Security of Nakamoto Consensus under Congestion

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*—Nakamoto consensus (NC) powers major proof-of-work (PoW) and proof-of-stake (PoS) blockchains such as Bitcoin or Cardano. Given a network of nodes with certain communication and computation capacities, against what fraction of adversarial power (the resilience) is Nakamoto consensus secure for a given block production rate? Prior security analyses of NC used a *bounded delay* model which does not capture network congestion resulting from high block production rates, bursty release of adversarial blocks, and in PoS, spamming due to equivocations. For PoW, we find a new attack, called *teasing attack*, that exploits congestion to increase the time taken to download and verify blocks, thereby succeeding at lower adversarial power than the *private attack* which was deemed to be the worst-case attack in prior analysis. By adopting a *bounded bandwidth* model to capture congestion, and through an improved analysis method, we identify the resilience of PoW NC for a given block production rate. In PoS, we augment our attack with equivocations to further increase congestion, making the vanilla PoS NC protocol insecure against *any* adversarial power except at very low block production rates. To counter equivocation spamming in PoS, we present a new NC-style protocol *Sanitizing PoS* (SaPoS) which achieves the same resilience as PoW NC.**

I. 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* of individual nodes and the security of the system.

In order to remain secure under adversaries controlling up to 50% of the network, blockchain protocols have been parameterized to leave a 'security margin' between the block production rate under normal operation, and each node's capacity limits. For instance, Bitcoin only produces one block of transactions per ten minutes, though it usually only takes a few seconds for a node to download and process each block [1]. On the other hand, protocols that push close to the limits of their nodes become insecure as the processing capacities of nodes are overwhelmed (such as Solana [2]–[4]). The natural question that arises is: given a capacity limit of nodes, what is the trade-off between the block production rate and the fraction of adversarial power that the protocol tolerates?

In this work, we focus on Nakamoto consensus (NC) protocols (a.k.a. longest chain (LC) consensus [5])—a popular class of blockchain protocols that can be instantiated using various Sybil resistance mechanisms such as proof-of-work (PoW) [5], [6] and proof-of-stake (PoS) [7]–[10]. In NC protocols, a continuously running lottery selects which nodes can produce

LK, JN, SS and AZ are listed alphabetically.

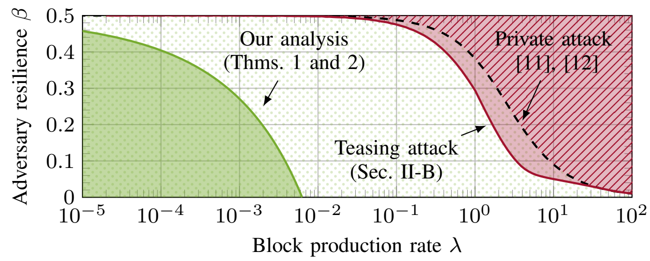

Fig. 1. Regions of fraction β of adversarial nodes and block production rate λ with security proofs (green) and attacks (red) for Nakamoto consensus under a fixed processing capacity of $C = 1$ block per second. Analysis in the bounded delay model [11], [12] (with $\Delta = 1$ s) proves that the private attack (- - -) succeeds (hatched) iff $\beta \geq \frac{1-\beta}{1+(1-\beta)\lambda}$, and that for all other values of β, λ, no attack succeeds (dotted). The latter claim is limited to the bounded delay model, as our teasing attack exploits congested block processing and succeeds at lower adversary β than the private attack (red, hatched). Our analysis in a bounded bandwidth model characterizes a security region (green) for both PoW and SaPoS (a PoS variant).

the next block and when they can do so. A selected node collects pending transactions, creates a new block extending the longest chain of blocks it sees, pushes the block's *header* to the network, and makes its transaction *content* available for download.

In order to extend a chain, nodes must first *process*, *i.e.*, download and verify, the content of blocks in that chain, to ensure that the content is available and valid. The lottery makes block production occur at random times, which means the download and verification load of the network is bursty. With limited processing capacity, during times of high load, new blocks will be *queued* for processing. This leads to a queuing delay, in addition to the processing delay of blocks. Nodes cannot produce ('*mine*') new blocks extending blocks that they have not yet processed, so the growth of the honest nodes' chain slows down. Since the security of NC is based on the honest chain outgrowing any adversarial chain, the reduced growth of the honest nodes' chain makes it easier for an adversary to attack the system.

Moreover, the adversary can selectively delay the release of blocks that it produces, so the network load can be adversarially controlled to some extent. These blocks may not extend the longest chain or may otherwise be invalid. Thus, to ensure that nodes can process the 'most important' blocks first, they must use a 'scheduling policy': given a set of new block headers, which blocks to download and verify first. Since a node extends their longest chain to produce new blocks, the most obvious policy is to first process blocks along the longest chain that the node has seen. Indeed, this policy can be found in major implementations of Nakamoto consensus (Bitcoin [13], Cardano [14]).

NC security depends critically on the *delay* between an honest node producing a new block and any other honest node being able to mine on top of it. How should this delay be modeled? To analyze the security of Nakamoto consensus, previous work [6], [9], [11], [12], [15]–[18] has considered the 'bounded delay' model. This model assumes that all honest nodes are able to mine on a block after a maximum delay of Δ seconds after the block is published. Using this model, the works [11], [12] give a tight characterization of the trade-off between the fraction β of adversarial nodes and the block production rate λ for a given delay bound Δ. However, the bounded delay model assumes that the processing time of each block is bounded by Δ, *irrespective of the total processing load*. Thus, the model fails to capture the earlier discussed effects of queuing.

The bounded delay analysis [11], [12] predicts that the *private attack* is the worst-case attack strategy. That is, under parameters where the private attack fails, all other attacks fail, too. In the private attack, the adversary produces a chain of blocks that it keeps private until it becomes longer than the public longest chain, which it can then displace. Note that during this attack, the adversary does not release any blocks. In this case, the scheduling policy ensures that whenever an honest block is proposed at a new height, each honest node first processes that block and thereafter can produce a new block extending it. Therefore, under the private attack, the honest nodes' chain grows at the same rate as under the bounded delay model with Δ taken to be the time to process one block (this is further explained and experimentally validated in Sec. II). Hence, the bounded delay analysis can be used to calculate the fraction of adversarial power with which the private attack succeeds, shown in Fig. 1. We ask whether there are stronger attacks in which the adversary adds to the network load to increase queuing delays, an effect that was not captured by the bounded delay model.

Result 1. *We show a* teasing attack *(Sec. II-B) which is stronger than the private attack,* i.e.*, it succeeds in regions of* (λ, β) *where the private attack does not succeed (Fig. 1).*

In this attack, the adversary 'teases' honest nodes to process a longer chain it announces, but makes this effort 'useless' by not releasing the block contents for the entire chain. The adversary effectively doubles the network load and queuing delays while also building a longer chain to break security. This halves the maximum secure block rate λ.

