Security of Blockchains at Capacity

Lucianna Kiffer
lkiffer@ethz.ch

Joachim Neu
jneu@stanford.edu

Srivatsan Sridhar
svatsan@stanford.edu

Aviv Zohar
avivz@cs.huji.ac.il

David Tse
dntse@stanford.edu

ABSTRACT

Given a network of nodes with certain communication and computation capacities, what is the maximum rate at which a blockchain can run securely? We study this question for proof-of-work (PoW) and proof-of-stake (PoS) longest chain protocols under a ‘bounded bandwidth’ model which captures queuing and processing delays due to high block rate relative to capacity, bursty release of adversarial blocks, and in PoS, spamming due to equivocations.

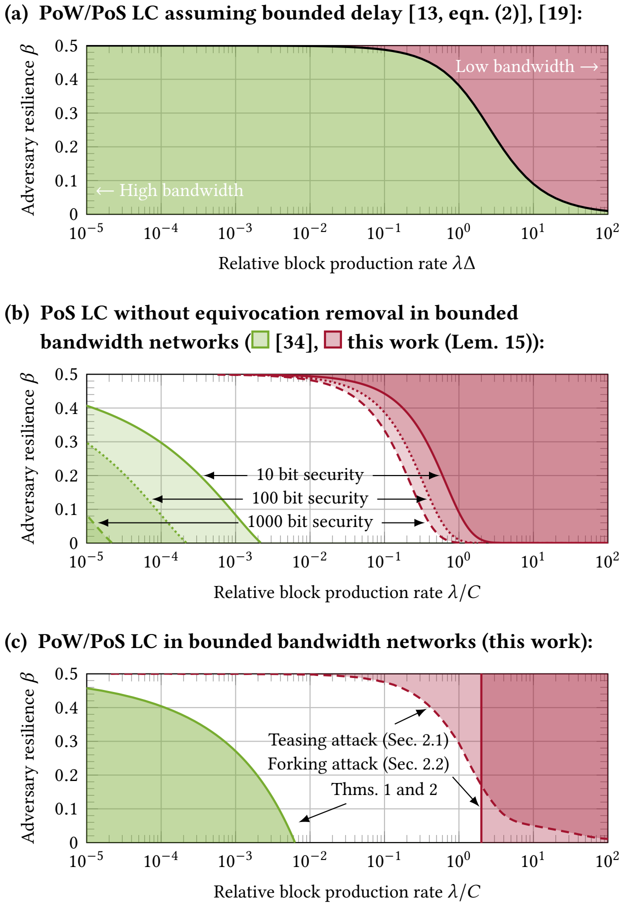
We demonstrate that security of both PoW and PoS longest chain, when operating at capacity, requires carefully designed scheduling policies that correctly prioritize which blocks are processed first, as we show attack strategies tailored to such policies. In PoS, we show an attack exploiting equivocations, which highlights that the throughput of the PoS longest chain protocol with a broad class of scheduling policies must decrease as the desired security error probability decreases. At the same time, through an improved analysis method, our work is the first to identify block production rates under which PoW longest chain is secure in the bounded bandwidth setting. We also present the first PoS longest chain protocol, SaPoS, which is secure with a block production rate independent of the security error probability, by using an ‘equivocation removal’ policy to prevent equivocation spamming.

1 INTRODUCTION

The goal of a blockchain protocol is to create a *secure* and *decentralized* ledger of transactions. This protocol is run by a network of nodes, each with certain capabilities in terms of communication rates and computing power. In this work, we study the connection between these processing capacities (in the wider sense) of individual nodes, and the security of the system.

In order to remain secure under adversarial conditions, blockchain protocols have been parameterized to leave a ‘security margin’ between the transaction rate under normal operation, and each node’s capacity limits. For instance, Bitcoin only produces one block of transactions per ten minutes, even though it usually only takes a few seconds for a node to download and process each block [12]. On the other hand, protocols that push close to the limits of their nodes, become insecure when the processing capacities of nodes are overwhelmed (such as Solana [30, 37, 31]). The natural question to ask then is: given a capacity limit of nodes, what is the maximum block rate under which a blockchain remains secure?

In this work, we focus on longest chain (LC) protocols (a.k.a. Nakamoto consensus [32])—a well-studied class of blockchain protocols that can be instantiated using various Sybil resistance mechanisms such as proof-of-work (PoW) [32, 18] and proof-of-stake (PoS) [11, 2, 36, 10]. This protocol selects the nodes that will mine the next block based on a continuously running lottery. A selected node collects pending transactions, creates a new block extending



(a) PoW/PoS LC assuming bounded delay [13, eqn. (2)], [19]:

(b) PoS LC without equivocation removal in bounded bandwidth networks (🟩 [34], 🟥 this work (Lem. 15)):

(c) PoW/PoS LC in bounded bandwidth networks (this work):

Figure 1: Regions of fraction β of adversarial nodes, block production rate λ, and network model parameters delay bound Δ or bandwidth C, with security proofs (🟩) and attacks (🟥) for Nakamoto consensus. (a) In the bounded delay model, the tradeoff is fully characterized by $\beta = \frac{1-\beta}{1+(1-\beta)\lambda\Delta}$. (b) In [34], λ/C decreases with the security parameter κ. One of our attacks (App. G) shows that *some* dependency of λ/C on κ is unavoidable for the protocols studied in [34]. (c) In this work, our attacks show that resilience β has a sharp cutoff at large λ/C, rather than an asymptotic decay suggested by the bounded delay analysis. Our security proof shows that λ/C independent of κ suffices for secure PoW, and PoS LC with a protocol modification *equivocation removal.*

LK, JN, SS and AZ are listed alphabetically.

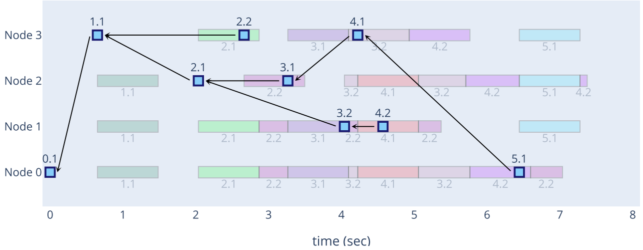


Figure 2: A trace of a simulation of the longest chain protocol using a scheduling rule. This trace displays the timeline of events for each of 4 nodes. The nodes have a total mining rate $\lambda = 1$, and each processes blocks at a rate of $C = 1$. Blocks appear as squares at coordinate (t, p) if they were mined by node p at time t. The i-th block of height h to be mined is named $h.i$. For each node, we also display which blocks it is processing throughout the simulation, using horizontal bars from start to end of the processing.

the longest chain of blocks it sees, and sends the block to the network. Nodes must download and process the transactions in a chain before they can extend it. Attackers prevail if they manage to grow a chain at a faster pace than the honest nodes do, which they can then use to double-spend or to censor transactions.

It seems at first that it is sufficient for nodes to have enough processing capacity to keep up with block production. However, in LC, the production of blocks occurs at random times, which means the network and computing load is bursty. With limited processing capacity, nodes must queue blocks for processing. Even without adversarial activity, the resulting queuing delays increase the time it takes to process blocks. Moreover, a malicious node can selectively delay the release of blocks that it produces, so that the processing load is not just purely random but is to some degree determined adversarially. In PoS, the adversary can additionally produce equivocations—conflicting versions of the block it was allowed to produce—and send them to different nodes. Nodes cannot always predict which of two conflicting blocks will eventually be part of the chain and may thus waste processing capacity on blocks that are later discarded. Attackers can use this to increase load and queuing delays. While blocks are waiting to be processed, nodes cannot mine on top of them, and the honest nodes' chain slows down. This makes it easier for an adversary to attack the system.

Fig. 2 shows a sample trace of a simulation of the proof-of-work longest chain protocol. The figure presents both block creation events, and block processing activity. Queuing effects are evident. For example, node 0 processes blocks almost without pause, occasionally preempting for higher over lower blocks. Several times, blocks are created on the same height due to delays in processing.

Due to queuing of blocks, nodes need a carefully chosen 'scheduling rule' to determine which blocks to process first. We observe that choosing the right scheduling policy is challenging. Different attacks can be carried out by the adversary, depending on this scheduling rule, slowing down the growth of the honest chain and breaking the blockchain's security (Figs. 6 and 7, details in Sec. 2).

To analyze the security of Nakamoto consensus, previous work [25, 18, 13, 36, 38, 19] has considered the 'bounded delay' model. In this model, blocks are processed by all honest nodes within a fixed time of Δ seconds after they are published. The works [13, 19] give a tight characterization of the tradeoff between the fraction β of adversarial nodes, delay bound Δ, and the block production rate λ (Fig. 1(a)). However, this model assumes the processing time of each block to be independent of the processing load. Thus, this model fails to capture the effect of queuing delays. The bounded delay model is only a suitable approximation for limited capacity when the block rate is much smaller than the capacity, and newly produced blocks typically find the queue empty. This leads to absurd conclusions, such as that the protocol remains secure against a non-zero adversary for arbitrarily high block rates (Fig. 1(a)).

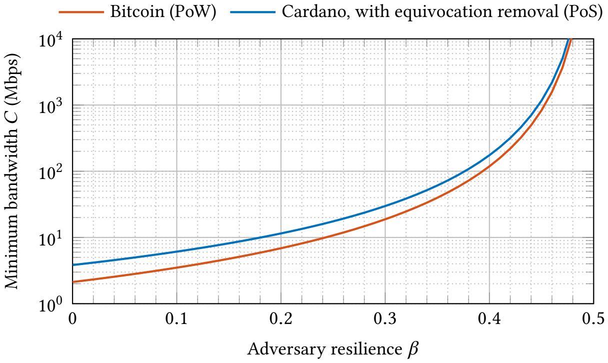


Figure 3: Calculation based on Thms. 1 and 2 of the minimum bandwidth per node that is sufficient to ensure security of LC with the parametrizations used by two major blockchains: Bitcoin (PoW, $\lambda = 1/600$ blocks/s, max. block size 1 MB), and Cardano (PoS, $\tau = 1$ s, $\rho = 1/20$ blocks/slot, max. block size 88 KB). This suggests that to defend against worst-case attacks, Bitcoin might need more per-node bandwidth than commonly recommended (0.4 Mbps [5]).

To study the security of blockchains 'at capacity', we adopt the 'bounded bandwidth' model proposed in [34]. *Thus, henceforth, we adopt the word 'bandwidth', but continue to mean 'capacity' in the wider sense, intending to model nodes' rate-limits across domains such as communication, computation, or storage. We also use 'download' to mean 'process' in the wider sense.* The work [34] analyzed suitable download rules to secure PoS LC where the adversary can spam nodes with equivocating blocks, and waste their download bandwidth. However, their rules and analysis result in an undesirable scaling of the block rate with the desired security parameter, *i.e.*, logarithm of the security error probability (Fig. 1(b)). We solve this by introducing a new variant proof-of-stake protocol that we call SaPoS. To the best of our knowledge, there is no security analysis of PoW LC under bounded bandwidth. It stands to reason that the analysis of [34] carries over to PoW 'in some form', but this analysis' undesirable scaling of the block rate might be too pessimistic for PoW, where the adversary cannot equivocate.

1.1 Our Results

The PoS LC protocols studied in [34] cannot have block rate independent of the security parameter (Fig. 1(b)). We show this with an attack (App. G). This indicates that overcoming the dependence of block rate on security parameter requires not just tighter analysis, but a change to protocol and/or scheduling policy.

PoW LC is secure with block rate independent of the security parameter (Fig. 1(c)). On a high level, bandwidth-related attacks require the adversary to release withheld blocks to distract honest nodes from downloading honestly produced blocks. In PoW, blocks spent for an attack today cannot be spent tomorrow, and vice versa. Thus, the adversary is subject to an overall budget constraint. The analysis of [34] ignores this constraint. Instead, it assumes that at every moment the adversary uses the maximum number of blocks it has available in *any* of its strategies (which is possible in PoS). Thus, [34] replaces the overall worst-case adversary with a fictitious one that acts worst-case *point-wise*. This makes the analysis of [34] overly pessimistic for PoW. We provide a *new analysis technique* (Sec. 1.3.1) that might be of independent interest and with which we can capture the budget constraint of the adversary.

SaPoS, a variant of PoS LC that is secure with block rate independent of the security parameter (Fig. 1(c)). We learn from the PoW result to modify PoS LC to achieve this. Due to equivocations in PoS, the budget constraint of PoW does not readily carry over to PoS. Rather, the adversary can produce many blocks per block production opportunity, and use these blocks to attack at different points in time. In fact, [34] explicitly gives this reasoning for their approach, and our attack in App. G exploits this effect.

To re-introduce the budget constraint of PoW LC into PoS LC, we propose *equivocation removal* (Sec. 1.3.2). Thereby, we preserve the LC protocol's simple blockchain structure, but modify it so that per block production opportunity, honest nodes download at most one of possibly many equivocating blocks. To this end, honest nodes collectively remove the content of equivocating blocks before they reach the ledger of confirmed transactions. We call the PoS LC protocol with this modification *SaPoS*, for *Sanitizing-Proof-of-Stake*. Based on our analysis, we calculate the minimum sufficient bandwidth to secure PoW LC and SaPoS with the parameters of major PoW/PoS blockchain implementations (Fig. 3).

Ensure all transactions have their fee paid. Equivocation removal comes with a drawback: At the time of block production, an honest node might not yet have learned about equivocating blocks in its prefix, and as a result might add transactions to the newly produced block that at execution turn out invalid, due to equivocation removal. This *lack of predictable transaction validity* leads to attacks where the adversary spams the ledger with transactions whose funding source is later invalidated, so no fees can be claimed for the resources they occupy. We present a mechanism (Sec. 1.3.3) to ensure appropriate fees get paid.

1.2 Related Works

Several earlier works have analyzed the security of PoW [18, 32, 13, 35, 26, 38, 19] and PoS [25, 11, 2, 36, 10, 13, 3] LC protocols in the bounded delay model. Our analysis builds on tools from several of these works, primarily pivots [36] (or Nakamoto blocks [13]), and convergence opportunities [35, 36, 26] (or similar [13, 38]).

In the bounded delay model, what is the value of the bound Δ? This is an important question because the parameters of the protocol, such as block production rate, must be tuned according to the delay bound. It is a tricky question because unlike the bandwidth limit, which is a physical limit of the hardware used, delay depends on the network load. One approach is to set the delay to the 'time taken to process one block', *i.e.*, $\Delta = 1/C$. While this may be reasonable at rates much smaller than the bandwidth (as processing queues are mostly empty), queuing delay breaks this bound otherwise. A more conservative approach is to set the delay to be at the tail of the probability distribution of the delay. In theory, given an enqueuing and dequeuing process, it is possible to characterize the distribution of the queuing delay, and this approach is taken in [16]. In practice, the delay distribution can be estimated through network experiments [12, 27]. Another work [39] analyzes security in a random (iid) delay model.

The problem here is that the network load, hence queuing delay, is not purely a random process, but it is controlled by the adversary. This effect is hard to see in experiments. The analysis in [34], although in a bounded bandwidth model, parameterizes the protocol according to a delay bound that holds under the worst-case adversary at all times with overwhelming probability. The above approaches that choose high-probability delay bounds lead to a conservative parameterization where the block rate must decrease as the error probability of the delay bounds decreases (increasing security parameter) as in [34]. As mentioned in Sec. 1.1, our analysis exploits the limited supply of blocks in PoW (and in PoS after equivocation removal) to show that while not all honest blocks may be downloaded in time, at least some of them will be, enough to overcome the adversary's power.

The unsuitability of the bounded delay model at high throughput led to a more careful modeling of limited communication capacities of nodes. Increase in delay due to increase in the block size was pointed out in early Bitcoin discussions [6] and also in experiments with the Bitcoin network [12]. Works [44, 4] model and analyze this effect, but the model still assumes that processing delays are bounded as long as the average network load is below the capacity, thus failing to capture increasing queuing delays while operating near capacity. Increase in delay due to increasing rate of block production was analyzed in [16], capturing the relationship between the bandwidth constraint and queuing delays. Work [34] extends and formalizes the model from [16] to consider adversarial spamming, particularly due to equivocations in proof-of-stake.

Capacity limits apply not only to downloads, but also to processing of blocks. For instance, to validate a block, an Ethereum validator must execute all smart contracts in it. While download and processing are similar in that the time taken increases with the number of transactions, they are different in that processing is hard to parallelize due to transactions that depend on each other. A line of work [14, 40] studies methods to parallelize execution of smart contracts to make use of multi-core architectures.

1.3 Overview of Methods

1.3.1 *New Analysis Technique.*

Traditional LC security analysis (Fig. 4(a)) is based on the notion of a *pivot* [36] (or *Nakamoto block* [13]). A pivot is a point of time in which a block is produced by an honest node (*i.e.*, it includes pending transactions) with an additional property that in every interval around the pivot, there are more honest than adversarial block production opportunities. A probabilistic argument shows that typically pivots happen

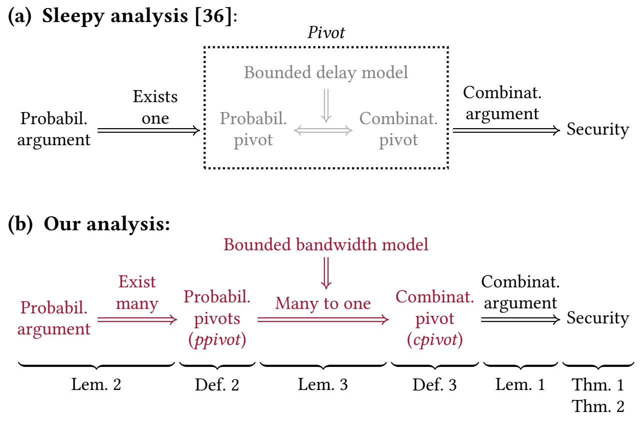


Figure 4: (a) Sleepy analysis [36] is based on *pivots*. Pivots are special honest blocks (cf. liveness) which by a combinatorial argument remain in the chain forever (cf. safety), and by a probabilistic argument happen frequently. The equivalence of the qualities required for the probabilistic/combinatorial argument follows from the bounded delay model. (b) Our new analysis (red) decomposes pivots' probabilistic/combinatorial qualities into *ppivots* and *cpivots*. These are no longer equivalent under bounded bandwidth, but among many consecutive ppivots exists one cpivot. A new probabilistic argument shows the abundance of ppivots.

frequently. A combinatorial argument shows that the pivot block remains in the longest chains of all honest nodes forever. Safety and liveness of LC with suitable parameters follow swiftly.

In the bounded delay network model, the qualities required for the probabilistic and combinatorial argument, respectively, are equivalent. As a result, it has not been widely observed that these properties are actually not identical. In the bounded bandwidth model, these properties are no longer equivalent. Our first conceptual contribution is to decompose pivots' probabilistic/combinatorial qualities into *ppivots* and *cpivots* (Fig. 4(b)). Ppivots are honest block production events where in every time interval around them there are more honest than adversarial block production opportunities (same as pivots in the bounded delay analysis). Cpivots are honest block production events where in every time interval around them there are more *chain growth* events than non-chain-growth events, where chain growth occurs only when an honest block is produced *and downloaded soon* by all honest nodes.

Some ppivots no longer turn into cpivots under bounded bandwidth, because adversarial block release can delay the download of honestly produced blocks, and thus some honest block production opportunities might not translate to chain growth. Our first technical contribution is a combinatorial argument to show that if there is a sufficiently high density of ppivots over a sufficiently long time interval, then one of these ppivots is typically a cpivot. This relies on the adversary's limited budget of blocks it can spam with.

The original probabilistic argument of Sleepy [36] guarantees only a fairly low density of ppivots. Thus, our second technical contribution is to show, using a Chernoff-style tail bound for weakly dependent random processes, that long time intervals typically have a high density of ppivots. This completes the analysis for PoW LC.

1.3.2 *Equivocation Removal.*

To control bandwidth consumption, we stipulate that in PoS, per block production opportunity, every honest node downloads at most one equivocating block. To ensure that honest nodes can still switch from one chain to another longer chain, both of which might contain a different equivocating block from the same block production opportunity, we allow honest nodes to not download, but treat as empty, any block for which they see an equivocation. Note that headers of two equivocating blocks from the same block production opportunity can serve as a *succinct equivocation proof* in that they suffice for honest nodes to convince one another that an equivocation was committed. Therefore, if an honest node that sees an equivocation for a block in its longest chain, publishes an equivocation proof in the block that it produces, nodes can agree on which blocks were equivocated and hence consistently treat them as empty.

A caveat so far is that an adversary could reveal an equivocation late and cause inconsistent ledgers across honest nodes and/or time. To avoid this, we enforce a deadline for how late an equivocation proof can be included in the chain. Our analysis shows how to parameterize the deadline and the LC protocol's confirmation time such that, if any honest node has removed the content of any equivocating block on its longest chain, then an appropriate equivocation proof is timely included on-chain, and all honest nodes remove the block's content before it reaches the output ledger.

1.3.3 *Ensuring Fees Get Paid Despite Lack Of Predictable Validity.*

Equivocation removal leads to *lack of predictable transaction validity*, which risks that the adversary gets to spam the ledger with transactions 'for free'. To ensure that honest block producers only include transactions that pay for their blockspace, we propose to introduce *gas deposit accounts* that can only be used for transaction fees. We also require that any deposit to such an account is not reflected in the balance until the deadline has passed for the inclusion of any equivocation proof that might lead to removal of transactions from the deposit's prefix. This gives honest block producers a lower bound on the account's balance (*e.g.*, more funds than the lower bound might be available if a transaction spending from the gas account gets removed due to an equivocation) which they can use to reliably determine whether a transaction can pay fees. Withdrawals from these accounts can take place immediately.

2 SCHEDULING POLICIES & ATTACKS

Since download and processing resources are constrained, it becomes increasingly important to correctly prioritize the blocks that are downloaded and validated. In this section we describe two possible scheduling policies for nodes running Nakamoto consensus. We show attacks tailored to each such policy and thus show that the choice of policy has a high impact on security. The attacks in this section apply to both PoW and PoS Nakamoto consensus, as the attacks only exploit the block production process that is common to both. The precise setup of the attacks is described in App. B.

In the PoW setting, since headers contain all information needed to verify that enough work has been spent to mine the block, invalid block headers can be ignored, and the attacker is unable to produce blocks without spending computation. Similarly in the PoS setting, the attacker cannot produce blocks for a slot where it is not elected a leader as per the PoS lottery. We restrict ourselves to process only blocks whose parent block is already fully validated. Thus, when we describe the priority of some header block as high, we actually start to process its first unprocessed ancestor.

The Longest-Header-Chain policy. This policy aims to match Nakamoto consensus' confirmation rule. It prioritizes the processing of blocks that are on the longest announced header chain, regardless of which blocks we already have. We assign each unprocessed header a priority h if it is on a header chain of height h.

The Greedy policy. This policy prioritizes downloading blocks that extend the chain a node has already processed. If a header of a block at height h is announced, and we already have h_i blocks from that chain, then we set the priority of the block to be (h_i, h) and compare between the two priorities lexicographically.

2.1 Attacking the Longest-Header-Chain Policy: The Teasing Attack

The Longest-Header-Chain policy seems to be a natural policy when considering the longest-chain protocol. We would like to consider attacks that break the safety or liveness of the chain using as little mining power as possible. The naive attack strategy, that bounded-delay analysis suggests to be worst-case [13], is to have the attacker mine a secret chain of blocks without releasing any blocks to the network. If the attacker is able to outpace the rate of growth of the honest chain, it can publish its blocks at will, and undo all transactions in the blockchain. Since honest nodes take time to process each block, the rate of growth of the chain is slower than the honest nodes' block creation rate and the attacker can more easily succeed in this naive attack if bandwidth is low. This effect is detailed in App. B.1. But can we do better than this attack?

