + K_1, \infty\}$ into disjoint chunks of length k_{good} , where the q^{th} interval is $[r + K_1 + (q - 1)k_{\text{good}} : r + K_1 + qk_{\text{good}}]$. For $P_{\text{lead}} = f_v \beta k_{\text{good}} + 1$, we have

$$\mathbb{P}(Z[r + K_1 + (q - 1)k_{\text{good}} : r + K_1 + qk_{\text{good}}] < P_{\text{lead}}) \geq \frac{1}{2} \quad \forall q \in \mathbb{Z}^+. \quad (60)$$

Since the events $\{Z[R_1 + (q - 1)k_{\text{good}} : R_1 + qk_{\text{good}}] < P_{\text{lead}}\}$ are independent for $q \in \mathbb{Z}^+$, we have

$$Q_{\text{final}} = \left\{ \min q \in \mathbb{I} \text{ s.t. } Z[r + K_1 + (q - 1)k_{\text{good}} : r + K_1 + qk_{\text{good}}] < P_{\text{lead}} \right\}$$

$$\mathbb{E}[K_2] = \mathbb{E}[Q_{\text{final}}]k_{\text{good}} \leq 2k_{\text{good}} \leq \frac{1}{f_v}.$$

□

□

E Chain quality

The total number of unique proposer blocks mined by the honest users is $Y^p[0 : r_{\text{max}}] \sim \text{Bin}((1 - \beta)\bar{f}'_v, r_{\text{max}})$ and the total number of proposer blocks mined by the adversary is $Z^p[0 : r_{\text{max}}] \sim \text{Poiss}(\beta\bar{f}_v r_{\text{max}})$.

Lemma 17.

$$Y^p[0 : r_{\text{max}}] > \frac{\bar{f}_v r_{\text{max}}(3 + 2\beta)}{8} \quad (61)$$

$$Z^p[0 : r_{\text{max}}] < \frac{\bar{f}_v r_{\text{max}}(1 + 6\beta)}{8} \quad (62)$$

$$wp \ 1 - 2e^{-\frac{(1-2\beta)^2 \bar{f}_v r_{\text{max}}}{200}}.$$

Proof. On applying Chernoff bound on $Y^p[0 : r_{\text{max}}]$, we get

$$\mathbb{P}(Y^p[0 : r_{\text{max}}] < \mathbb{E}[Y^p[0 : r_{\text{max}}]] - a) < e^{-\frac{a^2}{3\mathbb{E}[Y^p[0 : r_{\text{max}}]]}}. \quad (63)$$

The mean is $\mathbb{E}[Y^p[0 : r_{\text{max}}]] = (1 - \beta)\bar{f}'_v r_{\text{max}}$ which is greater than $\frac{1}{2}\bar{f}_v r_{\text{max}}$ for $\bar{f}'_v = \frac{1-2\beta}{2}$. Substituting the value of mean along with $a = \left(\frac{1-2\beta}{8}\right)\bar{f}_v r_{\text{max}}$ in Equation (63), give us

$$\mathbb{P}\left(Y^p[0 : r_{\text{max}}] < \frac{\bar{f}_v r_{\text{max}}(3 + 2\beta)}{8}\right) < e^{-\frac{(1-2\beta)^2 \bar{f}_v r_{\text{max}}}{192(1-\beta)}} < e^{-\frac{(1-2\beta)^2 \bar{f}_v r_{\text{max}}}{200}}. \quad (64)$$

Similarly applying Chernoff bound on $Z^p[0: r_{\max}]$, we have

$$\mathbb{P}(Z^p[0: r_{\max}] > \mathbb{E}[Z^p[0: r_{\max}]] + a) < e^{-\frac{a^2}{\mathbb{E}[Z^p[0: r_{\max}]] + a}}. \quad (65)$$

The mean is $\mathbb{E}[Z^p[0: r_{\max}]] = \beta \bar{f}_v r_{\max}$. Substituting the value of mean along with $a = \left(\frac{1-2\beta}{8}\right) \bar{f}_v r_{\max}$ in Equation (65), gives us

$$\begin{aligned} \mathbb{P}\left(Z^p[0: r_{\max}] > \frac{\bar{f}_v r_{\max}(1+6\beta)}{8}\right) &< e^{-\frac{(1-2\beta)^2 \bar{f}_v r_{\max}}{8(1+4-\beta)}} \\ &< e^{-\frac{(1-2\beta)^2 \bar{f}_v r_{\max}}{200}}. \end{aligned} \quad (66)$$

Equations (64) and (66) proves Equations (61) and (62). \square \square

F Unique ledger decoding: Slow latency

F.1 Unique decoding: Slow confirmation

Theorem 8 (Unique Ordering Latency). *For $\beta < 1/2$, the unique ledger is decoded with latency $\frac{12}{f_v(1-2\beta)^2} \log \frac{2m}{\epsilon}$, and each transaction and its ordering is permanent wp at least $1 - \epsilon$.*

The first proposer block at level ℓ was presented at round R_ℓ . Then we show that the leader blocks upto level ℓ are permanent after $R_{\text{unique}, \epsilon} = O(\frac{1}{f} \log(\frac{2m}{\epsilon}))$ rounds wp $1 - \epsilon$. That is at round $R_\ell + R_{\text{unique}, \epsilon}$, all the leader blocks upto level ℓ are permanent wp $1 - \epsilon$.