While *insecurity* can be shown through attacks in experiments, *security* (*i.e.*, non-existence of any attacks) cannot be shown in experiments, but requires mathematical proof.

To study queuing effects on blockchain security, we adopt the 'bounded bandwidth' model proposed in [19]. As per this model, we consider the scheduling policy as a part of the protocol description as it affects the security of the protocol. *Henceforth, though we adopt the word 'bandwidth', we continue to mean 'capacity' in the wider sense,* i.e.*, rate limits on communication, computation, and storage. Similarly, we use 'download' to mean 'process' in the wider sense.*

Result 2. *Using the bounded bandwidth model, we characterize a region of block mining rate* λ *and adversarial fraction* β *for which we prove that proof-of-work Nakamoto consensus, with a wide range of suitable scheduling policies, is secure (Thm. 1). This region is shown in Fig. 1.*

In proof-of-stake, the block production lottery does not depend on the block content or the parent block, to avoid grinding attacks [20]. However, this allows the adversary to produce equivocations—multiple blocks with the same lottery but different content and/or parents—and send them to different nodes. Analysis in the bounded delay model predicted that this new attack vector does not change the security region [11], [21]. However, since nodes cannot always predict which of two conflicting blocks will eventually be part of the chain, they may waste processing capacity on blocks that are later discarded. Thus, the attacker has infinitely many blocks which it can use to increase the network load and queuing delays (more than in PoW), as observed in [19].

Result 3. *We show an* equiv-teasing attack *(Sec. II-C) which extends the teasing attack using equivocations. This attack succeeds with probability* ϵ *if* $\lambda = \Omega\left(\frac{1}{\log(1/\epsilon)}\right)$.

Therefore, in PoS, as the security error probability goes to zero, the throughput goes to zero as well. This was not the case in PoW because the adversary had a budget constraint: blocks spent to drive up congestion for an attack today cannot be spent tomorrow, and vice versa. To re-introduce the budget constraint into PoS NC, we propose *equivocation removal*: nodes download at most one of possibly many equivocating blocks (Sec. VI). Additionally, nodes use *equivocation proofs* to collectively remove the content of equivocating blocks from the ledger, so that no node needs to download the equivocating blocks to form the ledger. A deadline for considering equivocation proofs ensures that content is not removed from the ledger after it is confirmed. We present the protocol *SaPoS* (*Sanitizing Proof-of-Stake*) with these modifications.

Result 4. *SaPoS is secure for the same block production rates and adversarial fractions as PoW NC (Thm. 2).*

Based on Thms. 1 and 2, we calculate the bandwidth sufficient to secure PoW NC and SaPoS with the parameters of major PoW/PoS blockchain implementations (Fig. 3).

Equivocation removal in SaPoS 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 that are later invalidated. Moreover, the funding source of such transactions may also be invalidated, so no fees can be claimed for the resources these transactions occupy.

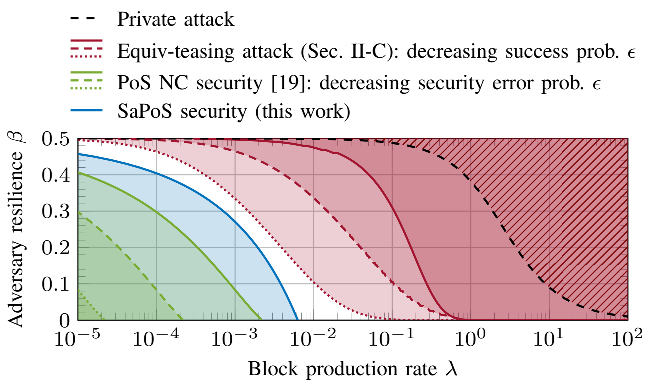

Fig. 2. Regions of fraction β of adversarial nodes and block production rate λ with security proofs [19] (green box) and attacks (red box) for PoS Nakamoto consensus under a fixed processing capacity of $C = 1$ block per second. The private attack succeeds with overwhelming probability for parameters in the hatched region while it fails with overwhelming probability otherwise. In contrary, our equiv-teasing attack continues to succeed at a lower adversarial fraction as the required attack success probability is decreased (in order, solid, dashed, dotted), as analyzed in App. H-B. This explains why the security region (against all attacks) proven in [19] shrinks as the desired security error probability decreases (in order, solid, dashed, dotted). Our new protocol SaPoS, by using equivocation removal, achieves a security region (blue box) that is independent of the security error probability.

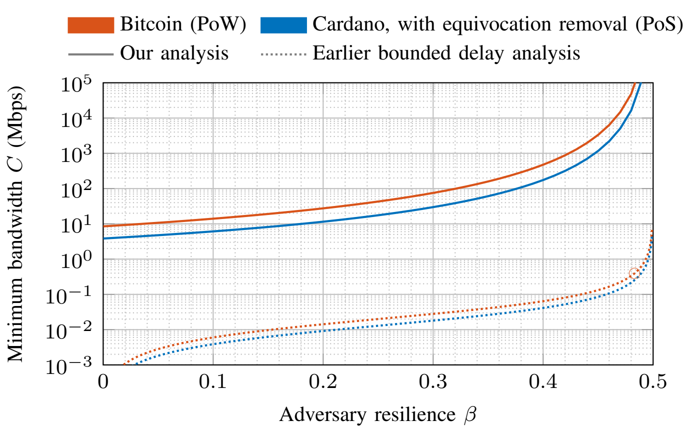

Fig. 3. Calculation based on Thms. 1 and 2 of the bandwidth per node that is sufficient to ensure security of NC with the parametrizations used by two major blockchains: Bitcoin (PoW, $\lambda = 1/600$ blocks/s, max. block size 4 MB, orange line), and Cardano (PoS, $\lambda = 1/20$ blocks/s, max. block size 88 KB, blue line). Dotted lines show the corresponding predictions from the bounded delay analysis, which holds for the private attack only (orange dotted, blue dotted). This suggests that while the commonly recommended 0.4 Mbps [22] is enough to secure Bitcoin against a 48% private attack adversary (○), higher bandwidth may be required to defend against all attacks.

Result 5. *We present mechanisms (Sec. VII) to ensure that miners receive fees for every transaction they include, and thus ensure that spamming the chain with invalidated transactions comes at a cost to the adversary.*

A. Related Works

Several earlier works have analyzed the security of PoW [5], [6], [11], [12], [16]–[18] and PoS [7]–[11], [15], [20] Nakamoto consensus in the bounded delay model. Our analysis builds on tools from several of these works, primarily pivots [9] (or Nakamoto blocks [11]), and convergence opportunities [9], [17], [18] (or similar [11], [16]).