The Teasing Attack. We show that in fact, an adversary can strategically announce headers and release blocks in a way that will exploit the longest-header-chain policy to waste some processing done by the honest nodes. In Fig. 5, we describe the teasing attack that achieves this. We see that this attack utilizes the chain that the attacker constructs not only to later overtake the public chain, but also to induce processing of one extra block for every block that grows the length of the honest chain. It therefore effectively doubles the processing invested per growth event of the public main chain. To entice nodes to process blocks needlessly, the attacker reveals a long header chain from its secret chain, teasing the nodes to download a block from that chain, but makes the rest of the blocks unavailable for download.

To demonstrate the attack, we simulate a network with 100 nodes that together create 1 block per second. The simulations were written as event-driven simulations using Python's simpy package.[1] See App. B for details on the implementation and setup. To start the attack, the attacker must pre-mine a short private chain longer than the honest chain. Even if the attacker's mining rate is lower than the honest mining rate, the attacker still succeeds in doing this with some probability. Thereafter, it starts the teasing attack and the subsequent increased processing delays slow down the honest chain growth rate, which allows the adversary to maintain its lead.

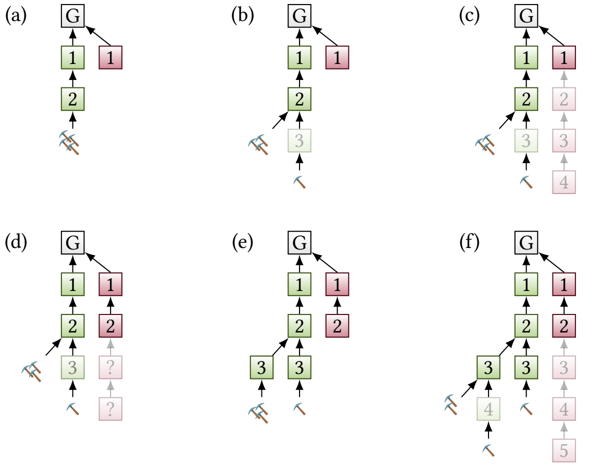


Figure 5: Teasing attack: Green/red are honest/adversarial blocks, and numbers on blocks indicate height in the blockchain. Semi-transparent blocks have been announced (*i.e.*, headers released) but were not yet downloaded by honest nodes. (a) We begin when honest nodes have a chain of length 2. All blocks have been downloaded and validated by honest nodes. (b) An honest node builds a block at height $k = 3$, and announces it. The majority of the nodes are still mining on top of the block at height $k - 1$. The adversary wishes to delay the download of the new block. (c) The adversary announces a block at height $k + 1$ from a chain it had been withholding. Since this is the longest announced chain, honest nodes prioritize its download beginning with the adversary's block 2. (d) Honest nodes have downloaded and validated the adversary's block 2. Since they do not yet have a longer validated chain, they keep mining as before. When they request the adversary's block of height $k = 3$, they find it to be unavailable ('?'), and so ignore the rest of the attacker's chain and resume downloading the honest block of height $k = 3$. (e) While download was delayed, some mining power may have been wasted and another conflicting honest block of height $k = 3$ may have been constructed. Notice now that we are in a scenario similar to the beginning of the attack and that a similar sequence can now repeat. (f) Once an honest node mines a block of height 4, the attacker announces a block at height 5, and proceeds to allow a download of height 3, delaying again the verification of the honest block at height 4. Eventually, the attack breaks safety as the adversary releases content for its chain to overtake the honest chain.

We compare the rate of growth of the honest chain when the attacker mines silently and does not publish blocks (private attack) to the scenario in which a teasing attack (Fig. 5) is being carried out. The chain's rate of growth then sets the bound on the system's security: if the attacker manages to mine faster than the honest network is growing (*i.e.* $\lambda_{\text{adv}} > \lambda_{\text{grwth}}$), then it is able to continue the attack indefinitely.

Fig. 6 depicts the result of our simulation. It shows that with the teasing attack, the network's processing power is slowed roughly by a factor of 2. As a result, the effective delay of block propagation

[1] Source code: https://github.com/avivz/finitebwlc

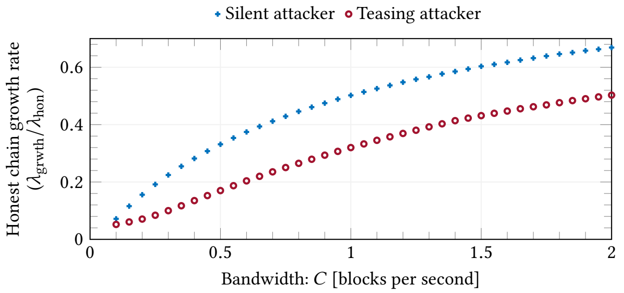


Figure 6: The rate of chain growth relative to honest block production, when nodes prioritize downloads towards the longest known header chain, for various bandwidths. With a teasing attack (○), processing is effectively slowed by a factor of 2, which lowers the growth rate of the chain (and hence lowers security, cf. Fig. 1(c)) compared to a silent attacker (◆).

increases, and the attacker succeeds with greater ease compared to the naive attack. Since the silent attack is the worst-case attack for Nakamoto consensus according to bounded-delay analysis [13], the teasing attack demonstrates that scheduling policies must be taken into account when considering the security of the protocol, and sufficient capacity needs to be provisioned to ensure security.

2.2 The Greedy Policy and the Forking Attack

The teasing attack relied strongly on the fact that the attacker could entice nodes with a long header chain that is later discovered to be unavailable for download. It is natural in this case to consider adjusting the download rule to one that prefers the proverbial 'bird in the hand over two birds in the bush', *i.e.*, to extend the blocks we already downloaded over the illusive promise of a longer chain that the attacker may withhold from us.

While the greedy policy performs well at high processing rates, we unfortunately find that it preforms poorly in the low processing rate regime. Specifically, if a fork in the chain occurs, and nodes are split evenly between the two alternatives, the fork may never resolve. This is because nodes extend their own chain, and prioritize download on their side of the split, while having insufficient processing power to catch up with the other alternative chain. A fork in the chain can result from a deliberate attack by an attacker that releases blocks selectively to different nodes, by a network split, or worse, by an unlucky timing of honest node mining events. In this case, the blockchain fails even for small attackers. Importantly, a fork that never resolves is either a safety or a liveness failure, as no transaction on either side of the split can be safely accepted.

To demonstrate this download rule in action, we simulate a network of 100 nodes that are split evenly between two partitions for only 15 seconds, *i.e.*, for an expected time required to produce 15 blocks.[2]Once the network split ends, the simulation continues for another 4000 seconds, allowing nodes the opportunity to converge on a chain. We measure the height of the latest block all nodes agree upon. If nodes do not recover from the partition, this block will be

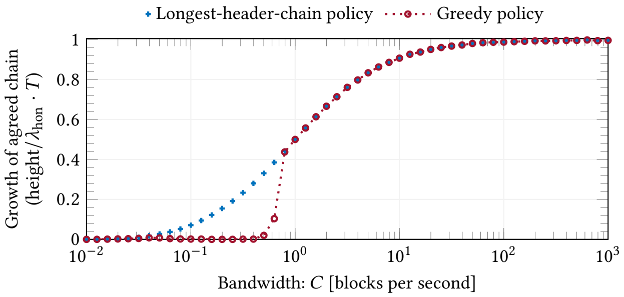


Figure 7: The rate nodes grow the agreed chain after the network splits into two sets of 50 nodes for 15 secs, when the download rule is "longest-header-chain" (◆) or "greedy" (···○···). Nodes using the greedy policy prioritize downloads on their current chain. Under low bandwidth, they do not recover from the split, resulting in two chains forking at genesis, providing no growth of the agreed chain. Thus, longest chain is insecure *without an adversary* (cf. Fig. 1(c)).

the genesis and the liveness of the protocol has failed. Otherwise, nodes quickly agree on the main chain and the height of the latest agreed block is just a little behind the longest tip of the chain.

We simulate the evolution after a brief partition for both the longest-header-chain policy as well as for the greedy policy. Our results (Fig. 7) show that in settings where bandwidth is greater than 1/2, nodes manage to catch up with the chain and the rate of growth matches for both scheduling policies. In lower bandwidth settings, however, nodes never catch up. Note that this attack requires no adversarial mining, yet the protocol is insecure (cf. Fig. 1(c)). This is in stark contrast to the bounded-delay analysis which suggests that the protocol retains security against a non-mining adversary at any bandwidth (cf. Fig. 1(a)), and highlights again the need to study the security of blockchains at capacity.

3 PROTOCOL & MODEL

Pseudocode of an idealized LC protocol $\Pi^{\rho,\tau,k_{\text{conf}}}$ is provided in Alg. 1. Details of the protocol's resource-based block production lottery, *i.e.*, of production and verification of blocks, are abstracted through an idealized functionality $\mathcal{F}_{\text{hdrtree}}$ (cf. [36, Fig. 2], [34, Alg. 3]). Pseudocode for instantiations $\mathcal{F}_{\text{hdrtree}}^{\text{PoW},\rho}$ and $\mathcal{F}_{\text{hdrtree}}^{\text{PoS},\rho}$ modeling proof-of-work (PoW) and proof-of-stake (PoS) are provided in Alg. 2 and Alg. 3, respectively. Helper functions used in the pseudocode are detailed in App. A.1. We study these protocols in a unified model for a network $\mathcal{Z}$ with finite bandwidth (Fig. 8), and for the powers and limits of an adversary $\mathcal{A}$.

3.1 Longest Chain Protocols

For ease of exposition, the execution features a *static* set of N *equipotent nodes*, each of which runs an independent instance of $\Pi^{\rho,\tau,k_{\text{conf}}}$. Temporary crash faults ('sleepiness') of nodes (in PoW and PoS), heterogeneous distribution of hash power (in PoW) or stake (in PoS), and stake shift (in PoS) or difficulty adjustment (in PoW), are left to be addressed with techniques from [11, 10, 36, 17]. We are interested in the large system regime $N \longrightarrow \infty$. Nodes interact with each other and with the adversary $\mathcal{A}$ through an

[2] Such short splits are relatively easy to induce in reality (transient problems with Internet routing, denial-of-service on the network, etc.) and thus a practical scheduling rule must recover from such splits.

Algorithm 1 Idealized LC consensus protocol $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ with download logic (helper functions: App. A.1, environment $\mathcal{Z}$: App. A.2, functionality $\mathcal{F}_{\mathrm{hdrtree}}$: Alg. 2 for PoW, Alg. 3 for PoS)

```
 1: ▷ Global counter of slots $t \leftarrow 1, 2, \ldots$ of duration $\tau$ (for PoW: $\tau \rightarrow 0$, cf. Sec. 5)
 2: on INIT($\mathrm{genesis}C, \mathrm{genesisTxs}$)
 3:     ▷ Initialize header tree $\mathrm{h}\mathcal{T}$, longest downloaded chain $\mathrm{d}C$, and mappings from block header to content blkTxs
 4:     $\mathrm{h}\mathcal{T}, \mathrm{d}C \leftarrow \{\mathrm{genesis}C\}, \mathrm{genesis}C$
 5:     $\mathrm{blkTxs}[\mathrm{genesis}C] \leftarrow \mathrm{genesisTxs}$        ▷ Unset entries of blkTxs are UNKNOWN
 6: on RECEIVEDHEADERCHAIN($C$)                                 ▷ Called by $\mathcal{Z}$ or $\mathcal{A}$
 7:     assert $\mathcal{F}_{\mathrm{hdrtree}}$.VERIFY($C$)                       ▷ Validate header chain
 8:     $\mathrm{h}\mathcal{T} \leftarrow \mathrm{h}\mathcal{T} \cup \mathrm{prefixChainsOf}(C)$          ▷ Add $C$ and its prefixes to $\mathrm{h}\mathcal{T}$
 9:     $\mathcal{Z}$.BROADCASTHEADERCHAIN($C$)
10: on RECEIVEDCONTENT($C, \mathrm{txs}$)                            ▷ Called by $\mathcal{Z}$ or $\mathcal{A}$
11:     ▷ Defer processing the content until all prefixes' contents are downloaded
12:     defer until $\forall C' \prec C\colon \mathrm{blkTxs}[C'] \neq$ UNKNOWN
13:     assert $C.\mathrm{txsHash} = \mathrm{Hash}(\mathrm{txs})$
14:     RECEIVEDHEADERCHAIN($C$)                              ▷ Validate header chain
15:     $\mathrm{blkTxs}[C] \leftarrow \mathrm{txs}$
16:     $\mathcal{Z}$.UPLOADCONTENT($C, \mathrm{txs}$)
17:     ▷ Update the longest downloaded chain among downloaded chains
18:     $\mathcal{T}' \leftarrow \{C' \in \mathrm{h}\mathcal{T} \mid \mathrm{blkTxs}[C'] \neq$ UNKNOWN$\}$
19:     $\mathrm{d}C \leftarrow \arg\max_{C' \in \mathcal{T}'} |C'|$
20: at slot $t \leftarrow 1, 2, \ldots$                                  ▷ LC protocol main loop
21:     $\mathrm{txs} \leftarrow \mathcal{Z}$.RECEIVEPENDINGTXS()
22:     ▷ Produce and disseminate a new block if eligible
23:     if $C' \neq \bot$ with $C' \leftarrow \mathcal{F}_{\mathrm{hdrtree}}$.EXTEND($\mathrm{d}C, \mathrm{txs}$)
24:         $\mathcal{Z}$.BROADCASTHEADERCHAIN($C'$)
25:         $\mathcal{Z}$.UPLOADCONTENT($C', \mathrm{txs}$)
26:     ▷ Confirm all but the last $k_{\mathrm{conf}}$ blocks on the longest downloaded chain
27:     $\mathsf{LOG}^t \leftarrow \mathrm{txsLedger}(\mathrm{blkTxs}, C^{\lceil k_{\mathrm{conf}}})$   ▷ Ledger of node $p$ at time $t$: $\mathsf{LOG}_p^t$
28: ▷ Throughout, download content for some $C$ chosen by download rule (e.g. Alg. 4)
```

Algorithm 2 Idealized functionality $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoW},\rho}$: block production lottery and header chain structure for PoW (helper functions: App. A.1)

```
 1: on INIT($\mathrm{genesis}C, \mathrm{numNodes}$)
 2:     $N \leftarrow \mathrm{numNodes}$
 3:     $\mathcal{T} \leftarrow \{\mathrm{genesis}C\}$                          ▷ Global set of valid header chains
 4: on EXTEND($C, \mathrm{txs}$) from node $P$ (possibly adversarial) at slot $t$
 5:     ▷ Abstraction of proof-of-work lottery: each node can call this once per slot and produces a block with probability $\rho/N$ independently of other nodes and slots
 6:     if $\mathrm{lottery}[P, t] \neq \bot$ return $\bot$   ▷ allow only one block per successful lottery
 7:     $\mathrm{lottery}[P, t] \xleftarrow{\$}$ (true with probability $\rho/N$, else false)
 8:     if $C \in \mathcal{T} \wedge \mathrm{lottery}[P, t]$   ▷ New header chain valid if parent chain $C$ is valid
 9:         ▷ Produce a new block header extending $C$
10:         $C' \leftarrow C \| \mathrm{newBlock}(\mathrm{txsHash}\colon \mathrm{Hash}(\mathrm{txs}))$
11:         $\mathcal{T} \leftarrow \mathcal{T} \cup \{C'\}$              ▷ Register new header chain in header tree
12:         return $C'$
13:     return $\bot$
14: on VERIFY($C$)
15:     return $C \in \mathcal{T}$      ▷ Header chain is valid if previously added to header tree
```

Algorithm 3 Idealized functionality $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$: block production lottery and header chain structure for PoS (helper functions: App. A.1)

```
 1: ▷ INIT($\mathrm{genesis}C, \mathrm{numNodes}$) and VERIFY($C$) same as in Alg. 2
 2: on ISLEADER($P, t$) from $\mathcal{A}$ (only for adversarial node $P$) or $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$
 3:     ▷ Abstraction of proof-of-stake lottery: each node is chosen leader in each slot with probability $\rho/N$ independently of other nodes and slots
 4:     if $\mathrm{lottery}[P, t] = \bot$
 5:         $\mathrm{lottery}[P, t] \xleftarrow{\$}$ (true with probability $\rho/N$, else false)
 6:     return $\mathrm{lottery}[P, t]$
 7: on EXTEND($t', C, \mathrm{txs}$) from $\mathcal{A}$ (only for adversarial node $P$) or $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$
 8:     ▷ New header chain is valid if parent chain $C$ is valid, $P$ is leader for slot $t'$, and $t'$ is later than the tip of $C$ and is not in the future
 9:     if $(C \in \mathcal{T}) \wedge \mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$.ISLEADER($P, t'$) $\wedge (C.\mathrm{time} < t' \leq t)$
10:         ▷ Produce a new block header extending $C$
11:         $C' \leftarrow C \| \mathrm{newBlock}(\mathrm{time}\colon t', \mathrm{node}\colon P, \mathrm{txsHash}\colon \mathrm{Hash}(\mathrm{txs}))$
12:         $\mathcal{T} \leftarrow \mathcal{T} \cup \{C'\}$              ▷ Register new header chain in header tree
13:         return $C'$
14:     return $\bot$
15: on EXTEND($C, \mathrm{txs}$) from node $P$ (possibly adversarial) at slot $t$
16:     return $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$.EXTEND($t, C, \mathrm{txs}$)
```

Algorithm 4 'Download longest header chain' rule $\mathcal{D}_{\mathsf{long}}$

```
 1: function dlLongestHdrChain($\mathrm{h}\mathcal{T}, \mathrm{blkTxs}$)
 2:     $\mathcal{T}' \leftarrow \{C \in \mathcal{T}' \mid \mathrm{blkTxs}[C] =$ UNKNOWN$\}$        ▷ Ignore downloaded chains
 3:     $C \leftarrow \arg\max_{C' \in \mathcal{T}'} |C'|$                         ▷ Select the longest chain
 4:     $C' \leftarrow \arg\min_{C'' \preceq C\colon \mathrm{blkTxs}[C''] = \mathsf{UNKNOWN}} |C''|$   ▷ First unknown block on that chain (if non-existent: $\bot$)
 5:     return $C'$
```

environment $\mathcal{Z}$ that models the network and is detailed in Secs. 3.2 and A.2. The protocol proceeds in *slots* of duration τ (Alg. 1, l. 20). At each slot t, the protocol queries the block production lottery $\mathcal{F}_{\mathrm{hdrtree}}$ in an attempt to extend the longest downloaded chain $\mathrm{d}C$ in the node's view with a new block of pending transactions txs. If successful, the node disseminates both the resulting *block header* C' and the associated *block content* txs via the environment $\mathcal{Z}$ to all nodes. Finally, the protocol identifies the k_{conf}-deep prefix $\mathrm{d}C^{\lceil k_{\mathrm{conf}}}$ containing all but the last k_{conf} blocks of $\mathrm{d}C$. The transactions along $\mathrm{d}C^{\lceil k_{\mathrm{conf}}}$ are concatenated to produce the output ledger LOG^t.

When a node p receives a new valid block header C (Alg. 1, l. 6), p adds C to its header tree $\mathrm{h}\mathcal{T}$, records C as first seen at the current slot, and relays C to all other nodes via $\mathcal{Z}$. Throughout the execution, the protocol requests from $\mathcal{Z}$ the content for block headers decided by a download priority rule (Alg. 1, l. 28). As a concrete example, we use the 'download longest header chain' rule (Alg. 4) in which a node downloads content for the first block header with unknown content on the longest header chain it has seen. Once a valid block's content is received (Alg. 1, l. 10), the node makes it available to other nodes via $\mathcal{Z}$, and updates its $\mathrm{d}C$.

Proof-of-Work. The characteristics of PoW-based block production, *e.g.*, in Bitcoin [32, 18], are captured by the idealized functionality $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoW},\rho}$ (Alg. 2). Each block production attempt is committed to a parent block and block content (Alg. 2, l. 4), and only a single block is produced when the attempt is successful. Per slot, each node can make one block production attempt that will be successful with probability ρ/N, independently of other nodes and slots (Alg. 2, l. 7). This model, for ease of exposition, assumes uniform hash power across all nodes. Since each slot represents a single PoW evaluation, we study PoW in the regime $\rho = \Theta(\tau)$, $\tau \rightarrow 0$. In turn as $\rho \rightarrow 0$, with probability 1, each slot produces at most one block across all nodes. The PoW model thus implies that every block must be produced in a slot *strictly after* its parent block.

Proof-of-Stake. PoS LC protocols such as from the Ouroboros [25, 11, 2] or Sleepy Consensus [36, 10] families can be modeled using $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}$ (Alg. 3). As in PoW, each node can make one block production attempt per slot that will be successful with probability ρ/N, independently of other nodes and slots (Alg. 3, l. 5)[3], modeling uniform stake. In PoS, however, (even past) block production opportunities can be 'reused' to produce multiple blocks with dif-

[3] There may be multiple blocks in one slot, as in the Ouroboros [25, 11, 2] and Sleepy Consensus [36, 10] protocols.

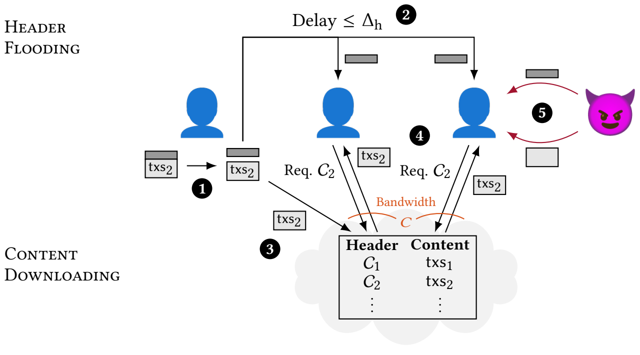


Figure 8: Bandwidth constrained network model [34, Fig. 4]: ❶ Honest node produces a block, made of header and content. A hash in the header commits to the content. ❷ Header is flooded ($\mathcal{Z}.\textsc{broadcastHeaderChain}$), and arrives at all nodes ($\Pi^{\rho,\tau,k_{\mathrm{conf}}}.\textsc{receivedHeaderChain}$) with at most Δ_{h} delay. ❸ Content is made available for peer-to-peer pull-based download ($\mathcal{Z}.\textsc{uploadContent}$). ❹ Content associated with the header is downloaded ($\Pi^{\rho,\tau,k_{\mathrm{conf}}}.\textsc{receivedContent}$), subject to a maximum rate of C. ❺ The adversary can push headers and content to nodes, bypassing the delay and bandwidth constraints.

ferent parents and/or content, *i.e.*, to equivocate (Alg. 3, ll. 2, 7). The regime of interest is $\tau = \Theta(1)$.

3.2 Bandwidth Constrained Network

We borrow the bandwidth constrained network model of [34] (Fig. 8). In this model, $\mathcal{Z}$ abstracts push-based flooding of 'small' block headers and pull-based downloading of 'large' block contents from peers. Block header chains sent via $\mathcal{Z}.\textsc{broadcastHeaderChain}$ are eventually delivered by $\mathcal{Z}$ to every node, cf. Alg. 1, l. 6. Headers are delivered with a per-node per-header delay determined by $\mathcal{A}$, up to a commonly known delay upper bound Δ_{h}. Block content made available via $\mathcal{Z}.\textsc{uploadContent}$ is kept by $\mathcal{Z}$ in what can be thought of as a 'cloud'. Nodes can request the content associated with a particular header. If content matching the header is available, then it is delivered by $\mathcal{Z}$ to the node, cf. Alg. 1, l. 10. Content download is subject to a per-node bandwidth constraint of C. See App. A.2 for a more formal description of $\mathcal{Z}$.