Lemma 18. *The first proposer block(s) at level ℓ is proposed in round R_ℓ . By round $R_\ell + k$, a main voting chain will have a **permanent** vote on depth d proposer block w.p $1 - 2e^{-\gamma k}$. Here $\gamma \geq \frac{1}{12} f_v(1 - 2\beta)^2$.*

Proof. Consider the events from expressions (23) and (24):

$$E_j[r: r+k] = \{Y_j([r: r+k]) > Z_j([r: r+k]) + \frac{1}{8} f(1-2\beta)k\} \quad (67)$$

$$F_j[r: r+k] = \bigcap_{a, b \geq 0} E_i[r-a: r+k+b] \quad (68)$$

If event $F_j[R_\ell: R_\ell + k]$ has occurred Lemma 4 and 5, we know that the voting blocktree j is permanent. Lemma 1 prove that $\mathbb{P}(F_j[r: r+k]) > 1 - 2e^{-\gamma k}$ and hence completing the proof \square

Lemma 19 (Consistency). *The first proposer block(s) at level ℓ is proposed in round R_ℓ . By round $R_\ell + \frac{12}{f_v(1-2\beta)^2} \log(\frac{2m}{\epsilon})$ the proposer block at level ℓ is permanent wp $1 - \epsilon$*

Proof. Substituting $k = \frac{1}{\gamma} \log(\frac{2m}{\epsilon})$ in Lemma 18, we get that the vote of voter blocktree j is confirmed wp $\frac{\epsilon}{m}$. Thus using union bound over all the m voter blocktree, votes on level ℓ proposer blocks from all the m voting blocktrees are permanent whp $1 - \epsilon$. This implies that the proposer block at level ℓ is permanent with probability $1 - \epsilon$ \square

Remark 3 (Adaptive confirmation:). Lemma 19 requires votes on each voter blocktree to be secure with probability $1 - \epsilon/m$ and this is an strong condition. This is because in the worst-case, the adversary can ensure that the top two voted proposer blocks (at level ℓ) have same number of votes. However, the vote difference between the top two voted blocks (at level ℓ) is an observable quantity and if this vote difference is $O(m)$, then the leader-block at level ℓ can be confirmed in constant time for $\epsilon > e^{-O(m)}$ because each vote needs to be secure w.p $1 - c$ for $c > 0$. As a result, under no *observable* attack, Prism obtains a constant latency for total ordering independent of ϵ . Note that this is not true for Bitcoin because the adversary's attack on bitcoin is in private, hence non-observable.

Lemma 20. (*Common-prefix*) *The first proposer block(s) at level ℓ is proposed in round R_ℓ . By round $R_\ell + \frac{12}{f_v(1-2\beta)^2} \log(\frac{2m}{\epsilon})$ all the proposer block **up to** level ℓ is permanent wp $1 - \epsilon$. Hence the latency for unique decoding is $\frac{12}{f_v(1-2\beta)^2} \log(\frac{2m}{\epsilon})$ rounds.*

Proof. For a given voter blocktree j , if it's vote on proposer block at level ℓ is permanent and say it is cast by voter block $V_{j,\ell}$. Then from the *Voting rule* of Prism in Section 5.2.1, all the proposer blocks up to level $\ell' \leq \ell$ are also permanent because they are voted by either $V_{j,\ell}$ or its ancestor blocks along the its path to its genesis block. Thus this theorem directly follows from Lemma 19. \square

Proof. Lemma 20 proves that common-prefix property of the leader sequence. In expectation it takes $O(1)$ time for a transaction to enter a unique leader block. Thus this result provides an overall latency of $\frac{12}{f_v(1-2\beta)^2} \log(\frac{2m}{\epsilon})$ for unique-ledger decoding for $\beta < \frac{1}{2}$. \square

Corollary F1. *Suppose $\beta < 0.5$. For $1 - \epsilon$ security, substituting $m = 1$ in Theorem 8 we prove that the latency of Prism and Bitcoin is at most $\frac{12}{f_v(1-2\beta)^2} \log \frac{1}{\epsilon}$ rounds.*