Limitations of the bounded delay model have been observed in previous work. To use the bounded delay model to set the protocol's block production rate, one needs to find the value of the bound Δ. This is tricky 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. We also see that parameterizing using $\Delta = 1/C$ suffices for defending against the private attack. For security against other attacks that may manipulate queuing delays, 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 [23]. In practice, the delay distribution can be estimated through network experiments [1], [24]. Another work [25] analyzes security in a random (iid) delay model.

The problem is that the network load, hence queuing delay, is not purely a random process, but is controlled by the adversary. Experiments cannot show us the security impact under all possible adversarial manipulations of queuing delays. In analytical work [19], the bounded bandwidth model captures this effect. Since the analysis in [19] focuses on proof-of-stake, it allows the adversary to use infinitely many equivocations of each block it produces. Therefore, *at every moment* the adversary has infinitely many blocks to add to the network load. As a result, the earlier analysis proves security only when $\lambda = O\left(\frac{1}{\log(1/\epsilon)}\right)$ where ϵ is the desired maximum security error probability. In fact, our equiv-teasing attack succeeds with probability ϵ when $\lambda = \Omega\left(\frac{1}{\log(1/\epsilon)}\right)$, showing that the dependence of λ on ϵ is not just a result of weak analysis. Fig. 2 shows the security results of [19] and the parameters under which the equiv-teasing attack succeeds, as the security error probability varies.

Applying the analysis of [19] to PoW is too pessimistic because the adversary can not repeatedly re-use its blocks at every moment. Our first improvement over [19] is a new analysis technique which accounts for the budget constraint in PoW to eliminate the dependence on ϵ. In PoS however, our equiv-teasing attack shows that the dependence of block rate on the security error probability requires not just tighter analysis, but a change to protocol and/or scheduling policy. Thus, our second improvement over [19] is to design SaPoS which overcomes the dependence of block rate on ϵ in PoS. It is also worth noting that while [19] shows that PoS NC is insecure with the 'longest header chain' download rule and secure with the 'freshest block' rule, we show that both PoW NC and SaPoS are secure with a wide range of download rules including the ones considered in [19]. Moreover, since our equiv-teasing attack works against all those download rules, it indicates that securing PoS NC requires a change such as

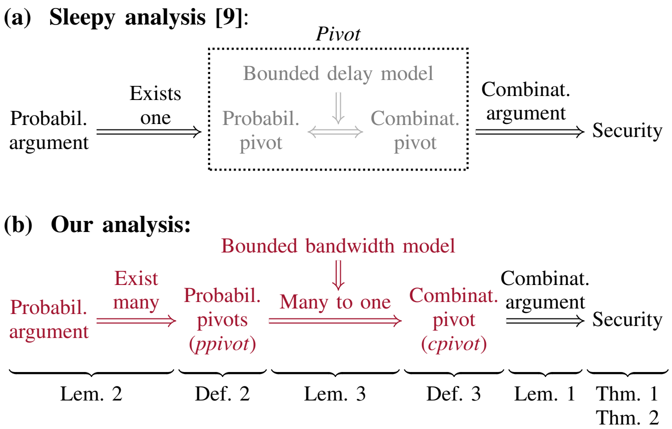

Fig. 4. (a) Sleepy analysis [9] is based on *pivots*. Pivots are special honest blocks (cf. liveness) which by a combinatorial argument remain in the chain forever (cf. safety), and by a probabilistic argument happen frequently. The equivalence of the qualities required for the probabilistic/combinatorial 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.

equivocation removal and not just a different download rule. See Fig. 2 to compare the security regions of the standard PoS NC protocol and our SaPoS protocol.

Capacity limits apply not only to downloads, but also to computational processing of blocks. For instance, to validate a block, an Ethereum validator must execute all smart contracts in it. While download and validation are similar in that the time taken increases with the number of transactions, they are different in that validation is hard to parallelize due to transactions that depend on each other. A line of work [26], [27] studies methods to parallelize execution of smart contracts to make use of multicore architectures.

B. Overview of Methods

1) New Analysis Technique: Traditional Nakamoto consensus security analysis (Fig. 4(a)) is based on the notion of a *pivot* [9] (or *Nakamoto block* [11]). 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 time interval around the pivot, there are more honest than adversarial block production opportunities. A probabilistic argument shows that typically pivots happen frequently. A combinatorial argument shows that the pivot block remains in the longest chains of all honest nodes forever. Safety and liveness of NC 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 (chain growth occurs only when an honest block is produced *and soon downloaded* 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 second technical contribution is a combinatorial argument to show that if there is a sufficiently high density of ppivots over a 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 [9] guarantees only a fairly low density of ppivots. Thus, our third 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 NC.

2) Equivocation Removal: The above analysis does not hold for the vanilla PoS NC protocols in [9], [10], [15], [19] because the analysis relies on the adversary having a limited budget of blocks that it can make honest nodes download. However, in PoS NC, the adversary has unlimited budget through equivocations. Indeed, our equiv-teasing attack in Sec. II-C exploits this to show that the vanilla PoS NC is not secure under the same parameters as PoW NC.

In SaPoS, we stipulate that per block production opportunity, every honest node downloads at most one block. This makes honest nodes immune to the effects of equivocation spamming. However, we need to ensure that honest nodes can still switch from one chain to another longer chain, both of which might contain different equivocating blocks from the same block production opportunity. For this, note that headers of two equivocating blocks from the same block production opportunity can serve as a *succinct equivocation proof* to convince other nodes that an equivocation was committed. Therefore, in SaPoS, if an honest node sees an equivocation for a block in its longest chain, it publishes an equivocation proof in the block that it produces, which allows all nodes to consistently treat the equivocating block's content as *empty* without downloading it.

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 NC 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.

3) Ensuring Fees Get Paid despite Lack of Predictable Validity: Equivocation removal in SaPoS leads to *lack of predictable transaction validity*, *i.e.*, honest nodes do not know whether transactions they include in their block will be valid, since the content of blocks in the prefix may later be removed due to an equivocation. This risks that the adversary gets to spam the ledger with invalid transactions for free. In one solution to prevent this, we focus on guaranteeing *transaction fees* are always paid regardless of equivocations, by introducing *gas deposit accounts* that can only be used to pay transaction fees. Any deposit to such an account takes effect only after 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 which they can use to reliably determine whether a transaction can pay fees.