The 'cloud' captures key properties of pull-based peer-to-peer downloading. At first, content matching a particular header might not be available (*e.g.*, $\mathcal{A}$ produced a block and disseminated its header, but withheld its content). Later, such content can become available (*e.g.*, $\mathcal{A}$ releases the content to one honest node). Thus, the 'cloud' ensures neither data availability nor strong consistency of query outcomes, unlike stronger primitives such as verifiable information dispersal [7, 22, 46, 33]. However, once content for a header does become available, it is unique and remains available. This captures the header's binding commitment to the content, and the fact that honest nodes share content with peers. Requests for unavailable content do not count towards the download budget. Also note that the adversary can push headers and content bypassing bandwidth and delay constraints, and this models non-uniform bandwidth across nodes, and additional effects (analogous to adversarially controlled delay up to maximum Δ in the bounded delay model).

Powers and Limits of the Adversary. The *static* adversary $\mathcal{A}$ chooses a set of nodes (up to a fraction β of all N nodes, where β is common knowledge) to corrupt before the randomness of the execution is drawn and the execution commences. Uncorrupted *honest* nodes follow $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ at all times. Corrupted *adversarial* nodes follow arbitrary computationally bounded *Byzantine* behavior, coordinated by $\mathcal{A}$ in an attempt to break consensus. Among other things, the adversary can: withhold block headers and content, or release them late or selectively to honest nodes; push headers and content to nodes while bypassing the delay and bandwidth constraints; break ties in $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$'s chain selection and content download policy; in PoS, reuse block production opportunities to produce multiple blocks (*equivocations*, cf. $\mathcal{F}_{\mathrm{hdrtree}}^{\mathrm{PoS},\rho}.\textsc{extend}$), and extend chains using past opportunities as long as the purported block production slots along every chain remain strictly increasing.

3.3 Security of Ledger Protocols

For an execution of $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ where every honest node p at every slot t outputs a ledger LOG_p^t, we recall the security desiderata:

- *Safety:* For all adversarial strategies, for all slots t, t', and for all honest nodes p, q: $\mathsf{LOG}_p^t \preceq \mathsf{LOG}_q^{t'}$ or $\mathsf{LOG}_q^{t'} \preceq \mathsf{LOG}_p^t$.
- *Liveness with parameter* T_{live}*:* For all adversarial strategies, if a transaction tx is received by all honest nodes by slot t, then for every honest node p and for all slots $t' \geq t + T_{\mathrm{live}}$, $\mathrm{tx} \in \mathsf{LOG}_p^{t'}$.

A consensus protocol is *secure over slot horizon* T_{hrzn} *with parameter* T_{live} iff it satisfies safety, and liveness with parameter T_{live}, with overwhelming probability over executions of slot horizon T_{hrzn}.

3.4 Notation

Nodes are identified using p, q. Our notation distinguishes between three notions of 'time': *Slots* of $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ are indicated by r, s, t. Slots in which one or more blocks are produced form a sub-sequence $\{t_k\}$, defined in Sec. 4.2. *Indices* into this sub-sequence are denoted by i, j, k. Physical parameters of our model, header propagation delay Δ_{h} and bandwidth C, are specified in units of *real time*.

We denote intervals of indices (or slots) as $(i, j] \triangleq \{i+1, ..., j\}$, with the convention that $(i, j] \triangleq \emptyset$ for $j \leq i$. We study executions over a finite horizon of T_{hrzn} slots (or K_{hrzn} indices), and any interval $(i, j]$ with $i < 0$ or $j > K_{\mathrm{hrzn}}$ considered truncated accordingly. The notation $(i, j] \succ K$ (resp. $\succeq, \prec, \preceq, \asymp$) is short for $j - i > K$ (resp. $\geq, <, \leq, =$). In the analysis, we denote with upper-case Latin letters several random processes over indices (*e.g.*, X_k) or slots (*e.g.*, H_t). For any set I of indices (analogously for slots), we define $X_I \triangleq \sum_{k \in I} X_k$.

We denote by κ the security parameter. An event $\mathcal{E}_\kappa$ occurs *with overwhelming probability* (*wop*) if $\Pr[\mathcal{E}_\kappa] \geq 1 - \mathrm{negl}(\kappa)$. Here, a function $f(\kappa)$ is *negligible* $\mathrm{negl}(\kappa)$, if for all $n > 0$, there exists κ_n^* such that for all $\kappa > \kappa_n^*$, $f(\kappa) < \frac{1}{\kappa^n}$.

4 SECURITY ANALYSIS

4.1 Unified Model for PoW and PoS

We develop a unified probabilistic model for the block production of both PoW and PoS as per Algs. 2 and 3. This enables us to prove properties of the block production process and block tree structure that are common to both variants (Sec. 4.3). We then use these properties to prove security of PoW LC (Sec. 5) and PoS LC (Sec. 6).

Recall that the protocol runs in discrete units of time called slots, and that we consider $\tau \longrightarrow 0$ to model PoW. A *block production opportunity* (BPO) is a pair (p, t) where according to the PoW/PoS block production lottery, node p is eligible to produce a block in slot t. A BPO is called *honest* (resp. *adversarial*) if node p is honest (resp. adversarial). The random variables H_t and A_t denote the number of honest and adversarial BPOs in slot t, respectively. When the number of nodes $N \longrightarrow \infty$ and each node holds an equal rate of block production, by the Poisson approximation of a binomial random variable, we have $H_t \overset{\text{i.i.d.}}{\sim} \text{Poisson}((1-\beta)\rho)$ and $A_t \overset{\text{i.i.d.}}{\sim} \text{Poisson}(\beta\rho)$, independent of each other and across slots. The total number of BPOs per slot is $Q_t = H_t + A_t$. An *execution* refers to a particular realization of the random process $\{(H_t, A_t)\}$.

In PoW, as we take $\tau \longrightarrow 0$, the block production process converges to a *Poisson point process*. As noted in Sec. 3.1, each BPO corresponds to a different slot, and in both PoW and PoS we may assume that blocks in one chain must come from increasing slots.

In this unified model, we make the adversary's powers the strongest of both PoW and PoS. Specifically, we allow the adversary to create multiple blocks from the same BPO (equivocations) which is only possible in PoS but not in PoW. However, we assume in the unified analysis that honest nodes use a download rule which downloads at most one block per BPO. From a bandwidth perspective, this puts both PoW and PoS on an equal footing. Then as seen in [13, 19], the additional ability to equivocate does not change the block tree properties and therefore allows us to use similar techniques in our unified analysis. The assumption of downloading at most one block per BPO clearly holds for any download rule in PoW, but we define an equivocation removal policy to achieve this in PoS, so that the unified model applies to PoS as well.

4.2 Definitions

'Good' slots are slots with exactly one honest BPO and no adversarial BPOs in that slot, and no BPOs in ν slots after. This definition is inspired by convergence opportunities [35, 36, 26], loners [13], and laggers [38]. Here, ν is an analysis parameter whose value is chosen such that each honest node can receive the block header from the honest BPO, and download content for $\widetilde{C}$ within $\nu + 1$ slots. That is,

$$(\nu + 1)\tau \triangleq \Delta_\text{h} + \widetilde{C}/C. \tag{1}$$

Definition 1. We call a slot t *good*, *bad*, *empty*, respectively, denoted as $\text{Good}(t)$, $\text{Bad}(t)$, $\text{Empty}(t)$, respectively, iff:

$$\text{Good}(t) \triangleq (H_t = 1) \wedge (A_t = 0) \wedge (H_{(t,t+\nu]} + A_{(t,t+\nu]} = 0) \tag{2}$$

$$\text{Bad}(t) \triangleq (H_t + A_t > 0) \wedge \neg\text{Good}(t) \tag{3}$$

$$\text{Empty}(t) \triangleq (H_t + A_t = 0). \tag{4}$$

Note that $\text{Empty}(t) = \neg\text{Good}(t) \wedge \neg\text{Bad}(t)$. We denote by t_k the k-th non-empty slot. Then, we can introduce random processes over *indices*, with index k corresponding to the k-th non-empty slot t_k. The process $\{G_k\}$ counts good slots, with $G_k \triangleq 1$ if $\text{Good}(t_k)$, and $G_k \triangleq 0$ otherwise (*i.e.*, if $\text{Bad}(t_k)$). Correspondingly, $\{\overline{G}_k\}$ counts bad slots, $\overline{G}_k \triangleq 1 - G_k$.

Proposition 1. *The random variables $\{G_k\}$ are independent and identically distributed (iid) with*

$$\Pr[G_k = 1] \triangleq p_\text{G} = (1-\beta)\frac{\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}}. \tag{5}$$

Proof is in App. C. Throughout the analysis, we will assume that $p_\text{G} = \frac{1}{2} + \varepsilon_\text{G}$ with $\varepsilon_\text{G} \in (0, 1/2]$ ('honest majority' assumption).

A special role is played by good slots t_k as these are candidate slots in which the block produced at t_k is 'soon' downloaded by all honest nodes. We count these slots with $\{D_k\}$, and all other non-empty slots with $\{\overline{D}_k\}$. Specifically, $D_k \triangleq 1$ if $\text{Good}(t_k)$ and the block produced at t_k has been downloaded by all honest nodes by the end of slot $t_k + \nu$, $D_k \triangleq 0$ otherwise, and $\overline{D}_k \triangleq 1 - D_k$. We call slots k with $D_k = 1$ as D-slots and those with $\overline{D}_k = 1$ as $\overline{D}$-slots.

Finally, we define two random walks on indices of non-empty slots with increments $\{X_k\}$ and $\{Y_k\}$ that will come in handy for the definition of probabilistic and combinatorial pivots:

$$X_k \triangleq G_k - \overline{G}_k \qquad Y_k \triangleq D_k - \overline{D}_k \tag{6}$$

Note that the increments $\{X_k\}$ are iid, and not affected by adversarial action, while the increments $\{Y_k\}$ *do depend* on the adversarial action and are thus in particular *not* iid. Also note that $\forall k\colon Y_k \leq X_k$ since $D_k = 1 \implies G_k = 1$.

Definition 2. We call an index k a *ppivot* (short for *probabilistic pivot*), denoted as $\text{PPivot}(k)$, iff:

$$\text{PPivot}(k) \triangleq (\forall\, (i,j] \ni k\colon X_{(0,i]} < X_{(0,k]} \leq X_{(0,j]}) \tag{7}$$

This definition of ppivots captures the *probabilistic* aspects of [36, Def. 5] used in [36, Sec. 5.6.3] and casts them as conditions on a random walk, inspired by [13, 29], to simplify the analysis.

These alternative characterizations of ppivots are insightful:

Proposition 2.

$$\begin{aligned} \text{PPivot}(k) &\iff (\forall\, (i,j] \ni k\colon X_{(i,j]} > 0) && (8)\\ &\iff (\forall\, (i,j] \ni k\colon G_{(i,j]} > \overline{G}_{(i,j]}) && (9)\\ &\iff (X_k = 1) \wedge (\forall j \geq k : X_{(k,j]} \geq 0)\\ &\qquad \wedge (\forall i < (k-1) : X_{(i,k-1]} \geq 0) && (10) \end{aligned}$$

Proof. Elementary, using $X_{(i,j]} = X_{(0,j]} - X_{(0,i]}$. □

In particular, eqn. (10) characterizes a ppivot as an index k such that $G_k = 1$ and the simple random walks $\ell \mapsto X_{(k,k+\ell]}$ and $\ell \mapsto X_{(k-1-\ell,k-1]}$ starting at 0 remain non-negative forever (Fig. 9). The process $\{P_k\}$ counts ppivots, with increments $P_k \triangleq \mathbb{1}_{\{\text{PPivot}(k)\}}$.

Definition 3. We call an index k a *cpivot* (short for *combinatorial pivot*), denoted as $\text{CPivot}(k)$, iff:

$$\text{CPivot}(k) \triangleq (\forall\, (i,j] \ni k\colon Y_{(0,i]} < Y_{(0,k]} \leq Y_{(0,j]}) \tag{11}$$

This definition of cpivots captures the *combinatorial* aspects of [36, Def. 5] used in [36, Sec. 5.6.2] and casts them as conditions on a random walk, inspired by [13], to simplify the analysis. The equivalences of Prop. 2 hold for cpivots analogously. Note that a

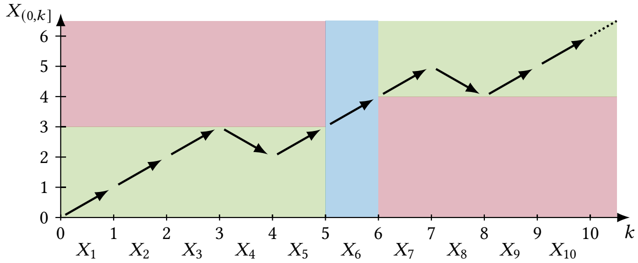


Figure 9: Illustration of ppivot (eqn. (10)): A ppivot as an index k so that $X_k = 1$ (🟦) and $X_{(0,.]}$ is strictly below $X_{(0,k]}$ left of k and weakly above $X_{(0,k]}$ right of k (🟩), elsewhere (🟥).

cpivot is also a ppivot because $Y_i \leq X_i$.

We denote by $\mathrm{d}\mathcal{C}_p(t)$ the longest fully downloaded chain of an honest node p at the end of slot t, and let $|b|$ denote the height of a block b. We use the same notation $|\mathcal{C}|$ to denote the length of a chain $\mathcal{C}$, define $L_p(t) = |\mathrm{d}\mathcal{C}_p(t)|$ and $L_{\min}(t) = \min_p L_p(t)$.

4.3 Unified Analysis in the Probabilistic Model

In this section, we develop all the tools needed to prove the safety and liveness of the PoW and PoS longest chain protocols.

In Sec. 4.3.1 we show that a block produced in a slot corresponding to a cpivot stabilizes, *i.e.*, remains in the longest downloaded chain of all honest nodes. This is useful because if transactions in a block are confirmed after waiting long enough so at least one cpivot occurs, the prefix of the cpivot stabilizes and so those transactions remain in every honest node's ledger (safety). The occurrence of cpivots also guarantees liveness because the block from a cpivot is honest, so it adds new valid transactions to the ledger.

Further, we show that ppivots occur very often (Sec. 4.3.2) and the adversary cannot prevent all ppivots from becoming cpivots (Sec. 4.3.3). Thus, at least one cpivot occurs in a long enough time interval, the length of which can be set as the confirmation time.

4.3.1 Combinatorial Pivots Stabilize. In this section, we show that the honest block produced in a slot corresponding to a cpivot persists in the longest downloaded chain of all honest nodes after ν slots. Towards this, we first show that if $D_k = 1$, *i.e.*, if all honest nodes download the block produced in the good slot t_k, then the length of the longest downloaded chain of honest nodes increases (made precise in Prop. 3). Due to this, since all intervals around a cpivot contain more indices with $D_k = 1$ than those with $D_k = 0$, there can never be a chain which is longer than an honest node's longest downloaded chain and does not contain the block corresponding to the cpivot (Lem. 1). In turn, this means that the block corresponding to the cpivot remains in all honest nodes' longest downloaded chains forever. Lem. 1 is proved in App. C.1.

Proposition 3. *If $D_k = 1$, then $L_{\min}(t_k + \nu) \geq L_{\min}(t_k - 1) + 1$.*

Proof. Since $D_k = 1$, slot t_k is a good slot. Let b be the unique honest block produced in slot t_k, and let honest node p be its producer. Since honest nodes produce blocks on their longest downloaded chain, $|b| = L_p(t_k - 1) + 1 \geq L_{\min}(t_k - 1) + 1$. Further, $D_k = 1$ means that the block b is downloaded by all honest nodes by the end of slot $t_k + \nu$. Therefore, $L_{\min}(t_k + \nu) \geq |b|$. □

Lemma 1. *Let b^* be the block produced in a non-empty slot t_k such that* CPivot(k)*. Then for all header chains $\mathcal{C}'$ that are valid at slot $t \geq t_k + \nu$ and $|\mathcal{C}'| \geq L_{\min}(t)$, $b^* \in \mathcal{C}'$. Further, for all honest nodes p and for all slots $t \geq t_k + \nu$, $b^* \in \mathrm{d}\mathcal{C}_p(t)$.*

4.3.2 Probabilistic Pivots Are Abundant. Sufficiently long intervals of indices contain a number of ppivots proportional to the interval length. Recall that throughout, $p_{\mathrm{G}} = \frac{1}{2} + \varepsilon_{\mathrm{G}}$ with $\varepsilon_{\mathrm{G}} \in (0, 1/2]$.

Lemma 2. *For $K_{\mathrm{cp}} = \Omega(\kappa^2)$, and $K_{\mathrm{hrzn}} = \mathrm{poly}(\kappa)$,*

$$\begin{aligned}\Pr\left[\forall\, (i,j] \geq K_{\mathrm{cp}}\colon P_{(i,j]} \geq (1-\delta) p_{\mathrm{ppivot}} K_{\mathrm{cp}}\right] \\ \geq 1 - \exp(-\Omega(\kappa)) = 1 - \mathrm{negl}(\kappa). \qquad (12)\end{aligned}$$

The proof is in App. C.2.

4.3.3 Many Probabilistic Pivots Imply One Combinatorial Pivot. As a concrete choice of download rule, we consider the 'download longest header chain' rule $\mathcal{D}_{\mathrm{long}}$ (Alg. 4). This rule has some useful properties that we prove below. Intuitively, nodes using this rule

(P1) do not download the same block twice,
(P2) download at most one block from each BPO (in PoW),
(P3) either download the most recent honest block, or fully utilize their bandwidth to download other blocks (don't stay idle), and
(P4) download only blocks that were produced 'recently'.

(P1) clearly holds as this rule only downloads content for headers whose content is yet `UNKNOWN`, hence was not downloaded before. (P2) holds in PoW because there is only one block per BPO. In PoS, the download rule is modified to satisfy this property (Sec. 6). (P3) holds because the download rule $\mathcal{D}_{\mathrm{long}}$ is never idle, and will always download towards an honest block when it has downloaded all longer chains and there is bandwidth remaining. Moreover, we expect that under a secure execution, (P4) holds because the longest header chain can not fork off too much from the longest downloaded chain of an honest node, otherwise it would cause a safety violation. More precisely, due to Lem. 1, any longest header chain in any honest node's view must extend the block produced in the most recent cpivot, and therefore blocks with higher download priority must have been produced after the most recent cpivot (Prop. 4).

Given the above properties of the download rule, we now want to show that cpivots occur often. To start with, let us show that there is at least one cpivot in $(0, K_{\mathrm{cp}}]$. From Lem. 2, there are many ppivots in $(0, K_{\mathrm{cp}}]$. If there were no cpivots in $(0, K_{\mathrm{cp}}]$, then the adversary must prevent each ppivot from turning into a cpivot. We know that in any interval around a ppivot, there are more good indices than bad indices, in fact good indices outnumber bad indices by a margin that increases linearly with the size of the interval. Therefore, for a ppivot to not be a cpivot, the adversary must prevent an honest node from downloading the most recent honest block in several of these good slots. From Prop. 4, for each such index, the adversary must 'spend' at least $\widetilde{C}$ blocks that the honest node downloads. These blocks must come from a 'budget' that can at most contain all blocks mined since the beginning of the protocol. However, these intervals around the ppivots begin no later than slot K_{cp}, so the number of such blocks is limited. If this 'budget' falls short of the number of blocks required to overthrow all cpivots, then there must be at least one cpivot in $(0, K_{\mathrm{cp}}]$.

Next, we would like to show that there is at least one cpivot in $(mK_{\mathrm{cp}}, (m+1)K_{\mathrm{cp}}]$ for all $m \geq 0$ (where we just saw the base case $m = 0$) Here, the adversary might save up many blocks from the past and attempt to make honest nodes download these blocks at a particular target slot t_k. This is where the property of the download rule proven in Prop. 4 becomes useful. Given that one cpivot occurred in $\big((m-1)K_{\text{cp}}, mK_{\text{cp}}\big]$, Prop. 4 ensures that honest nodes will only download blocks that are produced after $(m-1)K_{\text{cp}}$. This allows us to bound the 'budget' of blocks that the adversary can use to overthrow cpivots, and therefore show that there is at least one cpivot in $\big(mK_{\text{cp}}, (m+1)K_{\text{cp}}\big]$. We thus prove the required claim inductively in Lem. 3.

While all the analysis below is done for the download rule $\mathcal{D}_{\text{long}}$, the proofs only use the properties (P1), (P2), (P3), (P4) and thus apply to several other simple download rules. A few examples are i) "download towards the freshest block" [34], ii) "download only blocks that are consistent with the node's confirmed chain", or iii) "at slot t_k, only download blocks produced in slots $(t_k - T_{\text{dl}}, t_k]$" for some T_{dl}. In fact, iii) gives an alternative definition of the property (P4) instead of the one in Prop. 4. In this work, we did not adopt i) because 'freshness' cannot be determined in PoW, and ii) and iii) because they would fail to recover from a network split (as demonstrated in the forking attack in Sec. 2.2). In Sec. 6, we modify the 'download longest header chain rule' to remove equivocations in PoS. We show that this rule satisfies the above properties, and hence the analysis of this section carries over in PoS as well.

Proposition 4. *If $G_k = 1$ and $D_k = 0$, then during slots $[t_k, t_k + \nu]$, all honest nodes using the download rule $\mathcal{D}_{\text{long}}$ download content of at least $\widetilde{C}$ blocks that are produced in $(i, k]$, where $i < k$ is the largest index such that $\mathsf{CPivot}(i)$ (if such an i does not exist, $i = 0$).*

Lemma 3. *If all honest nodes use the download rule $\mathcal{D}_{\text{long}}$, and if*

$$\forall\, (i,j] \geq K_{\text{cp}}: \qquad \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) > Q_{(i-2K_{\text{cp}},j]}, \text{ and} \tag{13}$$

$$\forall m \geq 0\colon\ \frac{\widetilde{C}}{4} P_{(mK_{\text{cp}},(m+1)K_{\text{cp}}]} > Q_{((m-2)K_{\text{cp}},(m+2)K_{\text{cp}}]}, \tag{14}$$

then $\forall m \geq 0\colon\ \exists k_m^* \in \big(mK_{\text{cp}}, (m+1)K_{\text{cp}}\big] : \mathsf{CPivot}(k_m^*)$.

This is proven inductively using Prop. 4. The proof is in App. C.3.

5 PROOF-OF-WORK

For PoW, we use the simple download rule 'download the longest header chain'. In Lem. 3, we showed that under this download rule, cpivots occur in every K_{cp}-interval. We will use this to prove safety and liveness and identify the protocol parameters for which this holds wop in Thm. 1. Proofs are in App. D.

As noted in Sec. 3.1, it is most appropriate for PoW to set $\tau \to 0$, and to state its security properties in terms of real time. In order to use the results from Sec. 4, we must bridge between indices and real time. This is easy to do as the number of indices or non-empty slots is proportional to the time interval. In fact, as $\tau \longrightarrow 0$, the block production process converges to a Poisson point process with rate $\lambda \triangleq \rho/\tau$. Moreover, each non-empty slot has exactly one BPO (arrivals of a Poisson point process do not coincide).

Proof details in App. D. Result with $\Delta_{\text{h}} \approx 0$ (reasonable approximation for large block sizes) plotted in Fig. 1(c).

Theorem 1. *For all $\beta < 1/2$, $\lambda > 0$, such that*

$$\lambda < \max_{\widetilde{C}} \frac{1}{\Delta_{\text{h}} + \widetilde{C}/C} \ln\left(\frac{2(1-\beta)\widetilde{C}}{\widetilde{C} + 4 + \sqrt{8\widetilde{C} + 16}}\right), \tag{15}$$

the PoW longest chain protocol $\Pi^{\rho,\tau,k_{\text{conf}}}$ with the download rule $\mathcal{D}_{\text{long}}$, $\tau \longrightarrow 0$, $\rho = \lambda\tau$, and $k_{\text{conf}} = \Theta(\kappa^2)$ is secure with liveness latency $T_{\text{live}}^{\text{real}} \triangleq T_{\text{live}}\tau = \Theta(\kappa^2)$ over a time horizon of $K_{\text{hrzn}} = \text{poly}(\kappa)$ block productions.