G Others

G.1 Reserve proposer blocks by the adversary

Say the honest users mine the first proposer block at level ℓ in round R_ℓ . Let $1 + Z_\ell^p(R_\ell)$ denote that the number of hidden proposal blocks on level ℓ by the adversary. In order to maximize $Z_\ell^p(R_\ell)$, all these blocks should have a

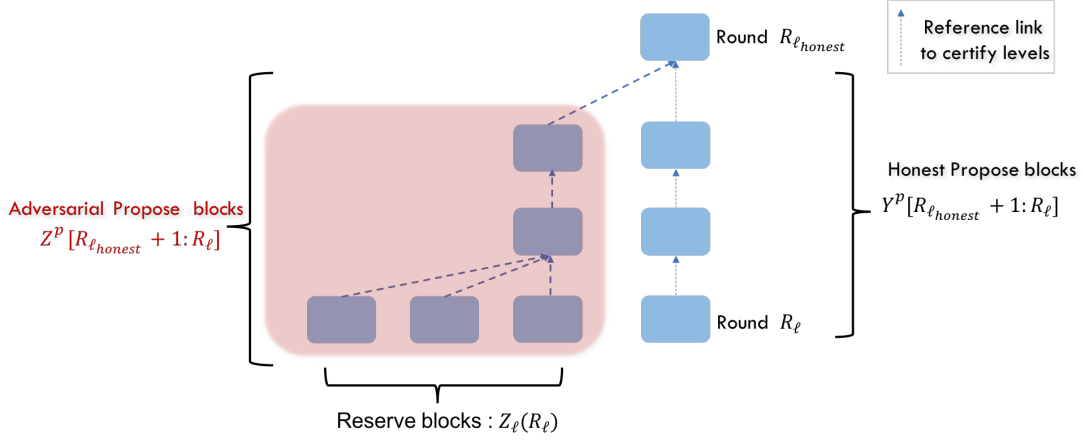


Fig. 19.

common honest parent proposer block at level (say) ℓ_{honest} linked via private proposal blocks as shown in the Figure 19. The total number of reserve blocks is given by

$$1 + Z_{\ell}^p(R_{\ell}) = \max_{\ell_{honest} \leq \ell} Z^p[R_{\ell_{honest}} + 1 : R_{\ell}] - Y^p[R_{\ell_{honest}} + 1 : R_{\ell}] + 1. \quad (69)$$

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

$$\begin{aligned} \mathbb{P}(Z_{\ell}^p(R_{\ell}) > k) &= \max_{\ell_{honest} \leq \ell} \mathbb{P}(Z^p[R_{\ell_{honest}} : R_{\ell}] - Y^p[R_{\ell_{honest}} + 1 : R_{\ell}] > k) \\ &= (2\beta)^k. \end{aligned}$$

Thus $Z_{\ell}^p(R_{\ell}) \in \text{Geometric}(1 - 2\beta)$.

G.2 Random Walk Proofs

Consider the following events from Equation (23) and (24)

$$E_j[r : r + k] = \{Y_j[r : r + k] > Z_j[r : r + k] + \frac{1}{4}f(1 - 2\beta)k\} \quad (70)$$

$$F_j[r : r + k] = \bigcap_{a, b \geq 0} E_j[r - a : r + k + b]. \quad (71)$$

The random variable $W_j[r : r + k] = Y_j[r : r + k] - Z_j[r : r + k]$ is a random walk with drift $\frac{(1-2\beta)f}{2}$. This random walk escapes to ∞ as $k \rightarrow \infty$ wp 1.

Lemma 21. *If $W_j[r : r + k] > c_1 k$, for $c_2 < c_1$ we have*

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

Proof. Refer [2]. □

If the random walk is to the right of $c_1 k$ after k 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 22. *If $W_j[r : r + k] > c_1 k$, for $c_2 < c_1$, then we have*

$$\begin{aligned}\mathbb{P}(W_j[r - b : r + k] > c_2 k \forall b > 0) &= 1 - (2\beta)^{(c_1 - c_2 + 1)k} \\ &= 1 - e^{\log(2\beta)(c_1 - c_2 + 1)k}.\end{aligned}$$

Proof. Refer [2]. □

The above lemma is mathematically characterizing the same event as Lemma 21.

Lemma 23. *If $W_j[r : r + k] > c_1 k$, then we have*

$$\begin{aligned}\mathbb{P}(W_j[r - b : r + k + a] > 0 \forall a > 0) &\geq 1 - 2(2\beta)^{c_1 k/2} \\ &= 1 - 2e^{\log(2\beta)c_1 k/2} \\ &\stackrel{(a)}{\geq} 1 - 2e^{-(1-2\beta)c_1 k/2}\end{aligned}$$

Proof. Using $c_2 = c_1/2$ in the above two Lemmas 21 and 22 we get the required result. The inequality (a) uses $\log 2\beta < 2\beta - 1$ for $\beta > 0$. □