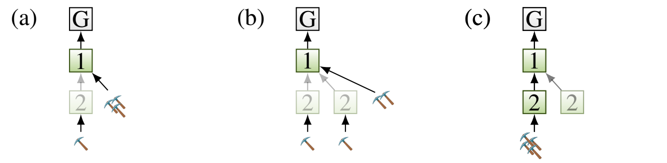

Fig. 5. Honest chain growth when the adversary does not release blocks (no attack or the private attack). Semi-transparent blocks have been announced (*i.e.*, headers released) but were not yet downloaded by honest nodes. (a) A new block is announced at height 2. This block is not yet processed by most of the nodes. As per the longest-header-chain policy, all honest nodes start processing this block. Until honest nodes process this block, they still attempt to mine a new block at height 2. (b) Some blocks may be mined at the same height 2, which do not lead to chain growth. (c) When honest nodes finish processing the first block at height 2 (which takes at most $1/C$ time since that block was announced), they now attempt to mine the next block extending it at height 3. Note: If the same node produces two consecutive blocks, then the second of the two blocks will be at a new height as this node has already processed the first. However, when the number of nodes is large, the probability that the same node produces two consecutive blocks is very small, and therefore can be neglected.

II. Attacks and Experiments

In this section, we describe the teasing attack and equiv-teasing attack. We simulate both attacks on a network of 100 nodes.[1] Setup details are in App. A. Honest nodes collectively produce blocks at a rate $\lambda_{\mathrm{hon}} = 1$ block per second. Each node has a limited processing rate of C blocks per second. Blocks consist of content (transactions) and a header (block production lottery information and parent block pointer). The header contains information to verify the block production lottery, thus nodes only process validly created blocks. Given a tree of valid block headers, nodes run the longest-header-chain policy: nodes attempt to process (download and verify) the first unprocessed block along the longest header chain. If the longest chain is already processed, or if the content of any block on that chain is unavailable or invalid, then the rule considers the next longest header chain, and so on.

A. Recap of the Private Attack

In the private attack, the adversary produces a chain of blocks that it keeps private until it becomes longer than the honest nodes' longest chain. Subsequently, releasing the chain causes honest nodes to switch their longest chain. Recall that in Nakamoto consensus, honest nodes confirm transactions which are at least k_{conf} blocks deep in their longest downloaded chain, for some parameter k_{conf} chosen by the node. Therefore, if the adversary's chain differs from the honest nodes' original longest chain by k_{conf} blocks, then the attack causes honest nodes to alter their ledger, which is a safety violation.

Note that during this attack, the adversary does not release any blocks. So, honest nodes undergo processing delay due to honest blocks only. The growth of the honest nodes' chain in this case is illustrated in Fig. 5. Due to processing delays, the chain growth rate $\lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$ is less than the honest mining rate λ_{hon} ('no attack' in Fig. 8). In this case, the honest chain growth rate is approximately the same as that under a network where all blocks are processed within $\Delta = 1/C$ time after they are announced (the bounded delay model), as seen in Fig. 8.

The success of the private attack (and the attacks we describe ahead) depend crucially on the race between the honest chain growth rate λ_{grwth} and the adversary's block production rate λ_{adv}. If $\lambda_{\mathrm{adv}} > \lambda_{\mathrm{grwth}}$, then with high probability, in the long run, the adversary's private chain is longer than the honest chain, so the attack succeeds. Thus, the honest chain growth rate determines the adversarial fraction β required for the attack.[2] Conversely, if $\lambda_{\mathrm{adv}} < \lambda_{\mathrm{grwth}}$, then with high probability, in the long run, the adversary's chain is shorter than the honest chain. In the short run however, even if $\lambda_{\mathrm{adv}} < \lambda_{\mathrm{grwth}}$, with some probability, the adversary's chain can get longer than the honest chain. Our attacks exploit this to save up blocks which will be released during the attack.

B. The Teasing Attack (PoW and PoS)

In Fig. 6, we describe the teasing attack. We see that the adversary utilizes the chain that it constructs not only to later overtake the public chain and break safety, 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.

Before the attack starts, the honest chain grows at the rate $\lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$ just as in the private attack, because the adversary does not release any blocks. To start the attack, the adversary mines a short private chain, which succeeds with some probability even if the adversary's mining rate is $\lambda_{\mathrm{adv}} < \lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$. The attacker then releases blocks from this chain during the teasing attack. Thereafter, the teasing attack slows down the honest chain growth rate to $\lambda_{\mathrm{grwth}}^{\mathrm{teaser}} < \lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$ ('teasing attack' in Fig. 6). If the adversary's mining rate λ_{adv} exceeds $\lambda_{\mathrm{grwth}}^{\mathrm{teaser}}$, then the adversary can maintain a chain that is longer than the honest chain and continue the attack forever. Since $\lambda_{\mathrm{grwth}}^{\mathrm{teaser}} < \lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$, the teasing attack succeeds with lower adversarial

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

[2]Precisely, the attack succeeds if $\beta \triangleq \frac{\lambda_{\mathrm{adv}}}{\lambda_{\mathrm{adv}}+\lambda_{\mathrm{hon}}} > \frac{\lambda_{\mathrm{grwth}}}{\lambda_{\mathrm{grwth}}+\lambda_{\mathrm{hon}}}$

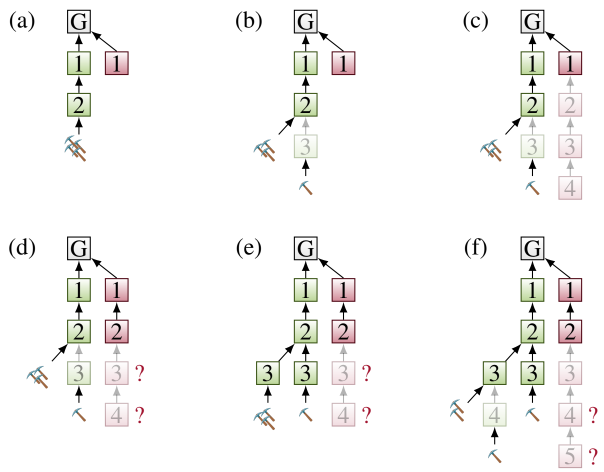

Fig. 6. **Teasing attack:** Green/red are honest/adversarial blocks, and numbers on blocks indicate height in the blockchain. Semi-transparent blocks have been announced (*i.e.*, headers released) but were not yet downloaded by honest nodes. (a) We begin when honest nodes have a chain of length 2. All blocks have been downloaded and validated by honest nodes. The adversary mines a private chain extending its own block at height 1 that it keeps withheld. (b) An honest node builds a block at height $h = 3$, and announces it. The majority of the nodes are still mining on top of the block at height $h - 1$. The adversary wishes to delay the download of the new block. (c) The adversary announces a block at height $h+1$ from the 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 $h = 3$, they find it to be unavailable ('?'), *i.e.* the adversary does not release its content. So honest nodes ignore the rest of the attacker's chain and resume downloading the honest block of height $h = 3$. (e) While download of the honest block was delayed, some mining power may have been wasted and another honest block of height $h = 3$ may have been produced. Notice now that we are in a scenario similar to that in step (a), so the adversary can repeat steps (b) with the next honest block of height $h+1$. (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 the rest of its chain to overtake the honest chain.