6 SANITIZING-PROOF-OF-STAKE (SAPOS)

6.1 Equivocation Removal

For PoS, due to spamming by equivocations, we need a policy to ensure that nodes download at most one block from each BPO. We therefore propose the Sanitizing-Proof-of-Stake (SaPoS) protocol, in which the contents of provably equivocating blocks are sanitized from the blockchain. Pseudocodes Alg. 5 and Alg. 6 are in App. E.1.

The Download Rule in SaPoS. On top of any existing download rule (such as $\mathcal{D}_{\text{long}}$), we add another rule that an honest node does not download content for a header C if it has seen another equivocating header from the same BPO (same producing node and slot) as C. Instead of downloading content for such a header, the node considers that content to be "downloaded" and sets it to be empty (Alg. 5, l. 21). This means that the node can continue to download content for headers that extend C, and these blocks will be candidates for the node's longest downloaded chain dC.

Equivocation Proofs. With only the above download rule, one honest node may download content for a header while another may not (depending on when each node saw an equivocating header). In order to output a consistent ledger that all honest nodes have downloaded, honest nodes must agree on which blocks had an equivocation, and must unilaterally blank their contents.

For this, when an honest node produces a new block header, it adds an 'equivocation proof' against any equivocating blocks among the recent blocks in its downloaded longest chain. Specifically, the node picks from among the last k_{epf} block headers in its longest downloaded chain dC, block headers C' for which the node has seen an equivocating block header C', and there is no equivocation proof against it in any block header in dC. The node then creates an equivocation proof which consists of the two block headers C and C' and adds the equivocation proof to the header of the block that it creates (Alg. 5, l. 6).

The deadline k_{epf} for adding equivocation proofs exists so that the adversary cannot release an equivocation after its block has been confirmed, and force honest nodes to then blank the content for that block, thereby altering the ledger. The deadline also keeps the size of equivocation proofs in a header limited. We also don't want an equivocation proof to be repeated in several headers in a chain. Therefore, a block header C is considered invalid if it contains an equivocation proof against a block not in the prefix of C, a block more than k_{epf} blocks above C, or contains an equivocation proof that has already been proven in the prefix of C (Alg. 6, l. 6).

Ledger Construction in SaPoS. To create the ledger at the end of slot t, an honest node takes all blocks on its longest header chain that are k_{conf}-deep, then blanks the contents of any block against which there is an equivocation proof in a block header following it (Alg. 5, l. 12).

6.2 Security Theorem

Recall that the analysis in Sec. 4.3.3 uses four properties of the download rule. It is easy to see that with the addition of equivocation removal, the 'download longest header chain' rule satisfies these properties in PoS. The equivocation removal rule in SaPoS clearly satisfies the property that each honest node never downloads the same block twice (P1), and downloads at most one block from each BPO (P2). The rule will never prohibit download of an honest block because it has no equivocations, and blocks in its prefix will either be downloaded or blanked. Moreover, the rule never remains idle as long as there are block headers remaining with `UNKNOWN` content (P3). Finally, SaPoS does not spend bandwidth on any more blocks than the base download rule does, and since the base download rule $\mathcal{D}_{\text{long}}$ does not download blocks before the most recent cpivot (Prop. 4), the rule with equivocation removal also does not (P4). This means that the analysis of Sec. 4.3.3 works for SaPoS. Just like in PoW, this leads to liveness and consistency of the confirmed header chains of all honest nodes. Therefore, to ensure consistency of the ledger, we only need to show that the ledger construction process in SaPoS retains consistency. That is, if one honest node blanks the content of a block in its ledger, then all honest nodes do. Conversely, if one honest node does not blank the content for a block in its ledger, no honest node does. Proof details in App. E.2. Result with $\Delta_{\text{h}} \approx 0$ (large block sizes), and $\tau \longrightarrow 0$ (small slot approximation) plotted in Fig. 1(c).

Theorem 2. *For all $\beta < 1/2$, $\widetilde{C} \in \mathbb{N}$, and ρ, τ satisfying*

$$\frac{\widetilde{C}}{16} \frac{(2p_{\text{G}} - 1)^2}{p_{\text{G}}} > \frac{\rho}{1 - e^{-\rho}}, \quad p_{\text{G}} = (1 - \beta) e^{-\frac{\rho}{\tau}\left(\Delta_{\text{h}} + \widetilde{C}/C\right)}, \qquad (16)$$

there exists k_{epf}, $k_{\text{conf}} = \Theta(\kappa^2)$ such that the SaPoS protocol $\Pi_{\text{SaPoS}}^{\rho,\tau,k_{\text{conf}},k_{\text{epf}}}$ with the download rule $\mathcal{D}_{\text{long}}$, is secure with liveness latency $T_{\text{live}}^{\text{real}} = \Theta(\kappa^2)$ slots over a time horizon of $K_{\text{hrzn}} = \text{poly}(\kappa)$ block productions.

7 PREDICTABLE TRANSACTION VALIDITY

As discussed in Sec. 6.1, our PoS protocol variant SaPoS requires that honest nodes build on top of a header chain whose full block contents they cannot download (due to having already downloaded other blocks from the same equivocating BPO). In Bitcoin's history, such mining without validating block contents (termed SPV mining) has led to forks where some miners were extending blocks which later turned out to be invalid [24]. Thus, reaching consensus on the header chain is not enough, we need to consider how nodes catch up to and handle missing block contents.

One option to consider is using excess bandwidth to *catch-up* to the longest chain's contents. Here we hit the crux of what is termed the *data availability (DA) problem* [1]. We must guarantee that the contents of the blocks is available. DA would be satisfied if nodes only extended blocks that at least one honest node had downloaded the contents of. *The problem is that we don't know who is honest.*

In lieu of a DA scheme, in SaPoS (Sec. 6.1) we choose instead to *sanitize*, or exclude, the contents of equivocations from the ledger so no additional block downloads are needed and chain quality (more honest than dishonest blocks in the ledger) holds. *We still have the problem of predictable transaction validity—honest blocks may include transactions that depend on content from blocks whose equivocations were not known at the time.* Additionally, the adversary could use equivocations to invalidate transactions, taking up free space in honest blocks and lowering the effective throughput (valid confirmed transactions) of the ledger. We thus decouple the validity of transactions at the time they are added to a block, from when they are executed post-consensus.

Definition 4. A transaction has *predictable validity* if it is valid both at the time an honest node adds it to a block and when that block is executed.

Traditional LC protocols, which require a node to download and validate blocks before building on them, satisfy this definition as the state before the block is executed is deterministic. With our equivocation scheme, an honest node cannot know if a *recent* block will be equivocated. What we can guarantee is that at the time of creating a block, honest nodes *have seen all transactions which will be executed.* Unfortunately, the converse is not correct: *not all transactions nodes have seen will be executed.* If nodes limit transactions included in a block to those that don't depend on any *recent* state, then they can be sure all equivocations that could affect the validity state of a transaction have been included. The following lemma follows naturally.

Lemma 4. *If a node produces a block whose transactions do not share state with any transaction included in the last k_{epf} blocks, then the block satisfies predictable transaction validity.*

See App. F for proof details.

Block creators thus choose which transactions to include in a block so they remain valid and pay for their block space, note this is not a consensus rule. In practice, the Defi-ecosystem consists of very interdependent transactions (e.g., transactions interacting with major token exchanges and other prominent smart contracts) [20, 9], so it may not be practical to limit the interaction between transactions. The key property we want to guarantee is that any transaction included in an honest block that is part of the canonical chain will be paid for, i.e., the adversary cannot take up free honest block space, so we relax the definition of predictable validity.

Definition 5. A transaction has predictable *fee validity* if its fee can be paid both when an honest node adds it to a block and when that block is executed.

The block creator only needs to ensure that the transaction will be paid for, regardless of the outcome of its execution, so *we decouple the fee mechanism from the transaction validity.* Since the cost of transacting in smart contract platforms depends on the length of execution which, in turn, depends on state, we must also be sure to include a cost for the transaction size in SaPoS that is paid independently of execution gas costs. This is to ensure that attackers won't be able to spam the blockchain with transactions that are later sanitized, but still take up space in blocks.

Then for the miner to get paid, we require that the account funding the transaction has enough funds to cover the maximum gas, even if all transactions in its recent ancestor blocks are make it to the sanitized ledger and consume the maximum gas they could possibly need. To do this, we introduce a notion of *gas deposit accounts* to SaPoS that can only be used for transaction fees (transactions internally do not have access to these accounts). The maximum gas for a transaction can be bounded using the maximum gas value set be the sender in Ethereum-style transactions. Therefore, users who primarily make simple transactions (direct transfers having low maximum gas) or transact infrequently (few transactions in recent ancestor blocks) only need to maintain little balance. We also require that any deposit to the account is not considered in the balance until k_{epf} blocks after the deposit transaction. Withdraws however can take place immediately, as direct transactions.

Lemma 5. *If a node produces a block whose transactions are funded by gas deposit accounts with sufficient balance (balance before* k_{epf} *blocks minus any fees since), then all transactions in the block satisfy predictable fee validity.*

See App. F for proof details.

Thus, by sanitizing the contents of equivocating blocks and using our gas deposit scheme, we ensure that nodes download a maximum of one block per slot and that honest block creators only include transactions that pay for their spot in the block.

Note that in our scheme, we are primarily concerned with effective throughput as honest block space taken up by transactions that pay their place in the blockchain. There are user-side complexities that our scheme does not directly address. Since transactions can be sanitized, we can no longer rely on transaction nonce schemes that are strictly incremental but instead must relax them to strictly increasing. In lieu of stronger validity guarantees, it is the onus of the user to make sure their transactions behave correctly in the event some get sanitized. Sanitizing block content also opens up the potential for the adversary to perform free options (for a limitted amount of time) by including transactions in a block that they can later decide to cancel (by revealing an equivocation at no cost within the allowed window).

8 CONCLUSION

In this work we focused on the security of the longest chain protocol both in the PoW and PoS settings. While block downloading and processing is usually implemented in an ad-hoc manner and is not typically discussed in the context of the protocol's security analysis, our work highlights the importance of correctly prioritizing block download and processing. In addition to providing a security proof using new techniques, and attacks on natural prioritization rules in the PoW setting, we also propose SaPoS, a new proof-of-stake variant. Several important open questions remain:

- There remain gaps between security bounds we provide in the PoW setting and the known attacks in this case (cf. Fig. 1(b)). Can better attacks be found? What are the optimal prioritization rules for which security is achieved?
- In the PoW setting, the attacker is unable to equivocate, but in SaPoS we were forced to deal with equivocations. This came at a cost to the latency of transaction execution, and with decreased certainty about the state at which the transaction is eventually executed. Can these costs be avoided, so that PoS based LC is on par with the PoW variant?
- The difficulty adjustment algorithm (DAA) seems to apply even more stress to limited capacity nodes. Can DAAs be designed for this setting and incorporated into the security analysis?
- Can processing and download parallelization, pre-processing and pre-fetching of blocks be utilized more efficiently in order to securely improve the throughput of LC based protocols?
- In SaPoS, the deadline for including equivocation proofs is not user-dependent, but baked into the protocol. A user cannot increase this to achieve lower error probability. This is a drawback compared to traditional Nakamoto consensus. Can it be avoided?

ACKNOWLEDGMENTS

We thank Lei Yang, Mohammad Alizadeh, Ertem Nusret Tas, Ghassan Karame, Florian Tschorsch, and George Danezis for fruitful discussions. The work of LK, JN, and AZ was conducted in part during Dagstuhl Seminar #22421. JN is supported by the Protocol Labs PhD Fellowship. JN and SS are supported by a gift from the Ethereum Foundation. LK is supported by the armasuisse Science and Technology CYD Distinguished Postdoctoral Fellowship. AZ is supported by grant #1443/21 from the Israel Science Foundation.

REFERENCES

[1] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. 2021. Fraud and data availability proofs: detecting invalid blocks in light clients. In *Financial Cryptography (2)* (LNCS). Vol. 12675. Springer, 279–298.

[2] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2018. Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability. In *CCS*. ACM, 913–930.

[3] Vivek Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. 2019. Proof-of-stake longest chain protocols: security vs predictability. (2019). arXiv: 1910.02218v3 [cs.CR].

[4] Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. 2019. Prism: deconstructing the blockchain to approach physical limits. In *CCS*. ACM, 585–602.

[5] Bitcoin Project. [n. d.] Running a full node — Bitcoin. Retrieved Mar. 15, 2023 from https://bitcoin.org/en/full-node#minimum-requirements.

[6] 2019. Block size limit controversy. (Apr. 24, 2019). Retrieved Jan. 19, 2023 from https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[7] Christian Cachin and Stefano Tessaro. 2005. Asynchronous verifiable information dispersal. In *DISC* (LNCS). Vol. 3724. Springer, 503–504.

[8] Miguel Castro, Peter Druschel, Ayalvadi J. Ganesh, Antony I. T. Rowstron, and Dan S. Wallach. 2002. Secure routing for structured peer-to-peer overlay networks. In *OSDI*. USENIX Association.

[9] Ting Chen, Zihao Li, Yuxiao Zhu, Jiachi Chen, Xiapu Luo, John Chi-Shing Lui, Xiaodong Lin, and Xiaosong Zhang. 2020. Understanding Ethereum via graph analysis. *ACM Trans. Internet Techn.*, 20, 2, 18:1–18:32.

[10] Phil Daian, Rafael Pass, and Elaine Shi. 2019. Snow white: robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography* (LNCS). Vol. 11598. Springer, 23–41.

[11] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)* (LNCS). Vol. 10821. Springer, 66–98.

[12] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the Bitcoin network. In *P2P*. IEEE, 1–10.

[13] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. 2020. Everything is a race and Nakamoto always wins. In *CCS*. ACM, 859–878.

[14] Thomas D. Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. 2020. Adding concurrency to smart contracts. *Distributed Comput.*, 33, 3-4, 209–225.

[15] John Duchi. [n. d.] Hoeffding's inequality. CS229 Supplemental Lecture notes. Retrieved Jan. 8, 2023 from http://cs229.stanford.edu/extra-notes/hoeffding.pdf.

[16] Matthias Fitzi, Peter Gaži, Aggelos Kiayias, and Alexander Russell. 2020. Proof-of-stake blockchain protocols with near-optimal throughput. Cryptology ePrint Archive, Paper 2020/037. (2020). https://eprint.iacr.org/2020/037.

[17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2017. The Bitcoin backbone protocol with chains of variable difficulty. In *CRYPTO (1)* (LNCS). Vol. 10401. Springer, 291–323.

[18] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin backbone protocol: analysis and applications. In *EUROCRYPT (2)* (LNCS). Vol. 9057. Springer, 281–310.

[19] Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Tight consistency bounds for Bitcoin. In *CCS*. ACM, 819–838.

[20] Dongchao Guo, Jiaqing Dong, and Kai Wang. 2019. Graph structure and statistical properties of Ethereum transaction relationships. *Inf. Sci.*, 492, 58–71.

[21] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on Bitcoin's peer-to-peer network. In *USENIX Security Symposium*. USENIX Association, 129–144.

[22] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. 2007. Verifying distributed erasure-coded data. In *PODC*. ACM, 139–146.

[23] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, 301, 13–30. DOI: 10.1080/01621459.1963.10500830.

[24] Puneet Kumar Kaushal, Amandeep Bagga, and Rajeev Sobti. 2017. Evolution of Bitcoin and security risk in Bitcoin wallets. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*. IEEE, 172–177.

[25] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO (1)* (LNCS). Vol. 10401. Springer, 357–388.

[26] Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. 2018. A better method to analyze blockchain consistency. In *CCS*. ACM, 729–744.

[27] Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. 2021. Under the hood of the Ethereum gossip protocol. In *Financial Cryptography (2)* (LNCS). Vol. 12675. Springer, 437–456.

[28] Andreas Lenz. 2022. Random walk with positive drift. Mathematics Stack Exchange. (May 12, 2022). Retrieved Jan. 7, 2023 from https://math.stackexchange.com/q/4449213.

[29] Jing Li, Dongning Guo, and Ling Ren. 2021. Close latency-security trade-off for the Nakamoto consensus. In *AFT*. ACM, 100–113.

[30] Michael McSweeney. 2021. Solana experiences transaction stoppage as developers report 'intermittent instability'. (Sept. 14, 2021). https://www.theblockcrypto.com/linked/117624/solana-experiences-transaction-stoppage-as-developers-report-intermittent-instability.

[31] Mike Millard. 2022. Solana restarted after seven-hour outage caused by surge of transactions. (May 1, 2022). https://www.theblockcrypto.com/linked/144639/solana-restarted-after-seven-hour-outage-caused-by-surge-of-transactions.

[32] Satoshi Nakamoto. 2008. Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. (2008).

[33] Kamilla Nazirkhanova, Joachim Neu, and David Tse. 2022. Information dispersal with provable retrievability for rollups. In *AFT*. ACM. https://eprint.iacr.org/2021/1544.

[34] Joachim Neu, Srivatsan Sridhar, Lei Yang, David Tse, and Mohammad Alizadeh. 2022. Longest chain consensus under bandwidth constraint. In *AFT*. ACM. https://eprint.iacr.org/2021/1545.

[35] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *EUROCRYPT (2)* (LNCS). Vol. 10211, 643–673.

[36] Rafael Pass and Elaine Shi. 2017. The sleepy model of consensus. In *ASIACRYPT (2)* (LNCS). Vol. 10625. Springer, 380–409.

[37] Brian Quarmby. 2022. Solana hit with another network incident causing degraded performance. (Jan. 5, 2022). https://cointelegraph.com/news/solana-hit-with-another-network-incident-causing-degraded-performance.

[38] Ling Ren. 2019. Analysis of Nakamoto consensus. Cryptology ePrint Archive, Paper 2019/943. (2019). https://eprint.iacr.org/2019/943.

[39] Suryanarayana Sankagiri, Shreyas Gandlur, and Bruce Hajek. 2021. The longest-chain protocol under random delays. (2021). arXiv: 2102.00973v1 [cs.CR].

[40] Vikram Saraph and Maurice Herlihy. 2019. An empirical study of speculative concurrency in Ethereum smart contracts. In *Tokenomics* (OASIcs). Vol. 71. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 4:1–4:15.

[41] Atul Singh, Miguel Castro, Peter Druschel, and Antony I. T. Rowstron. 2004. Defending against eclipse attacks on overlay networks. In *ACM SIGOPS European Workshop*. ACM, 21.

[42] Atul Singh, Tsuen-Wan Ngan, Peter Druschel, and Dan S. Wallach. 2006. Eclipse attacks on overlay networks: threats and defenses. In *INFOCOM*. IEEE.

[43] Emil Sit and Robert Tappan Morris. 2002. Security considerations for peer-to-peer distributed hash tables. In *IPTPS* (LNCS). Vol. 2429. Springer, 261–269.

[44] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure high-rate transaction processing in Bitcoin. In *Financial Cryptography* (LNCS). Vol. 8975. Springer, 507–527.

[45] Eric W. Weisstein. 2022. q-Pochhammer Symbol – from Wolfram Mathworld. https://mathworld.wolfram.com/q-PochhammerSymbol.html. [Online; accessed 10-November-2022]. (2022).

[46] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. 2022. DispersedLedger: high-throughput byzantine consensus on variable bandwidth networks. In *NSDI*. USENIX Association, 493–512.

A PROTOCOL ALGORITHMS REFERENCE

A.1 Helper Functions for Pseudocode

- $\mathrm{Hash}(\mathrm{txs})$: Cryptographic hash function to produce a binding commitment to txs (modelled as a random oracle)
- $C' \preceq C, C \succeq C'$: Relation describing that C' is a prefix of C
- $C \| C'$: Concatenation of C and C'
- $|C|$: Length of C
- $(\texttt{true}$ with probability x, else $\texttt{false})$: Bernoulli random variable with success probability x
- $\mathrm{prefixChainsOf}(C)$: Set of prefixes of C, *i.e.*, all C' with $C' \preceq C$
- $\mathrm{newBlock}(\mathrm{txsHash}\colon \mathrm{Hash}(\mathrm{txs}))$ and $\mathrm{newBlock}(\mathrm{time}\colon t, \mathrm{node}\colon P, \mathrm{txsHash}\colon \mathrm{Hash}(\mathrm{txs}))$: Produce a new PoW and PoS block header with given parameters, respectively
- $\mathrm{txsLedger}(\mathrm{blkTxs}, C)$: Concatenates the block contents stored in blkTxs for the blocks along the chain C, to obtain the corresponding transaction ledger
- $(C \overset{\text{BPO}}{\equiv} C') \triangleq (C \neq C') \wedge (C.\mathrm{node} = C'.\mathrm{node}) \wedge (C.\mathrm{time} = C'.\mathrm{time})$: Relation for distinct headers from the same BPO

A.2 Environment $\mathcal{Z}$

The environment $\mathcal{Z}$ initializes N nodes and lets $\mathcal{A}$ corrupt up to βN nodes at the beginning of the execution. Corrupted nodes are controlled by the adversary. Honest nodes run $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$. The environment maintains a mapping $\mathcal{Z}$.blkTxs from block headers to the block content (transactions). This mapping is referred to as the 'cloud' in Sec. 3 and Fig. 8. $\mathcal{Z}$ also maintains for each node a queue of pending block headers to be delivered after a delay determined by the adversary. If $\mathcal{A}$ has not instructed $\mathcal{Z}$ to deliver a header Δ_{h} real time after it was added to the queue of pending block headers, then $\mathcal{Z}$ delivers it to the node.

Honest nodes and $\mathcal{A}$ interact with $\mathcal{Z}$ via the following functions:

- $\mathcal{Z}.\textsc{broadcastHeaderChain}(C)$:
  If called by an honest node, $\mathcal{Z}$ enqueues C in the queue of pending block headers for each node, and notifies $\mathcal{A}$. Then, for each node P, on receiving $\textsc{deliver}(C, P)$ from $\mathcal{A}$, or when Δ_{h} time has passed since C was added to the queue of pending headers, $\mathcal{Z}$ triggers $P.\textsc{receivedHeaderChain}(C)$.
- $\mathcal{Z}.\textsc{uploadContent}(C, \mathrm{txs})$:
  $\mathcal{Z}$ stores a mapping from the header chain C to the content txs of its last block by setting $\mathcal{Z}.\mathrm{blkTxs}[C] = \mathrm{txs}$. $\mathcal{Z}$ only stores the content txs if $\mathrm{Hash}(\mathrm{txs}) = C.\mathrm{txsHash}$.
- $\mathcal{Z}.\textsc{receivePendingTxs}()$:
  $\mathcal{Z}$ generates a set of pending transactions and returns them.
- If node P at slot t requests the content associated with a block header C, $\mathcal{Z}$ acts as follows. If $\mathcal{Z}.\mathrm{blkTxs}[C]$ is set, then let $\mathrm{txs} = \mathcal{Z}.\mathrm{blkTxs}[C]$ (if not, $\mathcal{Z}$ ignores the request). If the request was received from an honest node P, if $\mathcal{Z}$ has recently triggered $P.\textsc{receivedContent}(.)$ at a rate below C, then $\mathcal{Z}$ triggers $P.\textsc{receivedContent}(C, \mathrm{txs})$ (else, $\mathcal{Z}$ ignores the request). If the request was received from $\mathcal{A}$, $\mathcal{Z}$ sends (C, txs) to $\mathcal{A}$.

At all times, $\mathcal{A}$ can trigger $P.\textsc{receivedHeaderChain}(C)$ and $P.\textsc{receivedContent}(C, \mathrm{txs})$ for honest nodes P (bypassing header delay and bandwidth constraint in an adversarially chosen way).

B SIMULATION DETAILS

To complement the theoretical analysis, we conducted simulations of a PoW blockchain with bandwidth constraints. We evaluated several download rules with and without the presence of attackers. The simulations were written as event-driven simulations using Python's simpy package.[4]

Nodes in our simulation generate blocks in a Poisson process with rate proportional to their mining power. We assume the mining difficulty is fixed, and do not include any adjustment by a difficulty adjustment algorithm (DAA). In fact, DAAs tend to worsen processing problems as they increase the block creation rate if the chain does not grow fast enough—which in turn requires more download from nodes.