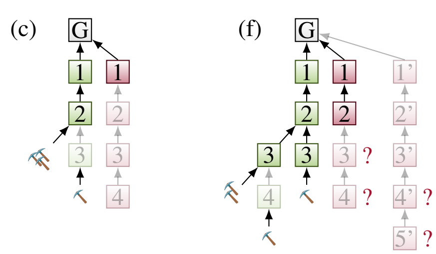

Fig. 7. **Equiv-teasing attack:** Steps (a)-(e) are the same as in the teasing attack (Fig. 6). Recall that in (c), to delay the download of the honest block at height $h = 3$, the adversary announces a block at height $h + 1$ which it was withholding. Since this is the longest announced chain, honest nodes prioritize downloading block 2 on this chain. New steps: (f) Once an honest node produces a block at height 4, the adversary announces equivocations $1', 2', \ldots$ of its withheld chain of length 5 instead of announcing one new block extending the chain $1, 2, \ldots$ Therefore, honest nodes have to process blocks $1', 2'$ even though they processed $1, 2$ earlier. The adversary makes content available for only $1', 2', 3'$. As the attack goes on, the adversary's announced chain gets longer, and it consumes even more of the honest node's processing capacity.

mining power than the private attack. This attack works in both PoW and PoS.

C. The Equiv-Teasing Attack (PoS)

In PoS, the adversary can greatly increase the network's processing load using equivocations. The equiv-teasing attack, described in Fig. 7, uses equivocations to announce a whole new chain at every instance when the teasing attack would have announced a single new block. As the attack goes on, the length of the new announced chain increases. This increases the time honest nodes spend downloading this chain, and *decelerates* the honest chain growth until it comes to a halt. As a result, in Fig. 8, the chain growth rate under the equiv-teasing attack is nearly zero (the small nonzero value is because there is some chain growth at the start).

As in the teasing attack, the adversary starts by producing a private chain. Assuming the adversary's block production rate λ_{adv} is less than the honest chain growth rate before the attack ($\lambda_{\mathrm{grwth}}^{\mathrm{pvt}}$), the probability that the adversary produces a chain of length L before the honest chain reaches length L is $e^{-O(L)}$ [5], [11]. This means that with probability ϵ, the adversary eventually produces a private chain of length $L = O(\log(1/\epsilon))$, of which it can announce equivocations during the attack. Since this chain is longer than the honest chain, it has higher download priority. It takes honest nodes L/C time to download such a chain, during which time, honest nodes do not download blocks on the honest chain. So, any honest blocks produced within L/C time after the first honest block at height h do not grow the honest chain (Fig. 6(e)). If $\lambda_{\mathrm{hon}} L/C$ is large, then there are many honest blocks that do not lead to chain growth, causing the chain growth rate λ_{grwth} to drop (Fig. 8). As in the teasing attack, if the adversary's block production rate λ_{adv} exceeds λ_{grwth}, then the adversary succeeds in maintaining the number of block productions required for the attack to go on forever. This eventually slows honest chain growth to a halt. Thus, if $\lambda_{\mathrm{hon}} L/C$ is large, *i.e.*, $\lambda_{\mathrm{hon}} = \Omega(1/L) = \Omega\left(\frac{1}{\log(1/\epsilon)}\right)$, then the attack succeeds with probability ϵ. A more detailed analysis can be found in App. H-B.

It may seem at first that the above attacks exploit the specific longest-header-chain scheduling policy to tease honest nodes into downloading adversarial blocks. However, even for other scheduling policies, it is possible to devise attack strategies which exploit increased queuing delays and thereby succeed for parameter regimes where the private attack does not succeed. For example, the teasing attack would not succeed if honest nodes 'greedily' prioritized processing blocks that extend their longest chain. But this is vulnerable to a forking attack in which, following a short network split, honest nodes build two separate chains and fail to ever catch up with the other chain. Some such generalizations of our attacks are described in App. H. The overarching conclusion is that models for security analysis must capture effects of adversarial queuing delay.

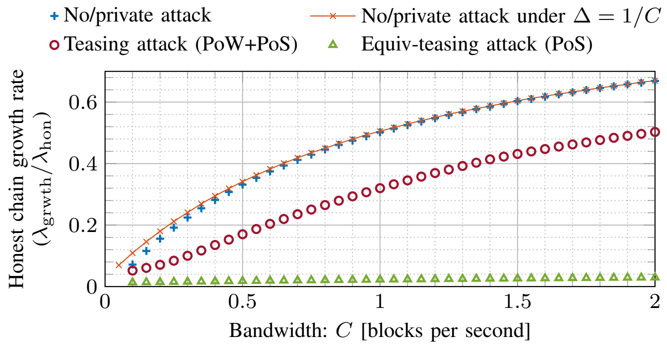

Fig. 8. Simulation results: The rate of chain growth relative to honest block production rate, when nodes prioritize downloads towards the longest header chain, for various bandwidths. When the attacker does not release any blocks (no attack or private attack), we already see $\lambda_{\mathrm{grwth}} < \lambda_{\mathrm{hon}}$ due to natural congestion (+). As described in Sec. II-A, the honest chain growth rate under the private attack is approximately the same whether simulated on a network with finite processing capacity C (+), or on an idealized network with bounded delay $\Delta = 1/C$ (—×—). With a teasing attack, processing is slowed roughly by a factor of 2, which lowers the growth rate of the chain further (o). This lowers security compared to a private attack, cf. Fig. 1. Due to equivocations in PoS, the honest chain grinds to a halt under the equiv-teasing attack (△), implying vulnerability even to adversaries with close to zero stake.

III. Protocol & Model

Pseudocode of an idealized Nakamoto consensus protocol $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ 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 $\mathcal{F}_{\mathrm{hdrtree}}$ (cf. [9, Fig. 2], [19, Alg. 3]). Pseudocode for instantiations $\mathcal{F}^{\mathrm{PoW},\rho}_{\mathrm{hdrtree}}$ and $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$ 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. G-A. We study these protocols in a unified model for a network $\mathcal{Z}$ with finite bandwidth (Fig. 9), and for the powers and limits of an adversary $\mathcal{A}$.