Nodes process blocks one at a time according to the priority dictated by the processing policy, at a rate determined by their capacity. They are allowed to preempt their current task if new information (headers that are published, blocks that they mined) presents them with a higher priority targets. Since queues can grow large if nodes do not manage to process all blocks in a timely manner, we maintain priority queues of bounded size (typically 100) and evict low priority tasks from the queue as needed. If nodes do keep up, queues remain small, and all is well. If however queues grow large, it is usually safe to discard low priority tasks, since higher priority alternatives are arriving at a fast pace, advertised by peers that continue to mine. The high rate of incoming header announcements implies the node will never manage to process all low priority blocks unless their priority changes (in which case they will be re-advertised).

As preemption of downloads may cause nodes to alternate between downloads, we run the risk of wasting work if we discard partially processed information. We therefore allow nodes to retain partial work in an LRU cache of size 10. Cached entries allow nodes to resume processing where they left off. (We note that in practice, it may be difficult to cache information, and that in realistic settings such caching mechanisms may be targeted by an adversary that will flood nodes with incorrect information that they cannot validate prior to completing the processing of the entire block.)

Except where we note otherwise, headers are assumed to propagate instantly in the simulations. Block headers in the PoW settings contain the proof-of-work itself, which can be easily validated. We therefore assume the adversary never publishes headers it did not actually mine. To remain close to the theoretical analysis, we model all processing tasks as dependent only on the resources available to the node itself. In reality, things are much more complex: nodes typically propagate blocks in a P2P network, which means both the overlay network topology and the underlying internet topology both greatly impact block download rates and performance. Block processing in turn, behaves differently and does not depend on the topology. With bandwidth nodes need to decide on ways to balance incoming and outgoing bandwidth between their peers, and attackers may try to isolate nodes via eclipse attacks [21, 43, 42, 41, 8]. Our simplified setting allows us to focus more on the priority rules in isolation from the effects of topology and other P2P related issues that are bandwidth-specific.

An Example Run. Fig. 2 is an example of a trace generated by our simulation for a simple setting with only 5 nodes. The x-axis is time, and each node's timeline is represented horizontally at a different height along the y-axis. Blocks that are created are shown as squares, placed at the time of their creation, and arrows point to their parent blocks. Each block is named $h.j$ to denote that it is the j'th block of height h to be created.

The timeline of each node also depicts the blocks it is processing at any particular time. For example, block 1.1 is created within the first second of the simulation by Node 3 and other nodes begin to process it immediately. This work concludes before the next block is mined. However, that is not always the case. Block 3.2 for example, is mined by Node 1 at around time 4, but a previous block at this height (block 3.1) was mined earlier. Node 1 had not finished validating it, and therefore did not mine on top of it.

Finally, it is possible to see processing tasks that are preempted and resumed later. For example, Node 0 is in the process of validating block 3.2 when block 4.1 is advertized. It stops its current download since 4.1 represents a longer chain. Node 0 resumes the download of block 3.2 one second later.

Each point in our simulation result graphs is typically computed from multiple repetitions of the same experiment. We normalize time so that a block is created by the honest nodes once every time unit in expectation. We thus consider the basic time unit as 1 second, and the total block creation rate as $\lambda_{\text{hon}} = 1$. Each time we start the chain at the genesis block and run for a period of $5,000$ to $10,000$ seconds (depending on the experiment). Bandwidth is measured in units of blocks per second. The standard deviation of values plotted is typically well below 1% of the values themselves. Error bars are thus too small to properly appear in the plot, and were not added.

B.1 Chain Growth Rate at Capacity

The rate at which the chain grows without the presence of an attacker sets a bound on the security of the system: if the chain grows at a rate λ_{grwth}, then an attacker mining at that rate or above is able to overtake the blockchain at will.

As a baseline comparison for other simulations, we consider a a network of 100 honest nodes without the presence of an attacker, and measure the rate of growth of the chain λ_{grwth} in two scenarios: (a) Non-zero header delay $\Delta_{\text{h}} > 0$, and infinite processing speed $C = \infty$. (b) No header delay $\Delta_{\text{h}} = 0$, and a constant processing rate $C < \infty$. The first case is equivalent to the bounded delay model, and blocks arrive at all nodes exactly Δ_{h} seconds after they are created. The second scenario is in our model and includes queueing delays of blocks. To properly compare between the two models, we note that Δ_{h}^{-1} can be considered as an effective rate at which a single block is propagated.

Fig. 10 depicts the results of the simulations. It shows that indeed as bandwidth decreases (or header propagation time increases) the chain grows at a slower pace. It is perhaps surprising to see that some growth of the chain occurs even when download rates are below the rate needed to download all blocks produced (*e.g.*, a processing rate of $1/2$ allows nodes to download at most $1/2$ of the blocks that are created). The reason progress still takes place at extremely low rates is that nodes that are behind create blocks

[4] Source code: https://github.com/avivz/finitebwlc

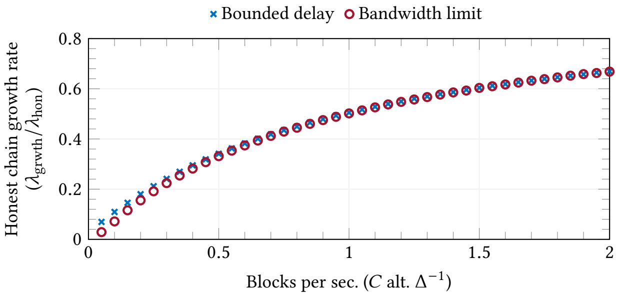


Figure 10: The rate of chain growth (as a fraction of block production rate) for 100 identical nodes in the bounded-delay model (×) and in the bounded-bandwidth model (○). No attacker is present. In the bounded-delay model the x-axis is Δ^{-1} (the inverse of the delay bound for delivering messages) which is interpreted as the effective rate at which a single block is delivered.

that others do not need to process, and hence their blocks, which do not contribute to the height of the chain, at least do not waste resources.

Fig. 10 also shows that the limited capacity case is slightly worse than the bounded delay setting for comparable delays. We note that in our simulation, even in the limitted capacity case, if blocks of the same height are created, the first one is advertised to all nodes instantly and is downloaded first. This results in this block being downloaded in a coordinated manner by all nodes, and thus most likely extended. This is the same as in the bounded delay setting. Queuing delays (*i.e.*, delaying a block while we are downloading its parent) occur only on rare occasions—when a miner has single-handedly managed to mine several consecutive blocks (a rare occurence in our highly decentralized setting).

C SECURITY ANALYSIS PROOFS

Refer to Tab. 1 for a recap of notation and definitions.

Proof of Prop. 1. First, for any k,

$$\Pr\left[G_k = 1\right] = \Pr\left[\mathsf{Good}(t_k) \mid \neg\mathsf{Empty}(t_k)\right] \tag{17}$$

$$= \frac{\Pr\left[\mathsf{Good}(t_k)\right]}{\Pr\left[\mathsf{Empty}(t_k)\right]} = \frac{(1-\beta)\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}}. \tag{18}$$

Take an iid random process $\{T_k\}$ with $\Pr\left[T_k = t\right] = (1-p_{\mathrm{E}})p_{\mathrm{E}}^t$ for $t \geq 0$ where $p_{\mathrm{E}} = \Pr\left[H_t + A_t = 0\right]$. The random variables $\{T_k\}$ describe the inter-arrival times between non-empty slots. Take another iid random process $\{G'_k\}$, independent of $\{T_k\}$, such that $G'_k = 1$ with probability $\Pr\left[H_t = 1 \wedge A_t = 0 \mid H_t + A_t > 0\right]$ and $G'_k = 0$ otherwise. The random process $\{G_k\}$ can be equivalently defined as $G_k = 1$ iff $G'_k = 1$ and $T_k \geq \nu$.

The independence of the random variables $\{G_k\}$ then follows from the independence of the random variables $\{(T_k, G'_k)\}$. □

C.1 Combinatorial Pivots Stabilize

Proposition 5. *For any $i < j$,*

$$L_{\min}(t_j + \nu) \geq L_{\min}(t_{i+1} - 1) + D_{(i,j]}. \tag{19}$$

Table 1: Summary of notation (cf. Secs. 3.4 and 4.2)

	Protocol parameters
τ	Slot duration (seconds)
ρ	Avg. no. of BPOs per slot
k_{conf}	Confirmation depth
	Model parameters
β	Fraction of adversarial nodes
Δ_{h}	Header propagation delay (seconds)
C	Bandwidth (blocks/second)
	Analysis variables
ν	No. of empty slots after a good slot
$\widetilde{C}$	No. of blocks downloaded in ν slots
t_k	k-th non-empty slot
G_k	1 iff slot t_k is good
D_k	1 iff $G_k = 1$ and block in t_k downloaded
P_k	1 iff index k is a ppivot

Proof. For each $k \in \{i+1, \ldots, j\}$, if $D_k = 1$,

$$L_{\min}(t_{k+1} - 1) \geq L_{\min}(t_k + \nu) \quad (D_k = 1 \implies t_{k+1} > t_k + \nu) \tag{20}$$

$$\geq L_{\min}(t_k - 1) + 1 \quad \text{(from Prop. 3).} \tag{21}$$

If $D_k = 0$, clearly $L_{\min}(t_{k+1} - 1) \geq L_{\min}(t_k - 1)$. Adding these up gives the required result. □

Proof of Lem. 1. Note that $\mathrm{d}\mathcal{C}_p(t)$ is a valid chain at slot t and $\left|\mathrm{d}\mathcal{C}_p(t)\right| = L_p(t) \geq L_{\min}(t)$. Therefore, it suffices to show the first claim of the lemma.

For contradiction, let $s \geq t_k + \nu$ be the first slot in which there is a valid header chain $\mathcal{C}'$ such that $|\mathcal{C}'| \geq L_{\min}(s)$ and $b^* \notin \mathcal{C}'$.

Let b' be the block with maximum height on the chain $\mathcal{C}'$, such that b' was produced in a slot t_i with $D_i = 1$. For $\mathcal{C}'$ to be a valid chain at slot s, we need $t_i \leq s$. Since the block b' is produced by an honest node, b' extends $\mathrm{d}\mathcal{C}_q(t_i - 1)$ for some honest node q. Therefore, $\mathrm{d}\mathcal{C}_q(t_i - 1)$ is a prefix of $\mathcal{C}'$. This means that $b^* \notin \mathrm{d}\mathcal{C}_q(t_i - 1)$. Moreover, $\left|\mathrm{d}\mathcal{C}_q(t_i - 1)\right| = L_q(t_i - 1) \geq L_{\min}(t_i - 1)$. If $i > k$, then $t_i - 1 \geq t_k + \nu$ (since $D_k = 1$) and $t_i - 1 < s$ (shown above). This is a contradiction because we assumed that s is the first slot such that $s \geq t_k + \nu$ and $b^* \notin \mathcal{C}'$ and $|\mathcal{C}'| \geq L_{\min}(s)$ for some valid chain $\mathcal{C}'$. Since b^* is the only block produced in slot t_k, $i = k$ is also not possible. We conclude that $i < k$.

Since $D_i = 1$ and b' is produced in slot t_i,

$$L_{\min}(t_i + \nu) \geq \left|b'\right|. \tag{22}$$

By assumption,

$$\left|\mathcal{C}'\right| \geq L_{\min}(s). \tag{23}$$

Let t_j be the last non-empty slot such that $t_j \leq s$. Note that $j \geq k > i$. We must consider two cases:

(1) Case 1: $s \geq t_j + \nu$ or $D_j = 0$. If $D_j = 0$, we don't have to worry about whether the block from slot t_j was downloaded by all honest nodes. If $D_j = 1$ but $s \geq t_j + \nu$, then we know that all honest nodes have downloaded the block from slot t_j before

the end of slot s. That is,

$$\begin{aligned} L_{\min}(s) &\geq L_{\min}(t_j + v) && (24)\\ &\geq L_{\min}(t_{i+1} - 1) + D_{(i,j]} \quad \text{(from Prop. 5)} && (25)\\ &\geq L_{\min}(t_i + v) + D_{(i,j]}. && (26) \end{aligned}$$

By definition of b', all blocks in C' appearing after b' correspond to $\overline{D}$-slots. These blocks must be from distinct indices greater than i but at most j. So,

$$\left|C'\right| \leq \left|b'\right| + \overline{D}_{(i,j]}. \tag{27}$$

From eqns. (22), (23), (26) and (27), we derive

$$D_{(i,j]} \leq \overline{D}_{(i,j]} \implies Y_{(i,j]} \leq 0 \implies Y_{(0,i]} < Y_{(0,j]} \tag{28}$$

where $i < k \leq j$.

(2) Case 2: $t_j \leq s < t_j + v$ and $D_j = 1$. In this case, the block from slot t_j may not have enough time to be downloaded by all honest nodes before the end of slot s. However, for any $l < j$ such that $D_l = 1$, $t_l + v < t_j \leq s$, so there is enough time to download the block from slot t_l. Let $l \in (i, j-1]$ be the greatest index such that $D_l = 1$. Then, $t_j > t_l + v$, and $D_{(i,l]} = D_{(i,j-1]}$.

$$\begin{aligned} L_{\min}(s) &\geq L_{\min}(t_j) && (29)\\ &\geq L_{\min}(t_l + v) && (30)\\ &\geq L_{\min}(t_{i+1} - 1) + D_{(i,l]} \quad \text{(from Prop. 5)} && (31)\\ &\geq L_{\min}(t_i + v) + D_{(i,j-1]}. && (32) \end{aligned}$$

Note that since $D_j = 1$, $\overline{D}_{(i,j]} = \overline{D}_{(i,j-1]}$. Therefore, as in the previous case,

$$\left|C'\right| \leq \left|b'\right| + \overline{D}_{(i,j-1]}. \tag{33}$$

From eqns. (22), (23), (29) and (33),

$$D_{(i,j-1]} \leq \overline{D}_{(i,j-1]} \implies Y_{(i,j-1]} \leq 0 \implies Y_{(0,i]} < Y_{(0,j-1]}. \tag{34}$$

Note that since we assumed $s \geq t_k + v$ and $s < t_j + v$, we know that $j > k$. Therefore, $i < k \leq j - 1$.

In either case, eqn. (28) or eqn. (34) contradict the assumption CPivot(k) (Def. 3). □

C.2 Probabilistic Pivots are Abundant

We build up to the proof of Lem. 2 through a series of propositions, starting with recalling a versatile tail bound.

Proposition 6 (Hoeffding's inequality [23] [15, Thm. 4]). *Let $Z_1, ..., Z_n$ be independent bounded random variables with $\forall i : Z_i \in [a, b]$, where $-\infty < a \leq b < \infty$. Then, $\forall t \geq 0$:*

$$\Pr\left[\left(\sum_{i=1}^{n} Z_i\right) - \mathbb{E}\left[\sum_{i=1}^{n} Z_i\right] \geq tn\right] \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right) \tag{35}$$

$$\Pr\left[\left(\sum_{i=1}^{n} Z_i\right) - \mathbb{E}\left[\sum_{i=1}^{n} Z_i\right] \leq -tn\right] \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right). \tag{36}$$

Proposition 7. *With $\alpha_2 \triangleq 2\varepsilon_{\mathrm{G}}^2$,*

$$\forall\, (i, j] : \forall \delta \geq 0 :$$
$$\Pr\left[X_{(i,j]} \leq (1-\delta) 2\varepsilon_{\mathrm{G}}(j-i)\right] \leq \exp(-\alpha_2 \delta^2 (j-i)). \tag{37}$$

Proof. By Hoeffding's inequality (Prop. 6). □

Proposition 8.

$$\forall k\colon \Pr\left[\mathsf{PPivot}(k)\right] \geq (2p_{\mathrm{G}} - 1)^2 / p_{\mathrm{G}} \triangleq p_{\mathrm{ppivot}} \tag{38}$$

Proof. Eqn. (10) characterizes PPivot(k) as the intersection of three independent events:

$$\begin{aligned} \mathcal{E}_1 &\triangleq \{X_k = 1\} && (39)\\ \mathcal{E}_2 &\triangleq \{\forall \ell\colon X_{(k,k+\ell]} \geq 0\} && (40)\\ \mathcal{E}_3 &\triangleq \{\forall \ell\colon X_{(k-1-\ell,k-1]} \geq 0\} && (41) \end{aligned}$$

Their probabilities are easily calculated [28]:

$$\Pr\left[\mathcal{E}_1\right] = p_{\mathrm{G}} \qquad \Pr\left[\mathcal{E}_2\right] = \Pr\left[\mathcal{E}_3\right] = (2p_{\mathrm{G}} - 1)/p_{\mathrm{G}} \tag{42}$$

□

Proposition 9. *With $\alpha_3 \triangleq 2p_{\mathrm{ppivot}}^2$,*

$$\begin{aligned} \forall\, (i, j] \asymp 2K_1K_2 : \quad & \Pr\left[P_{(i,j]} \leq (1-\delta) p_{\mathrm{ppivot}} 2K_1K_2\right] \\ & \leq 2K_1 \exp(-\alpha_3 \delta^2 K_2) + K_{\mathrm{hrzn}}^2 \exp(-\alpha_2 K_1). \end{aligned} \tag{43}$$

Proof. Let $\mathcal{E} \triangleq \{\forall\, (i, j] \geq K_1 : X_{(i,j]} > 0\}$. From Prop. 7 with $\delta = 1$, and a union bound over all intervals ($\leq K_{\mathrm{hrzn}}^2$ many), we get

$$\Pr\left[\neg\mathcal{E}\right] \leq K_{\mathrm{hrzn}}^2 \exp(-\alpha_2 K_1). \tag{44}$$

For any given index k, we can partition the intervals of eqn. (8) into 'long' and 'short' intervals (length at least vs. less than K_1):

$$\begin{aligned} \mathcal{E}_k &\triangleq \{\mathsf{PPivot}(k)\} = \mathcal{E}_k^{\mathrm{L}} \wedge \mathcal{E}_k^{\mathrm{S}} && (45)\\ \mathcal{E}_k^{\mathrm{L}} &\triangleq \{\forall k \in (i, j] \geq K_1 : X_{(i,j]} > 0\} && (46)\\ \mathcal{E}_k^{\mathrm{S}} &\triangleq \{\forall k \in (i, j] \prec K_1 : X_{(i,j]} > 0\}. && (47) \end{aligned}$$

Note that $\mathcal{E}_k^{\mathrm{L}} \supseteq \mathcal{E}$. Thus, for any two given indices k_1, k_2, if k_1, k_2 are 'far apart', *i.e.*, if $|k_1 - k_2| \geq 2K_1$, then $\mathcal{E}_{k_1}$ and $\mathcal{E}_{k_2}$ are conditionally independent given $\mathcal{E}$ (since $\mathcal{E}_{k_1}^{\mathrm{S}}$ and $\mathcal{E}_{k_2}^{\mathrm{S}}$ are).

We bound and decompose $I^* \triangleq (i, j] = (i, i + 2K_1K_2] = \bigcup_{\ell=1}^{2K_1} I_\ell$:

$$\begin{aligned} \forall \ell \in \{1, ..., 2K_1\} : \quad I_\ell \triangleq \{& i + 0 \cdot 2K_1 + \ell, ... \\ & ..., i + (K_2 - 1) \cdot 2K_1 + \ell\}. \end{aligned} \tag{48}$$

We define corresponding events, $\forall \ell \in \{1, ..., 2K_1\}$:

$$\begin{aligned} \mathcal{E}^* &\triangleq \left\{P_{I^*} \leq (1-\delta) p_{\mathrm{ppivot}} 2K_1K_2\right\} && (49)\\ \mathcal{E}_\ell &\triangleq \left\{P_{I_\ell} \leq (1-\delta) p_{\mathrm{ppivot}} K_2\right\}. && (50) \end{aligned}$$

Clearly, $\mathcal{E}^* \subseteq \bigcup_{\ell=1}^{2K_1} \mathcal{E}_\ell$. Thus, by a union bound,

$$\Pr\left[\mathcal{E}^* \,\middle|\, \mathcal{E}\right] \leq \sum_{\ell=1}^{2K_1} \Pr\left[\mathcal{E}_\ell \,|\, \mathcal{E}\right]. \tag{51}$$

Furthermore, $\forall \ell \in \{1, ..., 2K_1\}$, and with $\mu_\ell \triangleq \mathbb{E}\left[P_{I_\ell} \,\middle|\, \mathcal{E}\right]$:

$$\begin{aligned} \Pr\left[\mathcal{E}_\ell \,|\, \mathcal{E}\right] &= \Pr\left[P_{I_\ell} \leq (1-\delta) p_{\mathrm{ppivot}} K_2 \,\middle|\, \mathcal{E}\right] && (52)\\ &\overset{\text{(a)}}{\leq} \Pr\left[P_{I_\ell} \leq (1-\delta)\mu_\ell \,\middle|\, \mathcal{E}\right] && (53)\\ &\overset{\text{(b)}}{\leq} \exp(-2\delta^2 \mu_\ell^2 / K_2) \overset{\text{(c)}}{\leq} \exp(-2p_{\mathrm{ppivot}}^2 \delta^2 K_2), && (54) \end{aligned}$$

where (a) and (c) use

$$\mu_\ell = K_2 \mathbb{E}\left[\mathbb{1}_{\{\mathsf{PPivot}(k)\}} \,\middle|\, \mathcal{E}\right] \geq K_2 \mathbb{E}\left[\mathbb{1}_{\{\mathsf{PPivot}(k)\}}\right] \geq K_2 p_{\mathrm{ppivot}} \tag{55}$$

(Prop. 8), and (b) uses that $\{\mathsf{PPivot}(k_1)\}$ and $\{\mathsf{PPivot}(k_2)\}$ are conditionally independent given $\mathcal{E}$ for $k_1, k_2 \in I_\ell$, and Hoeffding's inequality (Prop. 6).

Thus, we complete the proof by observing, as desired, that

$$\begin{aligned}
\Pr\left[\mathcal{E}^*\right] &= \Pr\left[\mathcal{E}^* \cap \mathcal{E}\right] + \Pr\left[\mathcal{E}^* \cap \neg\mathcal{E}\right] && (56)\\
&\leq \Pr\left[\mathcal{E}^* \middle| \mathcal{E}\right] + \Pr\left[\neg\mathcal{E}\right] && (57)\\
&\leq 2K_1 \exp(-2p_{\mathrm{ppivot}}^2 \delta^2 K_2) + K_{\mathrm{hrzn}}^2 \exp(-\alpha_2 K_1). && (58)
\end{aligned}$$

□

Proof of Lem. 2. From Prop. 9 by setting $K_1, K_2 = \Omega(\kappa)$ and $K_{\mathrm{cp}} = 2K_1K_2$. □

C.3 Many Probabilistic Pivots Imply One Combinatorial Pivot

Proof of Prop. 4. In slot t_k, there is exactly one block b produced by an honest node, and the block header is made public at the beginning of the slot, and is seen by all honest nodes within Δ_{h} time. Thereafter, each node has enough time to download $\widetilde{C}$ blocks during slots $[t_k, t_k + \nu]$.

Under the download rule $\mathcal{D}_{\mathsf{long}}$, all honest nodes download content for their longest header chain. If $D_k = 0$ *i.e.* an honest node did not download content for the block b before the end of slot $t_k + \nu$, then that honest node must download the content for at least $\widetilde{C}$ blocks on chains longer than the height of the block b or in the prefix of the block b. Since honest nodes produce blocks extending their longest chain, b extends $\mathrm{d}\mathcal{C}_p(t_k - 1)$ for some p. Let b^* be the block produced in slot t_i where $\mathsf{CPivot}(i)$ (suppose i exists). $\mathsf{CPivot}(i) \implies Y_i = 1$, therefore this block is unique, and also $t_k > t_i + \nu$. Due to Lem. 1, any valid header chain longer than b at time slot t_k must contain b^*. Therefore, the only blocks that are downloaded by an honest node during slots $[t_k, t_k + \nu]$

(1) must be produced after t_i because they extend b^*, and
(2) must be produced no later than t_k because there are no blocks produced in $(t_k, t_k + \nu]$.