A. Nakamoto's Longest Chain Consensus Protocols

For ease of exposition, the execution features a *static* set of N *equipotent nodes*, each of which runs an independent instance of $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$. 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 [7], [9], [10], [28]. We are interested in the large system regime $N \to \infty$. Nodes interact with each other and with the adversary $\mathcal{A}$ through an environment $\mathcal{Z}$ that models the network and is detailed in Sec. III-B and App. G-B. 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 $\mathrm{d}\mathcal{C}$ in the node's view with a new block of pending transactions txs. If successful, the node disseminates both the resulting *block header* $\mathcal{C}'$ and the associated *block content* txs via the environment $\mathcal{Z}$ to all nodes. Finally, the protocol identifies the k_{conf}-deep prefix $\mathrm{d}\mathcal{C}^{\lceil k_{\mathrm{conf}}}$ containing all but the last k_{conf}

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

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

on INIT(genesis$\mathcal{C}$, numNodes)
$N \leftarrow$ numNodes
$\mathcal{T} \leftarrow$ {genesis$\mathcal{C}$} ▷ *Global set of valid header chains*
on EXTEND($\mathcal{C}$, txs) **from** node P (possibly adversarial) **at** slot t
▷ *Abstraction of proof-of-work lottery: each node can call this once per slot and produces a block with probability ρ/N independently of other nodes and slots*
if lottery[P, t] $\neq \bot$ **return** $\bot$ ▷ *Only one ticket per node and slot*
lottery[P, t] $\overset{\$}{\leftarrow}$ (true with probability ρ/N, else false)
if $\mathcal{C} \in \mathcal{T} \wedge$ lottery[P, t] ▷ *Parent chain $\mathcal{C}$ is valid and lottery was won?*
▷ *Produce a new block header extending $\mathcal{C}$*
$\mathcal{C}' \leftarrow \mathcal{C} \,\|$ newBlock(txsHash: Hash(txs))
$\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{C}'\}$ ▷ *Register new header chain in header tree*
return $\mathcal{C}'$
return $\bot$
on VERIFY($\mathcal{C}$)
return $\mathcal{C} \in \mathcal{T}$ ▷ *Header chain is valid if previously added to header tree*

blocks of $\mathrm{d}\mathcal{C}$. The transactions along $\mathrm{d}\mathcal{C}^{\lceil k_{\mathrm{conf}}}$ are concatenated to produce the output ledger LOG^t.

When a node p receives a new valid block header $\mathcal{C}$ (Alg. 1, l. 6), p adds $\mathcal{C}$ to its header tree $\mathrm{h}\mathcal{T}$, records $\mathcal{C}$ as first seen at the current slot, and relays $\mathcal{C}$ 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 (also called scheduling policy) (Alg. 1, l. 29). As a concrete example, we use the 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 $\mathrm{d}\mathcal{C}$.

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

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

Algorithm 4 Longest-header-chain rule $\mathcal{D}_{\mathsf{long}}$

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

a) Proof-of-Work: The characteristics of PoW-based block production, *e.g.*, in Bitcoin [5], [6], are captured by the idealized functionality $\mathcal{F}^{\mathrm{PoW},\rho}_{\mathrm{hdrtree}}$ (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 $\rho = \Theta(\tau)$, $\tau \to 0$. In turn as $\rho \to 0$, 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.

b) Proof-of-Stake: PoS NC protocols such as from the Ouroboros [7], [8], [15] or Sleepy Consensus [9], [10] families can be modeled using $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$ (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 different parents and/or content, *i.e.*, to equivocate (Alg. 3, ll. 2 and 7). The regime of interest is $\tau = \Theta(1)$.

B. Bandwidth Constrained Network

We borrow the bandwidth constrained network model of [19] (Fig. 9). 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. G-B for a more formal description of $\mathcal{Z}$.

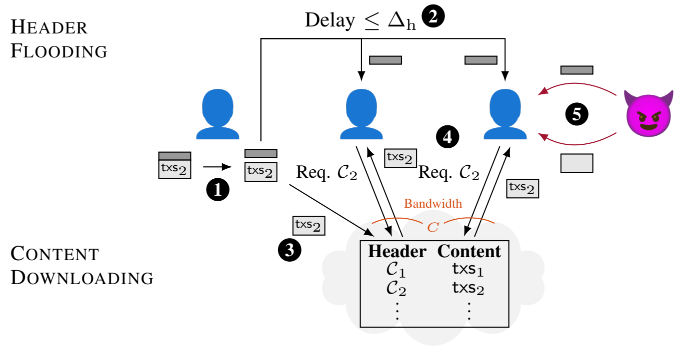

Fig. 9. Bandwidth constrained network model [19, Fig. 4]: ❶ Honest node produces a block, made of header and content. A hash in the header commits to the content. ❷ Header is flooded ($\mathcal{Z}.\textsc{broadcastHeaderChain}$), and arrives at all nodes ($\Pi^{\rho,\tau,k_{\mathrm{conf}}}.\textsc{receivedHeaderChain}$) 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.

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 [29]–[32]. 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).

a) 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

[3]There may be multiple blocks in one slot, as in the Ouroboros [7], [8], [15] and Sleepy Consensus [9], [10] protocols.

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. $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$.EXTEND), and extend chains using past opportunities as long as the purported block production slots along every chain remain strictly increasing.

C. 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^t_p, we recall the security desiderata:

- *Safety:* For all adversarial strategies, for all slots t, t', and for all honest nodes p, q: $\mathsf{LOG}^t_p \preceq \mathsf{LOG}^{t'}_q$ or $\mathsf{LOG}^{t'}_q \preceq \mathsf{LOG}^t_p$.
- *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 $t' \geq t + T_{\mathrm{live}}$, $\mathsf{tx} \in \mathsf{LOG}^{t'}_p$.

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

D. 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. IV-B. *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* 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 $\kappa > \kappa^*_n$, $f(\kappa) < \frac{1}{\kappa^n}$.

IV. Security Analysis

A. 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. IV-C). We then use these properties to prove security of PoW NC (Sec. V) and PoS NC (Sec. VI).

Recall that the protocol runs in discrete units of time of duration τ called slots, and that we consider $\tau \to 0$ 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 $N \to \infty$ and each node holds an equal rate of block production, by the Poisson approximation of a binomial random variable, we have $H_t \overset{\text{i.i.d.}}{\sim} \mathrm{Poisson}((1-\beta)\rho)$ and $A_t \overset{\text{i.i.d.}}{\sim} \mathrm{Poisson}(\beta\rho)$, 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 $\tau \to 0$, the block production process converges to a *Poisson point process*. As noted in Sec. III-A, each BPO corresponds to a different slot, and thus in PoW, blocks in one chain must come from increasing slots. In PoS the latter property is by design (Alg. 3, l. 9).

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 [11], [12], 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.