In case a cpivot $i < k$ does not exist, the claim is trivial. □

Proposition 10.

$$\neg\mathsf{CPivot}(k) \implies \exists\, (i,j] \ni k \colon Y_{(i,j]} \leq 0. \qquad (59)$$

Proof. From Def. 3, $\neg\mathsf{CPivot}(k)$ implies that either there exists $i < k$ such that $Y_{(0,i]} \geq Y_{(0,k]}$ or there exists $j \geq k$ such that $Y_{(0,k]} > Y_{(0,j]}$. In the first case, $(i,k] \ni k$ and $Y_{(i,k]} \leq 0$. In the second case, $(k-1, j] \ni k$ and $Y_{(k-1,j]} \leq Y_{(k,j]} + 1 \leq 0$. □

Proposition 11. *If* $Y_{(i,j]} \leq 0$, *then*

$$\overline{D}_{(i,j]} \geq D_{(i,j]}, \qquad (60)$$

$$G_{(i,j]} - D_{(i,j]} \geq \frac{1}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right). \qquad (61)$$

Proof. Eqn. (60) follows from the definition $Y_i = D_i - \overline{D}_i$. Then,

$$\begin{aligned}
G_{(i,j]} + \overline{G}_{(i,j]} &= D_{(i,j]} + \overline{D}_{(i,j]} && (62)\\
G_{(i,j]} + \overline{G}_{(i,j]} &\geq 2D_{(i,j]} && (63)\\
2G_{(i,j]} - 2D_{(i,j]} &\geq G_{(i,j]} - \overline{G}_{(i,j]}. && (64)
\end{aligned}$$

□

Proposition 12. *If* $P_{(i,j]} > 0$, *then* $G_{(i,j]} - \overline{G}_{(i,j]} \geq P_{(i,j]}$.

Proof. Let $n = P_{(i,j]}$. First, consider the case $n = 1$. There is exactly one ppivot $k \in (i,j]$. From Def. 2, $X_{(0,i]} < X_{(0,j]}$. Therefore, $X_{(i,j]} > 0$, hence $G_{(i,j]} - \overline{G}_{(i,j]} \geq 1$.

For the general case, let $k_1, ..., k_n$ be the ppivots in $(i,j]$. Then, we can apply the $n = 1$ case on the disjoint intervals $(i, k_1]$, $(k_1, k_2]$, ..., $(k_{n-1}, j]$ and then sum them up.

This can also be seen from Fig. 9. Each ppivot corresponds to a height that the random walk X_k attains exactly once. This means that in any interval containing n ppivots, the random walk X_k 'moves up' by at least n units, and this is possible only if there are n more 'ups' than 'downs'. □

Lemma 6. *If all honest nodes use the download rule* $\mathcal{D}_{\mathsf{long}}$, *and if*

$$\forall\,(i,j] \geq K_{\mathrm{cp}}, i < K_{\mathrm{cp}} \colon \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) > Q_{(0,j]}, \text{ and} \qquad (65)$$

$$\frac{\widetilde{C}}{4} P_{(0,K_{\mathrm{cp}}]} > Q_{(0,2K_{\mathrm{cp}}]}, \qquad (66)$$

then $\exists k_1^* \in \left(0, K_{\mathrm{cp}}\right] : \mathsf{CPivot}(k_1^*)$.

Proof. Due to eqn. (66), there is at least one ppivot in $\left(0, K_{\mathrm{cp}}\right]$ (otherwise $P_{\left(0,K_{\mathrm{cp}}\right]} = 0$). Suppose for contradiction that there is no cpivot in $\left(0, K_{\mathrm{cp}}\right]$. Since cpivots are also ppivots, it is enough to consider that none of the ppivots is a cpivot. Then around each ppivot, there must be at least one interval which violates the combinatorial pivot condition. Formally, there is a set of intervals $\mathcal{I}$ such that:

$$\bigcup_{I \in \mathcal{I}} I \supseteq \left\{k \in \left(0, K_{\mathrm{cp}}\right] : \mathsf{PPivot}(k)\right\} \qquad (67)$$

$$\forall I \in \mathcal{I} \colon Y_I \leq 0 \qquad \text{(from Prop. 10).} \qquad (68)$$

Without loss of generality, each interval $I \in \mathcal{I}$ contains at least one ppivot (removing all intervals that do not contain a ppivot maintains eqns. (67) and (68)). Then if $(i,j] \in \mathcal{I}$, $i < K_{\mathrm{cp}}$.

First, let's consider the large intervals with $|I| \geq K_{\mathrm{cp}}$. Consider indices $k \in I$ for which $G_k = 1$ (good) but $D_k = 0$ ($\overline{D}$-slot). From Prop. 4, for each such index, all honest nodes download $\widetilde{C}$ blocks that are produced no later than t_k. The number of indices $k \in I$ with $G_k = 1$ and $D_k = 0$ is exactly $G_I - D_I$. For each such index, there must exist $\widetilde{C}$ distinct blocks produced in or before the interval I. Therefore if $I = (i,j]$,

$$\begin{aligned}
Q_{(0,j]} &\geq \widetilde{C}\left(G_{(i,j]} - D_{(i,j]}\right) && (69)\\
&\geq \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) \quad \text{(from Prop. 11).} && (70)
\end{aligned}$$

This is a contradiction to eqn. (65).

Therefore all intervals $I \in \mathcal{I}$ are small ($|I| < K_{\mathrm{cp}}$). Then for each $I \in \mathcal{I}, I \subset \left(0, 2K_{\mathrm{cp}}\right]$. Also,

$$\begin{aligned}
G_I - D_I &\geq \frac{1}{2}\left(G_I - \overline{G}_I\right) \quad \text{(from Prop. 11)} && (71)\\
&\geq \frac{1}{2}P_I \quad \text{(from Prop. 12).} && (72)
\end{aligned}$$

Consider the indices $k \in \left(0, 2K_{\mathrm{cp}}\right]$ with $G_k = 1$ and $D_k = 0$. Let $\mathcal{I}_k = \{I \in \mathcal{I} : k \in I\}$ be the set of intervals that contain k. Let I_k^L be an interval in $\mathcal{I}_k$ that stretches farthest to the left, and let I_k^R be an interval that stretches farthest to the right (these may also be the same). Note that all other intervals in $\mathcal{I}_k$ are contained in $I_k^L \cup I_k^R$. Therefore, all intervals in $\mathcal{I}_k$ except I_k^L and I_k^R can be removed from

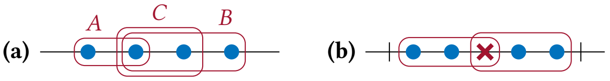


Figure 11: Blue circles represent ppivots, red crosses represent indices with $G_k = 1$ and $D_k = 0$. (a) Given intervals A, B, C all containing the 2nd blue circle from left, interval C is redundant. (b) Given n blue circles, the adversary needs at least $n/4$ red crosses to draw a set of intervals satisfying eqns. (67) **and** (68)**. Here is a placement of red crosses relative to blue circles that achieves the minimum number of red crosses.**

$\mathcal{I}$ while maintaining eqns. (67) and (68) (see Fig. 11(a)). This process is repeated for all $k \in \big(0, 2K_{\text{cp}}\big]$ with $G_k = 1$ and $D_k = 0$, so that in the resulting set $\mathcal{I}$, each such index k is contained in at most two intervals. Then,

$$\sum_{k\in\left(0,2K_{\text{cp}}\right]:\, G_k=1, D_k=0} |\mathcal{I}_k| \le \sum_{k\in\left(0,2K_{\text{cp}}\right]:\, G_k=1, D_k=0} 2 \tag{73}$$

$$= 2\left(G_{\left(0,2K_{\text{cp}}\right]} - D_{\left(0,2K_{\text{cp}}\right]}\right). \tag{74}$$

This sum can be rewritten as

$$\sum_{k\in\left(0,2K_{\text{cp}}\right]:\, G_k=1, D_k=0} |\mathcal{I}_k| = \sum_{I\in\mathcal{I}} (G_I - D_I) \tag{75}$$

$$\ge \sum_{I\in\mathcal{I}} \frac{1}{2} P_I \tag{76}$$

$$\ge \frac{1}{2} P_{\left(0,K_{\text{cp}}\right]} \quad \text{(due to eqn. (67))}. \tag{77}$$

Therefore,

$$G_{\left(0,2K_{\text{cp}}\right]} - D_{\left(0,2K_{\text{cp}}\right]} \ge \frac{1}{4} P_{\left(0,K_{\text{cp}}\right]}. \tag{78}$$

This can also be seen from Fig. 11(b).

Finally, as shown before, for each k with $G_k = 1$ and $D_k = 0$, all honest nodes download at least $\widetilde{C}$ distinct blocks produced in or before index k (Prop. 4). This gives

$$Q_{\left(0,2K_{\text{cp}}\right]} \ge \widetilde{C}\left(G_{\left(0,2K_{\text{cp}}\right]} - D_{\left(0,2K_{\text{cp}}\right]}\right) \tag{79}$$

$$\ge \frac{\widetilde{C}}{4} P_{\left(0,K_{\text{cp}}\right]} \tag{80}$$

which is a contradiction to eqn. (66). □

Proof of Lem. 3. This will be proved through induction. For the base case ($m = 0$), Lem. 6 shows that $\exists k_1^* \in \big(0, K_{\text{cp}}\big] : \mathsf{CPivot}(k_1^*)$.

For $m \ge 1$, assume that $\exists k_{m-1}^* \in \big((m-1)K_{\text{cp}}, mK_{\text{cp}}\big]$ such that $\mathsf{CPivot}(k_{m-1}^*)$. Now we want to show that $\exists k_m^* \in \big(mK_{\text{cp}}, (m+1)K_{\text{cp}}\big]$ such that $\mathsf{CPivot}(k_m^*)$. Suppose for contradiction that there is no cpivot in $\big(mK_{\text{cp}}, (m+1)K_{\text{cp}}\big]$. As in the proof of Lem. 6, there is a set of intervals $\mathcal{I}$ such that:

$$\bigcup_{I\in\mathcal{I}} I \supseteq \left\{k \in \big(mK_{\text{cp}}, (m+1)K_{\text{cp}}\big] : \mathsf{PPivot}(k)\right\} \tag{81}$$

$$\forall I \in \mathcal{I}: Y_I \le 0. \tag{82}$$

Without loss of generality, each interval $I \in \mathcal{I}$ contains at least one ppivot. Then if $(i, j] \in \mathcal{I}$, $i < (m+1)K_{\text{cp}}$ and $j > mK_{\text{cp}}$.

First, consider the large intervals with $|I| \ge K_{\text{cp}}$. Consider indices $k \in I$ for which $G_k = 1$ (good) but $D_k = 0$ ($\overline{D}$-slot). From Prop. 4, for each such index k, all honest nodes download $\widetilde{C}$ blocks that are produced in the interval $\big(k_{m-1}^*, k\big]$. The number of indices $k \in I$ with $G_k = 1$ and $D_k = 0$ is exactly $G_I - D_I$. For each such index, there must exist $\widetilde{C}$ distinct blocks from distinct BPOs that are downloaded by honest nodes. Therefore if $I = (i, j]$,

$$Q_{(k_{m-1}^*, j]} \ge \widetilde{C}\left(G_{(i,j]} - D_{(i,j]}\right) \tag{83}$$

$$\ge \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) \quad \text{(from Prop. 11)}. \tag{84}$$

But $k_{m-1}^* > (m-1)K_{\text{cp}}$ and $i < (m+1)K_{\text{cp}}$. Therefore $Q_{(k_{m-1}^*, j]} \le Q_{(i-2K_{\text{cp}}, j]}$. Then we have a contradiction to eqn. (13).

Therefore all intervals $I \in \mathcal{I}$ are small ($|I| < K_{\text{cp}}$). Then for each $I \in \mathcal{I}$, $I \subset \big((m-1)K_{\text{cp}}, (m+1)K_{\text{cp}}\big]$. Also,

$$G_I - D_I \ge \frac{1}{2}\left(G_I - \overline{G}_I\right) \ge \frac{1}{2} P_I \quad \text{(from Props. 11 and 12)} \tag{85}$$

Consider the indices $k \in \big((m-1)K_{\text{cp}}, (m+1)K_{\text{cp}}\big]$ with $G_k = 1$ and $D_k = 0$. Following the arguments in the proof of Lem. 6, we can reduce the set $\mathcal{I}$ so that in the resulting set $\mathcal{I}$, each such index k is contained in at most two intervals. Then,

$$\sum_{k\in\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]:\, G_k=1, D_k=0} |\mathcal{I}_k|$$

$$\le 2\left(G_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]} - D_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]}\right). \tag{86}$$

This sum can be rewritten as

$$\sum_{k\in\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]:\, G_k=1, D_k=0} |\mathcal{I}_k| = \sum_{I\in\mathcal{I}} (G_I - D_I) \tag{87}$$

$$\ge \sum_{I\in\mathcal{I}} \frac{1}{2} P_I \tag{88}$$

$$\ge \frac{1}{2} P_{\left(mK_{\text{cp}},(m+1)K_{\text{cp}}\right]}. \tag{89}$$

Therefore,

$$G_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]} - D_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]} \ge \frac{1}{4} P_{\left(mK_{\text{cp}},(m+1)K_{\text{cp}}\right]}. \tag{90}$$

Finally, for each k with $G_k = 1$ and $D_k = 0$, all honest nodes download at least $\widetilde{C}$ distinct blocks produced in or before the most recent cpivot before $(m-1)K_{\text{cp}}$. By induction assumption, we have a cpivot $k_{m-2}^* \in \big((m-2)K_{\text{cp}}, (m-1)K_{\text{cp}}\big]$. This gives

$$Q_{\left((m-2)K_{\text{cp}},(m+1)K_{\text{cp}}\right]} \ge \widetilde{C}\left(G_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]} - D_{\left((m-1)K_{\text{cp}},(m+1)K_{\text{cp}}\right]}\right) \tag{91}$$

$$\ge \frac{\widetilde{C}}{4} P_{\left(mK_{\text{cp}},(m+1)K_{\text{cp}}\right]} \tag{92}$$

which is a contradiction. □

D PROOF-OF-WORK SECURITY PROOFS

Proposition 13.

$$\forall k, K \in \mathbb{N}: \ \Pr\left[\tau(t_{k+K} - t_k) \ge \frac{K}{\lambda(1-\delta)}\right] \le \exp\left(-\frac{K\delta^2}{2(1+\delta)}\right) \tag{93}$$

Proof. This results from a Poisson tail bound for the number of BPOs in real time K/λ, and noting that each non-empty slot has exactly one BPO. □

Lemma 7. *If for some $K_{\mathrm{cp}} > 0$,*

$$\forall m \geq 0 \colon \exists k^*_m \in \left(mK_{\mathrm{cp}}, (m+1)K_{\mathrm{cp}}\right] : \mathsf{CPivot}(k^*_m), \tag{94}$$

then the PoW longest chain protocol $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ with $k_{\mathrm{conf}} = 2K_{\mathrm{cp}} + 1$ satisfies safety. Further, if

$$\forall k \in \mathbb{N}, K \geq K_{\mathrm{cp}} \colon t_{k+K} - t_k < \frac{K}{\lambda\tau(1-\delta)}, \tag{95}$$

then it also satisfies liveness with $T_{\mathrm{live}} = \frac{6K_{\mathrm{cp}}+2}{\lambda\tau(1-\delta)}$.

Proof. Safety: For an arbitrary slot t, let k be the largest index such that $t_k \leq t$. From eqn. (94), every interval of $2K_{\mathrm{cp}}$ indices contains at least one cpivot. Therefore, there exists $k^* \in \left(k - 2K_{\mathrm{cp}} - 1, k - 1\right]$ such that $\mathsf{CPivot}(k^*)$. Let b^* be the block from index k^*. Due to Lem. 1, for all honest nodes p, q and $t' \geq t$, $b^* \in \mathrm{d}\mathcal{C}_p(t)$ and $b^* \in \mathrm{d}\mathcal{C}_q(t')$. But $k^* \geq k - k_{\mathrm{conf}}$, so the block b^* cannot be k_{conf}-deep in any chain at slot t Therefore, LOG^t_p is a prefix of b^* which in turn is a prefix of $\mathrm{d}\mathcal{C}_q(t')$. We can thus conclude that either $\mathsf{LOG}^t_p \preceq \mathsf{LOG}^{t'}_q$ or $\mathsf{LOG}^{t'}_q \preceq \mathsf{LOG}^t_p$. Therefore, safety holds.

Liveness: Assume a transaction tx is received by all honest nodes before slot t. Again let k be the largest index such that $t_k \leq t$. We know that there exists $k^* \in (k, k + 2K_{\mathrm{cp}}]$ such that $\mathsf{CPivot}(k^*)$. The honest block b^* from index k^* or its prefix must contain tx since tx is seen by all honest nodes at time $t < t_{k^*}$. Since k^* is a cpivot, for all $(i, j] \ni k^*$, $D_{(i,j]} > \overline{D}_{(i,j]}$ (Def. 3 and eqn. (6)), and hence $D_{(i,j]} > \frac{j-i}{2}$. Particularly,

$$\begin{aligned} D_{(k^*-1,k^*+2k_{\mathrm{conf}}-1]} &> k_{\mathrm{conf}} &(96)\\ \implies D_{(k^*,k^*+2k_{\mathrm{conf}}-1]} &> k_{\mathrm{conf}} - 1. &(97)\end{aligned}$$

Then from Prop. 5,

$$\begin{aligned} L_{\min}(t_{k^*+2k_{\mathrm{conf}}-1} + \nu) - L_{\min}(t_{k^*+1} - 1) &\geq D_{(k^*,k^*+2k_{\mathrm{conf}}-1]} \\ &\geq k_{\mathrm{conf}}. &(98)\end{aligned}$$

Due to Lem. 1, $b^* \in \mathrm{d}\mathcal{C}_p(t')$ for all honest nodes p and $t' \geq t_{k^*} + \nu$, and $L_{\min}(t_{k^*+1} - 1) \geq |b^*|$. This means that b^* is k_{conf}-deep in $\mathrm{d}\mathcal{C}_p(t')$ for all honest nodes p and all $t' \geq t_{k^*+2k_{\mathrm{conf}}-1} + \nu$. Finally, with $k^* \leq k + 2K_{\mathrm{cp}}$ and eqn. (95),

$$\begin{aligned} t_{k^*+2k_{\mathrm{conf}}-1} + \nu - t &\leq t_{k+6K_{\mathrm{cp}}+1} + \nu - t_k \\ &\leq t_{k+6K_{\mathrm{cp}}+2} - t_k \\ &< \frac{6K_{\mathrm{cp}}+2}{\lambda\tau(1-\delta)}. &(99)\end{aligned}$$

Therefore, $\mathrm{tx} \in \mathsf{LOG}^{t'}_p$ for all $t' \geq t + T_{\mathrm{live}}$. □

Proof of Thm. 1. First, we show that the conditions of Lem. 3 hold, and therefore cpivots occur. Define the event

$$\mathcal{E}_1 = \left\{\forall\, (i,j] \geq K_{\mathrm{cp}} \colon P_{(i,j]} > (1-\delta)p_{\mathrm{ppivot}}(j-i)\right\} \tag{100}$$

Suppose that $\mathcal{E}_1$ occurs, and $\frac{\widetilde{C}}{16}p_{\mathrm{ppivot}}(1-\delta) > 1$. Then,

$$\begin{aligned} \forall\, (i,j] \geq K_{\mathrm{cp}} \colon \quad \frac{\widetilde{C}}{4}P_{(i,j]} &> \frac{\widetilde{C}}{4}(1-\delta)p_{\mathrm{ppivot}}(j-i) &(101)\\ &> 4(j-i) &(102)\\ &\overset{(a)}{=} Q_{(i-2K_{\mathrm{cp}},j+K_{\mathrm{cp}}]} &(103)\end{aligned}$$

where (a) is because as $\tau \longrightarrow 0$, each non-empty slot has exactly one BPO. This satisfies eqn. (14) in Lem. 3. Further,

$$\begin{aligned} \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) &\geq \frac{\widetilde{C}}{2}P_{(i,j]} &(104)\\ &> 3(j-i) &(105)\\ &> Q_{(i-2K_{\mathrm{cp}},j]} &(106)\end{aligned}$$

which satisfies condition eqn. (13) in Lem. 3. Therefore there is one cpivot in every interval of the form $\left(mK_{\mathrm{cp}}, (m+1)K_{\mathrm{cp}}\right]$. Also suppose the following event occurs:

$$\mathcal{E}_2 = \left\{\forall k \in \mathbb{N}, K \geq K_{\mathrm{cp}} \colon t_{k+K} - t_k < \frac{K}{\lambda\tau(1-\delta)}\right\}. \tag{107}$$

Then Lem. 7 guarantees safety and liveness with $k_{\mathrm{conf}} = 2K_{\mathrm{cp}}$ and $T_{\mathrm{live}} = \frac{6K_{\mathrm{cp}}+2}{\lambda\tau(1-\delta)}$.

By choosing $K_{\mathrm{cp}} = \Omega(\kappa^2)$, $K_{\mathrm{hrzn}} = \mathrm{poly}(\kappa)$, and using Lem. 2, Prop. 13,

$$\begin{aligned} \Pr\left[\neg\mathcal{E}_1\right] &= \mathrm{negl}(\kappa) &(108)\\ \Pr\left[\neg\mathcal{E}_2\right] &\leq K^2_{\mathrm{hrzn}} e^{-K_{\mathrm{cp}}\delta^2/(2(1+\delta))} = \mathrm{negl}(\kappa). &(109)\end{aligned}$$

By a union bound, the probability of failure of either $\mathcal{E}_1$ or $\mathcal{E}_2$ is $\mathrm{negl}(\kappa)$. Finally, indices are mapped to real time as $T^{\mathrm{real}}_{\mathrm{live}} \triangleq T_{\mathrm{live}}\tau$.