B. 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 [9], [17], [18], loners [11], and laggers [16]. 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 $\widetilde{C}$ blocks within $\nu + 1$ slots, *i.e.*,

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

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

$$\begin{aligned} \mathsf{Good}(t) \;&\triangleq\; (H_t = 1) \land (A_t = 0) \\ &\quad \land (H_{(t,t+\nu]} + A_{(t,t+\nu]} = 0) \end{aligned} \tag{2}$$

$$\mathsf{Bad}(t) \;\triangleq\; (H_t + A_t > 0) \land \neg\mathsf{Good}(t) \tag{3}$$

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

Note that $\mathsf{Empty}(t) = \neg\mathsf{Good}(t) \wedge \neg\mathsf{Bad}(t)$. 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 $\mathsf{Good}(t_k)$, and $G_k \triangleq 0$ otherwise (*i.e.*, if $\mathsf{Bad}(t_k)$). 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*

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

The proof is by noting that the inter-arrival times between non-empty slots are iid and independent of how many and what kind (honest/adversarial) BPOs occur in that non-empty slot. Details are in App. E-A. Throughout the analysis, we will assume that $p_{\mathrm{G}} > \frac{1}{2}$ ('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 $\mathsf{Good}(t_k)$ 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:

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

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 $\mathsf{PPivot}(k)$, iff:

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

This definition of ppivots captures the *probabilistic* aspects of [9, Def. 5] used in [9, Sec. 5.6.3] and casts them as conditions on a random walk, inspired by [11], [33], to simplify the analysis.

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

$$\mathsf{CPivot}(k) \triangleq (\forall\,(i,j] \ni k\colon Y_{(0,i]} < Y_{(0,k]} \leq Y_{(0,j]}) \tag{8}$$

This definition of cpivots captures the *combinatorial* aspects of [9, Def. 5] used in [9, Sec. 5.6.2] and casts them as conditions on a random walk, inspired by [11], to simplify the analysis. Note that a 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 $L_{\min}(t) = \min_p L_p(t)$.

C. Unified Analysis in the Probabilistic Model

In this section, we develop all the tools needed to prove the safety and liveness of PoW and PoS Nakamoto consensus.

In Sec. IV-C1 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. IV-C2) and the adversary cannot prevent all ppivots from becoming cpivots (Sec. IV-C3). Thus, at least one cpivot occurs in a long enough time interval, the length of which can be set as the confirmation time.

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. 2). 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. E-B using tools similar to [9], [11].

Proposition 2. *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 $\mathsf{CPivot}(k)$. 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 $t \geq t_k + \nu$, $b^* \in \mathrm{d}\mathcal{C}_p(t)$.*

2) Probabilistic Pivots Are Abundant: Sufficiently long intervals of indices contain a number of ppivots proportional to the interval length.

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

$$\Pr\left[\forall\,(i,j] \succeq K_{\mathrm{cp}}\colon P_{(i,j]} \geq (1-\delta)p_{\mathrm{ppivot}}K_{\mathrm{cp}}\right]$$

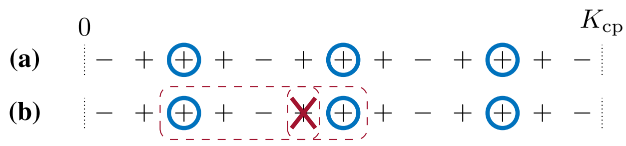

Fig. 10. (a) Example realization of block production opportunities in $(0, K_{\mathrm{cp}}]$, with good $(+)$ and bad $(-)$ indices, and resulting ppivots (O). (b) To prevent a ppivot from being a cpivot, the adversary needs to prevent the timely download (X) of some blocks produced at good indices, so that around the respective ppivot there is an interval in which the cpivot condition (cf. Def. 3) is violated. Here, the first two ppivots are not cpivots. To prevent the timely download of a block from a good index, the adversary must 'spend' $\widetilde{C}$ blocks. Once the adversary runs out of blocks, a ppivot remains cpivot (here the third ppivot).

$$\geq 1 - \exp(-\Omega(\kappa)) = 1 - \mathrm{negl}(\kappa). \qquad (9)$$

The proof is in App. B-B.

3) Many Probabilistic Pivots Imply One Combinatorial Pivot: The longest-header-chain rule $\mathcal{D}_{\mathsf{long}}$ has some useful properties. 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. VI). (P3) holds because the download rule $\mathcal{D}_{\mathsf{long}}$ 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 the highest download priority must have been produced after the most recent cpivot (Prop. 3).

Proposition 3. *If* $G_k = 1$ *and* $D_k = 0$*, then during slots* $[t_k, t_k + \nu]$*, all honest nodes using the download rule* $\mathcal{D}_{\mathsf{long}}$ *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,* $i = 0$*).*

Given the above properties of the download rule, we now want to show that cpivots occur often. Fig. 10 illustrates the key argument for this. To start, 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 (see top row in Fig. 10). 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 indices (so that the $G_k = 1$ indices have $D_k = 0$). Fig. 10 shows an example where the adversary prevented download of the honest block in one good index, and as a result, two of the ppivots fail to become a cpivot. In the proof of Lem. 3, through a combinatorial argument, we show that to prevent all of n ppivots in $(0, K_{\mathrm{cp}}]$ from becoming cpivots, the adversary must prevent download of the honest block in at least $n/4$ good indices in $(0, 2K_{\mathrm{cp}}]$. From Prop. 3, 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 contain at most all blocks mined during $(0, 2K_{\mathrm{cp}}]$. 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. 3 becomes useful. Given that one cpivot occurred in $((m-1)K_{\mathrm{cp}}, mK_{\mathrm{cp}}]$, Prop. 3 ensures that honest nodes will only download blocks that are produced after $(m-1)K_{\mathrm{cp}}$. 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 $(mK_{\mathrm{cp}}, (m+1)K_{\mathrm{cp}}]$. The above arguments are formalized in Lem. 3.

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

$$\forall (i,j] \succeq K_{\mathrm{cp}}\colon \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) > Q_{(i-2K_{\mathrm{cp}},j]}, \qquad (10)$$

$$\forall m \geq 0\colon \frac{\widetilde{C}}{4} P_{(mK_{\mathrm{cp}},(m+1)K_{\mathrm{cp}}]} > Q_{((m-2)K_{\mathrm{cp}},(m+2)K_{\mathrm{cp}}]}, \qquad (11)$$

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

Here, $Q_{(.,.]}$ is the adversary's block budget, and the expressions on the left in eqns. (10) and (11) are the minimum number of blocks the adversary needs to produce to ensure that there are no cpivots, in terms of the number of ppivots $P_{(.,.]}$ and number of good indices $G_{(.,.]}$. Lem. 3 is proven inductively using Prop. 3. The proofs of Prop. 3 and Lem. 3 are in App. B-C.

While all the analysis below is done for the download rule $\mathcal{D}_{\mathsf{long}}$, 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" [19], 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 $(t_k - T_{\mathrm{dl}}, t_k]$" for some T_{dl}. In fact, iii) gives an alternative definition of the property (P4) instead of the one in Prop. 3. 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 briefly mentioned in Sec. II). In Sec. VI, 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.