Finally, we take the limit $\tau \longrightarrow 0$. With the relations $\lambda = \rho/\tau$, $(\nu+1)\tau \geq \Delta_{\mathrm{h}} + \widetilde{C}/C$, and $p_{\mathrm{ppivot}} = (2p_{\mathrm{G}} - 1)^2/p_{\mathrm{G}}$,

$$\begin{aligned} p_{\mathrm{G}} = (1-\beta)\frac{\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}} &\rightarrow (1-\beta)e^{-\lambda\left(\Delta_{\mathrm{h}}+\widetilde{C}/C\right)}, &(110)\\ \frac{\widetilde{C}}{16}\frac{(2p_{\mathrm{G}}-1)^2}{p_{\mathrm{G}}}(1-\delta) &> 1 &(111)\end{aligned}$$

Note that $\widetilde{C}$ is an analysis parameter whose value is arbitrarily. To find the maximum block production rate λ that the protocol can achieve, we should optimize over $\widetilde{C}$. To find the maximum achievable λ, we can take $\delta \longrightarrow 0$ as we can increase the latency through increasing K_{cp} to still satisfy the error bounds. Then solving for p_{G} from eqn. (111),

$$p_{\mathrm{G}} > \frac{\widetilde{C} + 4 + \sqrt{8\widetilde{C} + 16}}{2\widetilde{C}}. \tag{112}$$

Then from eqn. (110),

$$\lambda\left(\Delta_{\mathrm{h}} + \widetilde{C}/C\right) > \ln\left(\frac{2(1-\beta)\widetilde{C}}{\widetilde{C} + 4 + \sqrt{8\widetilde{C} + 16}}\right). \tag{113}$$

Maximizing over $\widetilde{C}$ gives the resulting threshold. □

E PROOF-OF-STAKE

E.1 Pseudocodes for Equivocation Removal

Algs. 5 and 6

E.2 Security Proofs

Proposition 14. *For all $\delta \in (0,1)$, $k, K \in \mathbb{N}$,*

$$\Pr\left[t_{k+K} - t_k \geq \frac{K/(1-e^{-\rho})}{1-\delta}\right] \leq e^{-2K(1-e^{-\rho})\delta^2}, \tag{114}$$

Proof. This results from a Hoeffding bound for the number of non-empty slots in $K/(1-e^{-\rho})$ slots. □

Algorithm 5 PoS LC consensus protocol $\Pi_{\mathrm{SaPoS}}^{\rho,\tau,k_{\mathrm{conf}},k_{\mathrm{epf}}}$ with download logic and equivocation removal (helper functions: App. A.1, environment $\mathcal{Z}$: App. A.2, functionality $\mathcal{F}'^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$: Alg. 6)

```
▷ Global counter of time slots $t \leftarrow 1, 2, \ldots$ of duration $\tau$ (for PoW: $\tau \longrightarrow 0$, cf. Sec. 5)
▷ Same as in Alg. 1: INIT(genesis$C$, genesisTxs), RECEIVEDHEADERCHAIN($C$), RECEIVEDCONTENT($C$, txs)
▷ $(C \overset{BPO}{\equiv} C') \triangleq (C \neq C') \wedge (C.\mathrm{node} = C'.\mathrm{node}) \wedge (C.\mathrm{time} = C'.\mathrm{time})$
at time slot $t \leftarrow 1, 2, \ldots$                                        ▷ LC protocol main loop
    txs $\leftarrow \mathcal{Z}$.RECEIVEPENDINGTXS()
    ▷ Construct equivocation proofs against headers in prefix not already proven
    eqProofs $\leftarrow \{(C_1 \preceq \mathrm{d}C, C_2 \in \mathrm{h}\mathcal{T}) \mid (C_1 \overset{\mathrm{BPO}}{\equiv} C_2) \wedge (|C_1| > |\mathrm{d}C| - k_{\mathrm{epf}}) \wedge ((C_1, C_2) \notin \bigcup_{C \preceq \mathrm{d}C} C.\mathrm{eqProofs})\}$
    ▷ Produce and disseminate a new block if eligible
    if $C' \neq \perp$ with $C' \leftarrow \mathcal{F}_{\mathrm{hdrtree}}$.EXTEND(d$C$, txs, eqProofs)
        $\mathcal{Z}$.UPLOADCONTENT($C'$, txs)
        $\mathcal{Z}$.BROADCASTHEADERCHAIN($C'$)
    ▷ Blank the content of blocks against which there was an equivocation proof
    blkTxs$'[C] \leftarrow (\emptyset$ if $(C, \_) \in \bigcup_{C' \preceq C^*} C'.\mathrm{eqProofs}$, else blkTxs$[C])$
    ▷ Confirm all but the last $k_{\mathrm{conf}}$ blocks on the longest downloaded chain
    LOG$^t \leftarrow$ txsLedger(blkTxs, $C^{\lceil k_{\mathrm{conf}}}$)   ▷ Ledger of node $p$ at time $t$: LOG$_p^t$
do throughout
    Choose $C$ from download rule (e.g. Alg. 4)
    if $\exists C' \in \mathrm{h}\mathcal{T}$: $C' \overset{\mathrm{BPO}}{\equiv} C$
        blkTxs$[C] \leftarrow \emptyset$                  ▷ don't download $C$, query download rule again
    else
        Download content for $C$
```

Algorithm 6 Idealized functionality $\mathcal{F}'^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$: block production lottery and header chain structure for PoS (helper functions: App. A.1)

```
▷ INIT(genesis$C$, numNodes) and VERIFY($C$) same as in Alg. 2
▷ ISLEADER($P$, $t$) same as in Alg. 3
on EXTEND($t'$, $C$, txs, eqProofs) from $\mathcal{A}$ (from adversarial node $P$) or $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$
    ▷ New header chain is valid if parent chain $C$ is valid, $P$ is leader for slot $t'$, and $t'$ is later than the tip of $C$ and is not in the future
    if $(C \in \mathcal{T}) \wedge \mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$.ISLEADER($P$, $t'$) $\wedge (C.\mathrm{time} < t' \leq t)$
        ▷ Check equiv. pfs. are valid, point to 'recent' headers, and do not repeat
        if $\forall (C_1, C_2) \in \mathrm{eqProofs}: (C_1 \preceq C) \wedge (C_2 \in \mathcal{T}) \wedge (C_1 \overset{\mathrm{BPO}}{\equiv} C_2) \wedge (|C_1| > |C| - k_{\mathrm{epf}}) \wedge ((C_1, C_2) \notin \bigcup_{C' \preceq C} C'.\mathrm{eqProofs})\}$
            ▷ Produce a new block header extending $C$
            $C' \leftarrow C \|$ newBlock(time: $t'$, node: $P$, txsHash: Hash(txs))
            $\mathcal{T} \leftarrow \mathcal{T} \cup \{C'\}$                ▷ Register new header chain in header tree
            return $C'$
    return $\perp$
on EXTEND($C$, txs, eqProofs) from node $P$ (possibly adversarial) at time slot $t$
    return $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$.EXTEND($t$, $C$, txs, eqProofs)
```

Lemma 8. *If for some $K_{\mathrm{cp}} > 0$,*

$$\forall m \geq 0 \colon \exists k_m^* \in \left(mK_{\mathrm{cp}}, (m+1)K_{\mathrm{cp}}\right] : \mathrm{CPivot}(k_m^*), \tag{115}$$

$$\forall k \in \mathbb{N}, K \geq K_{\mathrm{cp}} \colon t_{k+K} - t_k < \frac{K/(1-e^{-\rho})}{1-\delta}, \tag{116}$$

then SaPoS with $k_{\mathrm{conf}} = 6K_{\mathrm{cp}} + 1$ and $k_{\mathrm{epf}} = 4K_{\mathrm{cp}}$ satisfies safety and liveness with $T_{\mathrm{live}} = \frac{14K_{\mathrm{cp}}+2}{(1-e^{-\rho})(1-\delta)}$.

Proof. First, we prove safety. Consider arbitrary slots $t \leq t'$ and let h be the largest index such that $t_h \leq t$. Consider a block $b_i \in \mathrm{d}C_p(t)^{\lceil k_{\mathrm{conf}}}$ which was produced in index $i \leq h - k_{\mathrm{conf}}$. From eqn. (115), every interval of $2K_{\mathrm{cp}}$ indices contains at least one cpivot. Therefore for any i, there exist cpivots j, k such that

$$i < j < k \leq i + 4K_{\mathrm{cp}}. \tag{117}$$

Also, let l be the last cpivot before (excluding) index h. Then

$$l \geq h - 2K_{\mathrm{cp}} \geq i + k_{\mathrm{conf}} - 2K_{\mathrm{cp}} > i + 4K_{\mathrm{cp}}. \tag{118}$$

From eqn. (117) and eqn. (118), we have

$$i < j < k \leq i + k_{\mathrm{epf}} < l < h. \tag{119}$$

These are shown in Fig. 12. Let b_j, b_k, b_l be the blocks corresponding to the respective cpivots (see Fig. 12). Due to Lem. 1 and $t \geq t_h > t_l + \nu$,

$$b_i \preceq b_j \preceq b_k \preceq b_l \preceq \mathrm{d}C_p(t) \cap \mathrm{d}C_q(t'). \tag{120}$$

Since the above holds for all $b_i \in \mathrm{d}C_p(t)^{\lceil k_{\mathrm{conf}}}$, we obtain that $\mathrm{d}C_p(t)^{\lceil k_{\mathrm{conf}}} \preceq \mathrm{d}C_q(t')$. We can thus conclude that

$$\mathrm{d}C_p(t)^{\lceil k_{\mathrm{conf}}} \overset{\preceq}{\succeq} \mathrm{d}C_q(t')^{\lceil k_{\mathrm{conf}}} \tag{121}$$

where $C_1 \overset{\preceq}{\succeq} C_2$ denotes $C_1 \preceq C_2$ or $C_2 \preceq C_1$. Due to eqn. (121), the k_{conf}-deep header chains of p at t and of q at t' are consistent. Without equivocation removal, this was enough to show safety of the corresponding ledgers. Now to show that the two ledgers LOG_p^t and $\mathrm{LOG}_q^{t'}$ are consistent, we only need to show that if the content of a block is blanked in LOG_p^t, it is also blanked in $\mathrm{LOG}_q^{t'}$, and conversely if it is not blanked in LOG_p^t, it is not blanked in $\mathrm{LOG}_q^{t'}$.

Suppose that the content of b_i is blanked in LOG_p^t. This means that either there was an equivocation for b_i in node p's view (hence node p did not download the content), or there is an equivocation proof against b_i in a header in $\mathrm{d}C_p(t)$. The header of b_i must be seen by all honest nodes p before the end of slot $t_j + \nu$ (since $b_j \in \mathrm{d}C_p(t_j + \nu)$). Then since block b_k is honest, $t_k > t_j + \nu$, and $k \leq i + k_{\mathrm{epf}}$, either b_k or another block in its prefix must include an equivocation proof against b_i. We know that $b_k \in \mathrm{d}C_q(t')$, so the content of block b_i will be blanked in $\mathrm{LOG}_q^{t'}$ as well.

Suppose that the content of b_i is not blanked in LOG_p^t. This means that there is no equivocation proof against b_i in $\mathrm{d}C_p(t)$. Since $l \geq h - 2K_{\mathrm{cp}}$, the block b_l cannot be more than $2K_{\mathrm{cp}}$-deep in $\mathrm{d}C_p(t)$, *i.e.*,

$$|b_l| \geq \left|\mathrm{d}C_p(t)\right| - 2K_{\mathrm{cp}}. \tag{122}$$

But b_i is k_{conf}-deep in $\mathrm{d}C_p(t)$ (as assumed), so

$$|b_i| \leq \left|\mathrm{d}C_p(t)\right| - k_{\mathrm{conf}}. \tag{123}$$

Together, we have

$$|b_l| \geq |b_i| + k_{\mathrm{conf}} - 2K_{\mathrm{cp}} > |b_i| + k_{\mathrm{epf}}. \tag{124}$$

Therefore b_l or any block extending it cannot contain an equivocation proof against b_i. Since $b_l \in \mathrm{d}C_p(t)$ and $b_l \in \mathrm{d}C_q(t')$, there cannot be any block in $\mathrm{d}C_q(t')$ before b_l, that is not in $\mathrm{d}C_p(t)$. Therefore, there is no equivocation proof against b_i in $\mathrm{d}C_q(t')$. Also, node q must have downloaded block b_i, otherwise there must have been an equivocation proof in b_k or its prefix as discussed in the previous paragraph. So, the content of b_i is not blanked in $\mathrm{LOG}_q^{t'}$. We can thus conclude that either $\mathrm{LOG}_p^t \preceq \mathrm{LOG}_q^{t'}$ or $\mathrm{LOG}_q^{t'} \preceq \mathrm{LOG}_p^t$. Therefore, safety holds.

We next prove liveness. Assume a transaction tx is received by all honest nodes before slot t. Again let h be the largest index such that $t_h \leq t$. We know that there exists $k^* \in (h, h + 2K_{\mathrm{cp}}]$ such that $\mathrm{CPivot}(k^*)$. The honest block b^* from index k^* or its prefix must

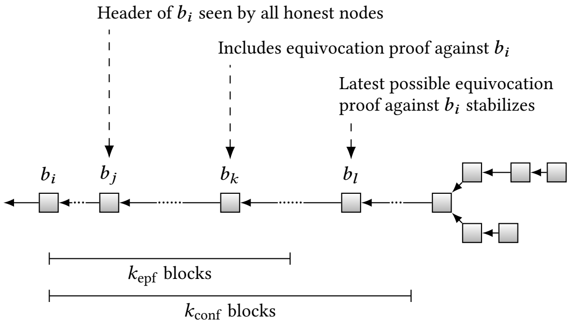

Header of b_i seen by all honest nodes
Includes equivocation proof against b_i
Latest possible equivocation proof against b_i stabilizes
b_i
b_j
b_k
b_l
k_{epf} blocks
k_{conf} blocks

Figure 12: Illustration for the proof of Lem. 8. Consider an arbitrary block b_i that is k_{conf}-deep in the longest chain of a node. Indices j, k, l are cpivots. Since cpivots stabilize, the corresponding blocks b_j, b_k, b_l are in all honest nodes' longest chains. At cpivot j, we know for sure that all honest nodes saw the header of b_i because they saw the header for b_j. At cpivot k, we know for sure that if b_i had an equivocation, then an equivocation proof against b_i must have entered the chain. At cpivot l, we know for sure that the last block that can add an equivocation proof against b_i has stabilized (as the deadline of k_{epf} blocks has passed). Thus, a ledger formed from k_{conf}-deep blocks (sufficient to obtain three cpivots) will remain safe.

contain tx since tx is seen by all honest nodes at time $t < t_{k^*}$. Since k^* is a cpivot, for all $(i, j] \ni k^*$, $D_{(i,j]} > \overline{D}_{(i,j]}$ (Def. 3 and eqn. (6)), and hence $D_{(i,j]} > \frac{j-i}{2}$. Particularly,

$$\begin{aligned} & D_{(k^*-1,k^*+2k_{\text{conf}}-1]} > k_{\text{conf}} && (125)\\ \implies\ & D_{(k^*,k^*+2k_{\text{conf}}-1]} > k_{\text{conf}} - 1. && (126) \end{aligned}$$

Then from Prop. 5,

$$\begin{aligned} L_{\min}(t_{k^*+2k_{\text{conf}}-1} + \nu) - L_{\min}(t_{k^*+1} - 1) &\geq D_{(k^*,k^*+2k_{\text{conf}}-1]}\\ &\geq k_{\text{conf}}. \qquad (127) \end{aligned}$$

Due to Lem. 1, $b^* \in \mathrm{d}\mathcal{C}_p(t')$ for all honest nodes p and $t' \geq t_{k^*} + \nu$, and $L_{\min}(t_{k^*+1} - 1) \geq |b^*|$. This means that b^* is k_{conf}-deep in $\mathrm{d}\mathcal{C}_p(t')$ for all honest nodes p and all $t' \geq t_{k^*+2k_{\text{conf}}-1} + \nu$. Further, the content of an honest block will never be blanked out in any honest node's ledger. Finally, with $k^* \leq h + 2K_{\text{cp}}$ and eqn. (95),

$$\begin{aligned} t_{k^*+2k_{\text{conf}}-1} + \nu - t &\leq t_{h+2K_{\text{cp}}+2k_{\text{conf}}-1} + \nu - t_h\\ &\leq t_{h+2K_{\text{cp}}+2k_{\text{conf}}} - t_h\\ &< \frac{2K_{\text{cp}} + 2k_{\text{conf}}}{(1-e^{-\rho})(1-\delta)}. \qquad (128) \end{aligned}$$

Therefore, tx $\in \mathrm{LOG}_p^{t'}$ for all $t' \geq t + T_{\text{live}}$ with T_{live} as in the lemma statement. □

We also need another proposition to bound the number of BPOs in a given number of slots, in order to bound $Q_{(i,j]}$.

Proposition 15.

$$\forall t, T \in \mathbb{N}\colon\ \Pr\left[\sum_{r=t}^{t+T} (H_t + A_t) \geq \rho T(1+\delta)\right] \leq \exp\left(-\frac{\rho T \delta^2}{2(1+\delta)}\right). \quad (129)$$

Proof. This results from a Poisson tail bound since $\sum_{r=t}^{t+T}(H_t + A_t) \sim \mathrm{Poisson}(\rho T)$. □

Proof of Thm. 2. First, we show that the conditions of Lem. 3 hold, and therefore cpivots occur. Suppose that the following three events occur.

$$\mathcal{E}_1 = \left\{\forall\,(i,j] \geq K_{\text{cp}}\colon P_{(i,j]} > (1-2\delta) p_{\text{ppivot}}(j-i)\right\}, \qquad (130)$$

$$\mathcal{E}_2 = \left\{\forall t \in \mathbb{N}, T \geq \frac{K_{\text{cp}}}{1-e^{-\rho}}\colon \sum_{r=t}^{t+T}(H_t + A_t) < \rho T(1+\delta)\right\}, \quad (131)$$

$$\mathcal{E}_3 = \left\{\forall k \in \mathbb{N}, K \geq K_{\text{cp}}\colon t_{k+K} - t_k < \frac{K/(1-e^{-\rho})}{1-\delta}\right\}. \qquad (132)$$

From $\mathcal{E}_2$ and $\mathcal{E}_3$, we get

$$\begin{aligned} \forall\,(i,j] \geq K_{\text{cp}}\colon\quad Q_{(i,j]} &\triangleq \sum_{k=i+1}^{j} (H_{t_k} + A_{t_k}) && (133)\\ \text{with } T = \frac{j-i}{(1-e^{-\rho})(1-\delta)},\quad &\leq \sum_{t=t_i}^{t_i+T}(H_t + A_t) && (134)\\ &< \frac{\rho(j-i)(1+\delta)}{(1-e^{-\rho})(1-\delta)} && (135)\\ &\leq \frac{\rho(j-i)}{(1-e^{-\rho})(1-2\delta)}. && (136) \end{aligned}$$

Then if $\frac{\widetilde{C}}{16} p_{\text{ppivot}}(1-4\delta) > \frac{\rho}{(1-e^{-\rho})}$,

$$\begin{aligned} \forall\,(i,j] \geq K_{\text{cp}}\colon\quad \frac{\widetilde{C}}{4} P_{(i,j]} &> \frac{\widetilde{C}}{4}(1-2\delta) p_{\text{ppivot}}(j-i) && (137)\\ &> \frac{4\rho(j-1)(1-2\delta)}{(1-e^{-\rho})(1-4\delta)} && (138)\\ &> \frac{4\rho(j-1)}{(1-e^{-\rho})(1-2\delta)} && (139)\\ &> Q_{(i-2K_{\text{cp}},j+K_{\text{cp}}]}. && (140) \end{aligned}$$

This satisfies eqn. (14) in Lem. 3. Further,

$$\begin{aligned} \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) &\geq \frac{\widetilde{C}}{2} P_{(i,j]} && (141)\\ &> \frac{3\rho(j-1)}{(1-e^{-\rho})(1-2\delta)} && (142)\\ &> Q_{(i-2K_{\text{cp}},j]} && (143) \end{aligned}$$

which satisfies condition eqn. (13) in Lem. 3. Therefore there is one cpivot in every interval of the form $\left(mK_{\text{cp}}, (m+1)K_{\text{cp}}\right]$. Then by Lem. 8, the protocol achieves safety and liveness with appropriately chosen $k_{\text{conf}}, k_{\text{epf}}, T_{\text{live}}$.

By using Lem. 2, $K_{\text{cp}} = \Omega(\kappa^2)$, $K_{\text{hrzn}} = \mathrm{poly}(\kappa)$, Props. 14 and 15, and union bounds, the probability of failure of either $\mathcal{E}_1$, $\mathcal{E}_2$ or $\mathcal{E}_3$ is $\mathrm{negl}(\kappa)$.

The required security threshold is obtained from $(\nu+1)\tau \geq \Delta_{\mathrm{h}} + \widetilde{C}/C$, $p_{\mathrm{G}} = (1-\beta)\frac{\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}}$, $\frac{C}{16} p_{\text{ppivot}} > \frac{\rho}{1-e^{-\rho}}$, and $p_{\text{ppivot}} = (2p_{\mathrm{G}} - 1)^2/p_{\mathrm{G}}$,. As in the case of PoW, $\widetilde{C}$ is a free parameter that can be optimized to find the best set of parameters. □

F TRANSACTION VALIDITY PROOFS

Proof of Lem. 4. In Sec. 6.1, we have that equivocation proofs against a block need to be included within the next k_{epf} blocks. A node creating a block thus knows all equivocation proofs that will ever be included in their header chain against blocks that are k_{epf}-deep, thus the state of the k_{epf}-deep chain is determined. Since equivocations for the last k_{epf} blocks can only remove transactions, the node knows all transactions that *may* be included in the final chain. From this, the node can determine all states $\mathcal{S}$ that could be touched by any transaction in the last k_{epf} blocks.[5] A transaction tx that does not depend on any state in $\mathcal{S}$ for its execution, can thus be executed on the state of the k_{epf}-deep chain, therefore, satisfying predictable transaction validity. A node then only includes transactions that don't rely on a state in $\mathcal{S}$. Note that transactions in the same block could depend on the same state. □

Proof of Lem. 5. Consider a funding gas account acc with balance b before the last k_{epf} blocks in the chain. This balance is set for that account as no equivocation proofs against blocks that are k_{epf}-deep are allowed by the protocol. Note that any transactions in the last k_{epf} blocks that fund the account can still be sanitized from the ledger so we do not consider them in the balance yet. The node instead considers all transactions $\mathcal{T}_{\text{acc}(k_{\text{epf}})}$ in the last k_{epf} which use the funds from the account (including any withdrawals). Since the transactions funded by acc that end up in the ledger are a subset of $\mathcal{T}_{\text{acc}(k_{\text{epf}})}$, and all fees are extracted regardless of how a transaction executes, the node will at worst underestimate the balance of acc at the tip of the chain. □

G ATTACK ON POS LC WITHOUT EQUIVOCATION REMOVAL

In this section, we present an attack to establish a bound (as a function of the security level) on the block production rate (and hence, throughput, or bandwidth requirement) of a single chain PoS LC protocol without an equivocation removal policy. For concreteness, we demonstrate this attack on PoS LC using any one of three download rules: 'download the longest header chain', 'download towards the freshest block', and 'equivocation avoidance'. For the 'download the longest header chain' rule, [34, Figure 3] showed one attack and the attack in this section generalizes that. On the other hand, [34] proved PoS LC secure under the other two download rules by setting the duration of a slot proportional to the security parameter κ, to achieve security with probability $1 - \text{negl}(\kappa)$. Hence the block production rate (and throughput) decays as $O\left(\frac{1}{\kappa}\right)$. In this section, we show an attack which succeeds if the block production rate is $\Omega\left(\frac{1}{\log(\kappa)}\right)$.

Furthermore, while the attack in [34, Figure 3] required that the PoS LC protocol rejects blocks with invalid transactions after downloading them, this attack does not require that. Therefore, this attack works even if the PoS LC protocol accepts blocks with invalid transactions into the output ledger (*e.g.*, to subsequently clean them up deterministically across honest nodes). This is because as noted in Sec. 7, even if the protocol accepts blocks with invalid transactions, honest nodes must download the block content (to ensure data availability). This is why we require an equivocation removal policy so that honest nodes can unilaterally discard content for blocks that they do not download. This is what allows us to overcome the $\Omega\left(\frac{1}{\log(\kappa)}\right)$ throughput bound in this work.

Before describing the attack, we briefly recap the download rules analyzed in this section. In the 'download towards the freshest block' rule (cf. [34, Alg. 2]), a node chooses the block produced in the most recent time slot ('freshest'), and if it not yet downloaded, downloads the first unknown block in the chain containing that block. One the node downloads the freshest block, it stops downloading any blocks until a block header from a more recent slot shows up. In the 'equivocation avoidance' rule ([34, Alg. 4]), the node first filters the tree of its headers by keeping only one leaf per BPO (ties broken by the adversary). From among the remaining headers, the node picks a block to download as per the 'download longest header chain' rule. The 'download longest header chain' rule is as described in Alg. 4.

G.1 Attack Strategy

The attack works in two phases. See Figure 13 for reference. C is the bandwidth constraint (in blocks per second), τ is the slot duration, and κ is the security parameter.

Setup phase. At time slot t_0, the adversary creates a chain C which forks off the honest chain C_0 by at least $L = \log(\kappa)$ blocks, and is at least as long as C_0. The prefix length L is chosen so that the setup succeeds with non-negligible probability. The adversary initially keeps C private.

Execution phase. The adversary creates different chains $C_1, C_2, \ldots$ which contain equivocations of the blocks in C, and pushes one chain to each honest node.

(1) Let $t_1 > t_0$ be the first time slot with a block production. For any block b_1 produced in slot t_1, if b_1 is produced by an honest node, then, the adversary breaks ties such that b_1 extends one of the equivocating chains C_i. If b_1 is produced by the adversary, the adversary produces b_1 at the tip of another chain made of equivocations of the blocks in C. Regardless, any block b_1 produced in t_1 extends a chain that forks off the downloaded longest chain by L new blocks that need to be downloaded, hence it will take a long time for an honest node to download up to the block b_1.
(2) The adversary repeats step 1 in all time slots $t_2, t_3, \ldots$ with a block production. Assuming there are many honest nodes, each block extends a different equivocating chain and is too long to catch up with. The adversary continues this until the following condition occurs.
(3) Let t^* be the first time slot since t_0 in which an honest block b^* is produced, such that there are no other blocks produced in slots $[t^*, t^* + L/(C\tau))$. This condition ensures that there is enough time for b^* to be downloaded by all honest nodes.
If the adversary had at least one block production opportunity $t' \in [t_0, t^* + L/(C\tau))$, then the adversary attaches a block b' produced in slot t' to the chain C. The adversary makes the following updates,
- $C_0 \leftarrow$ chain ending in b^*,
- $C \leftarrow$ chain ending in b',
- $t_0 \leftarrow t^*$,
- $L \leftarrow L + 1$,

[5] Note that this includes all states a transaction could have changed if it executed differently. This could be achieved by transactions needing to include an access list of all states they are allowed to change. One can imagine a DOS attack where a transaction's access list could prevent future transactions.

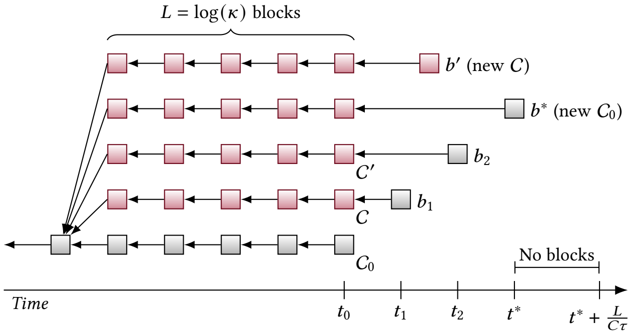


Figure 13: Illustration of the new attack of Section G.2. At time t_0, C_0 is the longest downloaded chain of all honest nodes, and the adversary produces a chain C that forks off C_0 by $L = \log(\kappa)$ blocks. Blocks produced in time slots $t_1, t_2, \ldots$ (whether honest or adversarial) extend the chain C or a chain C' containing equivocation of the blocks in C, and are not downloaded by all honest nodes in time before the next block production opportunity. Time slot t^* is the first slot such that there are no block productions in the $\frac{L}{C\tau}$ slots after t^*. The block b^* produced in slot t^* therefore gets downloaded. If the adversary had at least one block production opportunity $t' \in [t_0, t^* + L/(C\tau)]$, then the adversary sets the chain ending in b^* as new C_0 and the chain ending in b' as new C, and repeats the attack.

and thereafter repeats steps (1)–(3).

If the adversary failed to get one block production opportunity in $[t_0, t^* + L/(C\tau))$, then the adversary gives up.

G.2 Analysis Overview

The analysis below reuses notation defined in Section 4.2. For the attack to succeed, we assume the following:

- The protocol parameters ρ, τ satisfy $\frac{\rho}{\tau} > \frac{C}{\log \kappa} \log \frac{1-\beta}{\beta}$, where β is the fraction of adversarial nodes and C is the bandwidth constraint of each honest node in blocks per second.
- The total number of nodes N is large.
- The adversary is allowed to break ties among equally long chains in the fork choice rule.
- The adversary is allowed to break ties among equal priority chains in the download rule.

The fork length $L = \log(\kappa)$ is chosen such that the adversary can succeed in the setup phase with probability at least $e^{-O(L)} = 1/\text{poly}(\kappa)$ at any given time, even with a minority stake. Hence this setup can be achieved by the adversary with non-negligible probability eventually during an execution of length $\text{poly}(\kappa)$.

Now consider the execution phase. The key vulnerability exploited in this attack is that if the highest priority chain according to the download rule is on a long fork of which honest nodes have not downloaded any blocks, it will take a long time for honest nodes to download up to the tip of this chain. If the next block arrival happens before this chain is downloaded, the adversary makes honest nodes shift their download priority to a different chain, which is also on an equally long fork. This keeps repeating and honest nodes never finish downloading a chain that would help grow their longest downloaded chain.

Honest nodes get some respite when there is an honest block produced in slot t^* such that there are no other blocks produced in slots $[t^*, t^* + L/(C\tau))$. The three download rules 'download longest header chain', 'download towards the freshest block', and 'equivocation avoidance' ensure that the honest block b^* produced in slot t^* remains the highest priority chain to download during the slots $[t^*, t^* + L/(C\tau))$. Given a bandwidth constraint of C blocks per second, *i.e.*, $C\tau$ blocks per time slot, honest nodes can completely download a fork of length L in $L/(C\tau)$ time slots.

However, this does not end the attack. While waiting for one honest block production opportunity with $L/(C\tau)$ empty slots following it, if the adversary gets one block production opportunity, this allows the adversary to create a new chain whose length matches the longest downloaded chain of honest nodes. The situation now looks just like at the start of the execution phase, except that the adversary's chain now forks from the honest nodes' new downloaded longest chain by $L + 1$ blocks (one more than before). The adversary then and repeats the execution phase all over again with the new chain it has produced, and with $L \longleftarrow L + 1$. As the adversary's fork length L increase, it takes more time for honest nodes to download up to the tip of a newly produced block extending that fork. This means it takes even longer for honest nodes to produce a block after which there are $L/(C\tau)$ empty slots such that the block gets downloaded. Thereby, it becomes more likely that the adversary produces one block before honest downloads download a new chain, and continue the attack for another iteration with a larger fork length L. As a result of this vicious cycle, the adversary can continue this attack forever with non-negligible probability!

This attack breaks safety of the protocol because the downloaded longest chain of honest nodes switches to a different chain every time the condition in Section G.1 (3) occurs.

G.3 Analysis Details

Building up on the definitions from Section 4.2, define a time slot t to be *honest* if $H_t > 0$, and *attacking* if $A_t > 0$. Also define $\mathcal{H}_{(r,s]}$ and $\mathcal{A}_{(r,s]}$ as the number of honest and attacking slots respectively in the interval $(r, s]$. $\mathcal{B}_{(r,s]}$ is the number of slots $t \in (r, s]$ such that $H_t + A_t > 0$.

Definition 6. For all t, define the event

$$F_t := \left\{ \exists r < t \colon (H_r > 0) \wedge (\mathcal{A}_{(r,t]} \geq \mathcal{H}_{(r,t]}) \wedge (\mathcal{A}_{(r,t]} \geq L) \right\}. \quad (144)$$

Lemma 9. *If F_t occurs, then the setup phase of the attack succeeds at time slot t,* i.e., *there exists an adversarial strategy which creates a chain C that forks off the longest downloaded chain of all honest nodes at time t by L blocks and is at least as long as the longest downloaded chain.*

Proof. Let b be an honest block produced in slot $r < t$ where r satisfies $(H_r > 0) \wedge (\mathcal{A}_{(r,t]} \geq \mathcal{H}_{(r,t]}) \wedge (\mathcal{A}_{(r,t]} \geq L)$. The adversary's strategy is as follows. In time slot r, the adversary pushes the block b to all nodes irrespective of bandwidth, so that $\left|\mathrm{d}\mathcal{C}_p(r)\right| = |b|$ for all honest nodes i. The adversary then creates a private chain using its own blocks, extending the block b (all these blocks are kept hidden). The adversary can add one block to this chain in every slot in which the adversary produces a block, therefore the length of the adversary's chain at time t is $|b| + \mathcal{A}_{(r,t]}$. On the other hand, in every time slot that an honest block is produced, at most one block is added to the longest chain of all honest nodes, therefore the length of the honest chain at time t is at most $|b| + \mathcal{H}_{(r,t]}$. Since $\mathcal{A}_{(r,t]} \geq \mathcal{H}_{(r,t]}$, the adversary's chain has the same or greater length compared to the honest chain at time slot t. Since the last block that is common between the honest and adversary's chain is b, and $\mathcal{A}_{(r,t]} \geq L$, the adversary's chain forks off the honest chain by at least L blocks. Therefore, we have the required conditions for the attack setup. Note that the adversary does not need to be able to predict when the event F_t would occur. Since creating blocks in proof-of-stake does not require computation time, the adversary can create this chain after it observes that the event F_t occurred. □

Lemma 10. *Let $t > t_0$ be a successful time slot (*i.e., $H_t + A_t > 0$*) such that there exists another successful time slot $t' \in (t, t + L/(C\tau)]$. Then none of the blocks produced in slot t are ever downloaded by any honest node. Hence for all honest nodes p, $L_p(t' - 1) = L_p(t)$.*

Proof. For all blocks b that are produced in slot t, the attack strategy in Section G.1 ensures that the number of blocks to be downloaded in the prefix of b (including b) is $L+1$. Since each honest node can download at most $C\tau$ blocks per time slots, no honest node can download the entire prefix within $L/(C\tau)$ time slots (the adversary does not push any blocks to honest nodes during this period). At time slot t', either an adversarial block or an honest block (or both) are produced. In either case, step 2 of the execution phase ensures that at slot t', this new block has the highest priority under all three download rules. This is because i) it is clearly the freshest block at slot t', ii) it is one of the longest chains (and the adversary breaks ties), and iii) it has a non-equivocating tip and has length $L + 1$, which is one of the longest chains (and the adversary breaks ties), Therefore, at time slot t', all honest nodes switch to download a different block, and therefore the block b is not downloaded. Since for all honest nodes p, no block is downloaded, it is clear that $L_p(t' - 1) = L_p(t)$. □

Lemma 11. *Let t be a successful time slot such that for all time slots $t' \in (t, t + L/(C\tau)]$, there are no blocks produced in slot t' (*i.e., $H_{t'} + A_{t'} = 0$*). Then, each honest node downloads at least one block produced in slot t, and for all honest nodes p, $L_p(t + L/(C\tau)) = L_p(t) + 1$.*

Proof. Since t is an honest time slot, let b be one of the honestly produced blocks in this time slot. At time slot t, b is one of the freshest blocks. It remains one of the freshest blocks until time slot $t + L/(C\tau)$ because there are no other blocks produced in this interval. As per the attack strategy, for both honest and adversarial blocks b, the block b is on the longest chain in every node's view, and is not an equivocation.

In case of a tie in the download rules, we assume that all honest nodes break the tie in favour of the same block b (as this is chosen by the adversary). Therefore, the block b has the highest download priority for all honest nodes in slots $[t, t + L/(C\tau)]$. Since the number of blocks to be downloaded in the prefix of b (including b) is $L + 1$, these blocks can be downloaded before the end of slot $t + L/(C\tau)$. We know that b is longer than all honest nodes' longest downloaded chains at slot t because of the attack strategy and Lemma 10. Therefore the length of the longest downloaded chain of every honest node grows by 1. □

Lemma 12. *Let t_a be the first time slot such that $t_a > t_0 + T_{\text{conf}}$, t_a is a successful time slot, and there are no blocks produced in slots $(t_a, t_a + L'/(C\tau)]$ where L' is the value of the attacker's parameter L at time slot $t_0 + T_{\text{conf}}$. If the attacker does not terminate before slot t_a, then there is a safety violation.*

Proof. At the end of time slot $t_0 + T_{\text{conf}}$, let $\mathsf{LOG}_p^{t_0+T_{\text{conf}}}$ denote the ledger output by an honest node p. Note that this ledger contains all blocks mined before slot t_0 in the longest downloaded chain of node p, $\mathrm{d}\mathcal{C}_p(t_0 + T_{\text{conf}})$. As per the attack strategy Section G.1 steps (1) and (2), the block produced in time slot t_a extends a different equivocating chain that forks off $\mathrm{d}\mathcal{C}_p(t_0 + T_{\text{conf}})$ by L' blocks. Since there are no blocks produced in slots $(t_a, t_a + L'/(C\tau)]$, all honest nodes download this new chain and hence update their longest downloaded chain. However, note that at least L' blocks that were in $\mathsf{LOG}_p^{t_0+T_{\text{conf}}}$ are replaced by different blocks in $\mathrm{d}\mathcal{C}_p(t_a)$, and therefore $\mathsf{LOG}_p^{t_0+T_{\text{conf}}}$ and $\mathsf{LOG}_p^{t_a}$ are not prefixes of each other. This causes a safety violation. □

Lemma 13. *For all t,*

$$\Pr[F_t] \geq p_{\text{H}} \left(1 - 2e^{-L/9}\right) \frac{1}{\sqrt{8L}} e^{-4\alpha_4 L}, \tag{145}$$

where $\alpha_4 = \frac{1}{4}\ln\left(\frac{p}{4p_{\text{H}}}\right) + \frac{3}{4}\ln\left(\frac{3p}{4p_{\text{H}}}\right)$ and $p_{\text{H}} \triangleq \Pr[H_t > 0] = 1 - e^{-(1-\beta)\rho}$.

Proof. Let $T = \frac{2L}{p}(1 + \epsilon)$ for some $\epsilon > 0$ and let $s = t - T$.

$$\begin{aligned}
&\Pr[F_t] \\
&= \Pr\left[\exists r < t\colon (H_r > 0) \wedge (\mathcal{A}_{(r,t]} \geq \mathcal{H}_{(r,t]}) \wedge (\mathcal{A}_{(r,t]} \geq L)\right] \\
&\geq \Pr\left[H_s > 0 \wedge \mathcal{A}_{(s,t]} \geq \mathcal{H}_{(s,t]} \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&= \Pr[H_s > 0] \Pr\left[\mathcal{A}_{(s,t]} \geq \mathcal{H}_{(s,t]} \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&\geq \Pr[H_s > 0] \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&\overset{(a)}{\geq} \Pr[H_s > 0] \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge \mathcal{B}_{(s,t]} \geq 2L\right] \\
&\geq p_{\text{H}} \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge 2L \leq \mathcal{B}_{(s,t]} \leq 2L(1 + 2\epsilon)\right] \\
&\geq p_{\text{H}} \Pr\left[2L \leq \mathcal{B}_{(s,t]} \leq 2L(1 + 2\epsilon)\right] \\
&\qquad \Pr\left[\mathcal{H}_{(s,t]} \leq L \mid \mathcal{B}_{(s,t]} = 2L(1 + 2\epsilon)\right]
\end{aligned} \tag{146}$$

where (a) is because $\mathcal{H}_{(s,t]} + \mathcal{A}_{(s,t]} \geq \mathcal{B}_{(s,t]}$. By Chernoff bounds for $\delta \in (0, 1)$,

$$\Pr\left[\mathcal{B}_{(s,t]} < p(t - s)(1 - \delta)\right] \leq \exp\left(-\frac{p(t - s)\delta^2}{2}\right), \tag{147}$$

$$\Pr\left[\mathcal{B}_{(s,t]} > p(t - s)(1 + \delta)\right] \leq \exp\left(-\frac{p(t - s)\delta^2}{3}\right). \tag{148}$$

where $p \triangleq \Pr[H_t + A_t > 0] = 1 - e^{-\rho}$. With $t - s = \frac{2L}{p}(1 + \epsilon)$ and $\delta = \frac{\epsilon}{1+\epsilon}$,

$$\Pr\left[\mathcal{B}_{(s,t]} < 2L\right] \leq \exp\left(-\frac{2L\epsilon^2}{2(1 + \epsilon)}\right),$$

$$\Pr\left[\mathcal{B}_{(s,t]} > 2L(1 + 2\epsilon)\right] \leq \exp\left(-\frac{2L\epsilon^2}{3(1 + \epsilon)}\right)$$

$$\implies \Pr\left[2L \leq \mathcal{B}_{(s,t]} \leq 2L(1+2\epsilon)\right] \geq 1 - 2\exp\left(-\frac{2L\epsilon^2}{3(1+\epsilon)}\right). \quad (149)$$

Each non-empty time slot ($H_t + A_t > 0$) is an honest slot ($H_t > 0$) independently with probability $\frac{p_{\mathrm{H}}}{p}$. Therefore conditional on $\mathcal{B}_{(s,t]} = 2L(1+2\epsilon)$, $\mathcal{H}_{(s,t]}$ has a binomial distribution. Then we can use tail bounds for the binomial distribution to show that

$$\begin{aligned} &\Pr\left[\mathcal{H}_{(s,t]} \leq L \mid \mathcal{B}_{(s,t]} = 2L(1+\epsilon)\right] \\ &\geq \frac{1}{\sqrt{4L(1+2\epsilon)}} \exp\left(-2\alpha_4 L(1+2\epsilon)\right) \end{aligned} \quad (150)$$

where $\alpha_4 = D\left(\frac{1}{2(1+2\epsilon)} || \frac{p_{\mathrm{H}}}{p}\right)$ and

$$D(x||y) = x \ln\left(\frac{x}{y}\right) + (1-x)\ln\left(\frac{1-x}{1-y}\right). \quad (151)$$

Putting these together,

$$\Pr\left[F_t\right] \geq p_{\mathrm{H}}\left(1 - 2e^{-2L\epsilon^2/3(1+\epsilon)}\right)\frac{1}{\sqrt{4L(1+2\epsilon)}} e^{-2\alpha_4 L(1+2\epsilon)}. \quad (152)$$

Since ϵ is arbitrary, we may choose $\epsilon = \frac{1}{2}$ to get a lower bound on the required probability. □

Corollary 3. *For large κ, if $L = \Theta(\log\kappa)$ and $\rho = \Omega\left(\frac{1}{n}\right)$, then* $\Pr\left[F_t\right] \geq \frac{1}{\mathrm{poly}(\kappa)}$.

Recall that the attack goes on forever if the attacker gets one block production opportunity before the honest nodes download a longer chain. We have seen that honest nodes download a longer chain if and only if a non-empty slot is followed by at least L/C empty time slots.

Definition 7. A successful time slot t is called a T-loner if no blocks are produced in the T slots following t, *i.e.*, $\mathcal{B}_{(t+1,t+T]} = 0$. The predicate $\mathrm{Loner}_T(t)$ is true iff slot t is a T-loner.

We observe that

$$\Pr\left[\mathrm{Loner}_T(t) \mid H_t + A_t > 0\right] = (1-p)^T. \quad (153)$$

Lemma 14. *If $(1-\beta)e^{-\rho T} < \beta$, then the probability that the adversary gets one block production opportunity before a T-loner occurs is at least $1 - \frac{(1-\beta)e^{-\rho T}}{\beta} > 0$.*

Proof. We begin by calculating the probability there is at least one attacking slot before there is an T-loner. This ensures that the final step of the attack in Section G.1 is successful and that the adversary can updates it state and continue the attack. Let $t_1, t_2, \ldots$ be the sequence of successful slots since the start of the attack. Let t_N be the first T-loner in this sequence (note that N is a random variable).

$$\begin{aligned} &\Pr\left[\exists i \leq N \colon A_{t_i} > 0\right] \\ &= \sum_{k=1}^{\infty} \Pr\left[N = k\right] \Pr\left[\exists i \leq k \colon A_{t_i} > 0 \mid N = k\right]. \end{aligned} \quad (154)$$

Here,

$$\begin{aligned} \Pr\left[N = k\right] &= \prod_{i=1}^{k-1} \Pr\left[\neg\mathrm{Loner}_T(t_i) \mid H_{t_i} + A_{t_i} > 0\right] \\ &\quad \Pr\left[\neg\mathrm{Loner}_T(t_k) \mid H_{t_k} + A_{t_k} > 0\right] \\ &= \left(1 - (1-p)^T\right)^{k-1} (1-p)^T. \end{aligned} \quad (155)$$

Moreover, conditioned on t being a successful slot, the events $\mathrm{Loner}_T(t)$ and $A_t > 0$ are independent. Therefore,

$$\begin{aligned} &\Pr\left[\exists i \leq k \colon A_{t_i} > 0 \mid N = k\right] \\ &= 1 - \prod_{i=1}^{k} \Pr\left[A_{t_i} = 0 \mid A_{t_i} + H_{t_i} > 0\right] \\ &= 1 - \left(\frac{e^{-\beta\rho}(1 - e^{-(1-\beta)\rho})}{(1 - e^{-\rho})}\right)^k \\ &= 1 - \left(1 - \frac{1 - e^{-\beta\rho}}{1 - e^{-\rho}}\right)^k \\ &\geq 1 - (1-\beta)^k \end{aligned} \quad (156)$$

Putting them together,

$$\begin{aligned} &\Pr\left[\exists i \leq N \colon A_{t_i} > 0\right] & (157) \\ &\geq \sum_{k=1}^{\infty} \left(1 - (1-p)^T\right)^{k-1} (1-p)^T \left(1 - (1-\beta)^k\right) \\ &= 1 - \frac{(1-p)^T(1-\beta)}{1 - (1 - (1-p)^T)(1-\beta)} \\ &\geq 1 - \frac{(1-p)^T(1-\beta)}{\beta}. & (158) \end{aligned}$$

Finally, we substitute $p = 1 - e^{-\rho T}$. □

Lemma 15. *If the protocol parameters ρ, τ satisfy $\frac{\rho}{\tau} > \frac{C}{L}\log\frac{1-\beta}{\beta}$, then with probability non-negligible in κ, the attack never terminates.*

Proof. From Corollary 3, for $L = \log(\kappa)$ and large enough ρ, the attack setup occurs with non-negligible probability.

If the adversary gets one block production opportunity before an $L/(C\tau)$-loner, then the adversary can continue the attack by upgrading L to $L+1$. This means that in the next iteration of the attack, the adversary needs one block production opportunity before an $(L+1)/(C\tau)$-loner. Since an $(L+1)/(C\tau)$-loner is rarer than an $L/(C\tau)$-loner, the adversary has increased chances of getting one block production before an $(L+1)/(C\tau)$-loner, and therefore upgrading the attack to $L+2$. This process repeats whereby if the adversary upgrades the attack to the next phase, it increases the chance that the attacker can further upgrade the attack to the next phase, and so forth.

The probability that the attack continues forever is therefore

$$\begin{aligned} \Pr\left[\text{attack continues forever}\right] &\geq \prod_{l=L}^{\infty}\left(1 - \frac{(1-\beta)e^{-\frac{\rho l}{C\tau}}}{\beta}\right) \\ &\geq \prod_{l=L}^{\infty}\left(1 - e^{-\frac{\rho(l-L)}{C\tau}}\right) \\ &= \prod_{l=1}^{\infty}\left(1 - e^{-\frac{\rho l}{C\tau}}\right) \\ &= \left(e^{-\frac{\rho}{C\tau}}; e^{-\frac{\rho}{C\tau}}\right)_{\infty}. \end{aligned} \quad (159)$$

Here, $(x;x)_\infty$ is called the q-Pochhammer symbol and $(x;x)_\infty \in (0,1)$ for all $x \in (0,1)$ [45]. The condition $\frac{\rho}{\tau} > \frac{C}{L}\log\frac{1-\beta}{\beta}$ is derived from the condition in Lemma 14 with $T = \frac{L}{C\tau}$. □

Corollary 4. *For the protocol $\Pi^{\rho,\tau,k_{\text{conf}}}$ to satisfy safety and liveness, the throughput of the protocol must be $O\left(\frac{1}{\log \kappa}\right)$.*

This is seen by noting that the throughput is $\frac{1-e^{-\rho}}{\tau} \leq \frac{\rho}{\tau} \leq \frac{C}{\log \kappa} \log \frac{1-\beta}{\beta}$. If this is not true, then the attacker never terminates, hence there is a safety violation as per Lemma 12. The maximum block production rate $\lambda = \frac{\rho}{\tau}$ calculated from Lem. 15 is plotted in Fig. 1(b).