V. 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 with overwhelming probability in Thm. 1.

As noted in Sec. III-A, 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. IV, 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 $\tau \to 0$, 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. C. Result with $\Delta_{\mathrm{h}} \approx 0$ (reasonable approximation for large block sizes) plotted in Fig. 1.

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

$$\lambda < \max_{\widetilde{C}} \frac{1}{\Delta_{\mathrm{h}} + \widetilde{C}/C} \ln\left(\frac{2(1-\beta)\widetilde{C}}{\widetilde{C} + 4 + \sqrt{8\widetilde{C} + 16}}\right), \quad (12)$$

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

VI. Sanitizing-Proof-of-Stake (SaPoS)

A. 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. D-A.

a) The Download Rule in SaPoS: On top of any existing download rule (such as longest-header-chain), we add another rule that an honest node does not download content for a header $\mathcal{C}$ if it has seen another equivocating header from the same BPO (same producing node and slot) as $\mathcal{C}$. 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 $\mathcal{C}$, and these blocks will be candidates for the node's longest downloaded chain $\mathrm{d}\mathcal{C}$.

b) 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, reaching consensus on just the header chain is not enough. For nodes to later *catch up* to the confirmed header chain's contents, the content must be available in the network. Unfortunately, verifying *data availability* [34] comes with several challenges.

Instead, we ensure that honest nodes agree on which blocks had an equivocation, and 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 $\mathrm{d}\mathcal{C}$, block headers $\mathcal{C}'$ for which the node has seen an equivocating block header $\mathcal{C}'$, and there is no equivocation proof against it in any block header in $\mathrm{d}\mathcal{C}$. The node then creates an equivocation proof which consists of the two block headers $\mathcal{C}$ and $\mathcal{C}'$ 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 $\mathcal{C}$ is considered invalid if it contains an equivocation proof against a block not in the prefix of $\mathcal{C}$, a block more than k_{epf} blocks above $\mathcal{C}$, or contains an equivocation proof that has already been proven in the prefix of $\mathcal{C}$ (Alg. 6, l. 6).

c) 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).

B. Security Theorem

Recall that the analysis in Sec. IV-C3 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 $\mathcal{D}_{\mathsf{long}}$ does not download blocks before the most recent cpivot (Prop. 3), the rule with equivocation removal also does not (P4). This means that the analysis of

Sec. IV-C3 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 are in App. D-B.

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

$$\frac{\widetilde{C}}{16}\frac{(2p_{\mathrm{G}}-1)^2}{p_{\mathrm{G}}} > \frac{\rho}{1-e^{-\rho}}, p_{\mathrm{G}} = (1-\beta)e^{-\frac{\rho}{\tau}(\Delta_{\mathrm{h}}+\widetilde{C}/C)}, \quad (13)$$

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

By choosing $\rho, \tau \to 0$ such that $\rho/\tau = \lambda$ (small slot approximation), we get the same security region as PoW, which is shown in Fig. 1. However, in PoS, we do have the additional freedom to choose a larger τ, offering a potentially larger set of secure parameters. The exact confirmation depth and liveness latency are larger for SaPoS than for PoW NC (details in App. D-B and App. E-F).

VII. Predictable Validity

A. Predictable Transaction Validity

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

The sanitization in SaPoS leads to a loss of *predictable transaction validity*. An honest block B may include a transaction that depends on the contents of a previous block A whose equivocations were not known at the time. After block B is produced, the adversary could release an equivocation for the block A, forcing honest nodes to sanitize block A's contents, which may invalidate the transaction in block B. Such invalidated transactions take up free space in honest blocks and lower the effective throughput (valid confirmed transactions) of the ledger.

Traditional NC protocols, require a node to download and validate blocks before building on them, satisfying predictable validity. On the other hand, protocols which lack state determinism, such as DAG-based [35], [36] and LazyLedger [37] blockchain protocols, choose to forego predictable validity and accept that not all transactions will ultimately be executed.

We propose a simple solution to recover predictable validity for SaPoS: If nodes limit transactions included in a block to those that don't depend on any *recent* state, then they can be sure that all equivocations that could affect the validity state of a transaction have a corresponding equivocation proof included in the chain. This is because at the time of creating a block, honest nodes *have seen all transactions which will be executed*, however, *not all transactions nodes have seen will be executed*. The following lemma follows naturally. See App. F for proof details.

[4] In UTXO-based systems (e.g., Bitcoin), *valid* means the inputs of the transaction have not been spent. In account-based systems (e.g., Ethereum), *valid* means the transaction execution succeeds and fees are paid.

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.*

B. Predictable Fee Validity

In practice, for instance in popular Defi-ecosystems which consist of very interdependent transactions (e.g., transactions interacting with major token exchanges and other prominent smart contracts) [38], [39], it may not always be practical to limit the interaction between transactions. As an alternative to predictable transaction validity, we would like to preserve the minimum requirement that each transaction pays its fee, regardless of the outcome of its execution. This guarantees that miners are compensated for space used in their blocks, and also makes it costly for the adversary to take up space with invalid transactions.

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.

In systems like Ethereum, transactions have a *max gas* value set by the sender, which limits the computation allowed by the transaction and ultimately its fee. We consider a protocol with this gas mechanism, as well as a base transaction cost that covers the block space taken up by the transaction. 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). When a miner includes a transaction, it checks that the account funding the transaction has enough funds to cover the maximum gas, even if all transactions in its recent ancestor blocks make it to the sanitized ledger and consume their maximum gas. Users thus need to maintain a balance proportional to the complexity and frequency of the transactions they make. Thus, users who primarily make simple transactions (direct transfers having low max gas) or transact infrequently (few transactions in recent ancestor blocks) need to maintain smaller balances than those who are spending more on fees. 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.

The solutions in Sec. VII-A and Sec. VII-B are complementary and could each be adopted as per-miner heuristics (i.e., not a consensus rule), or by the system based on the use-case (e.g., expected inter-dependency of transactions). Note that there are user-side complexities both schemes do 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 limited 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).

VIII. Conclusion

In this work we focused on the security of Nakamoto consensus 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). Can better attacks be found?
- Dealing with equivocations in SaPoS came at a cost of higher latency of transaction execution, and decreased certainty about the state at which the transaction is eventually executed. Can these costs be avoided?
- The difficulty adjustment algorithm (DAA) seems to apply even more stress to limited capacity nodes. How should we design DAAs and analyze their security?
- Can processing and download parallelization, pre-processing and pre-fetching of blocks be utilized to securely improve the throughput of NC based protocols?
- In SaPoS, the equivocation proof deadline is not user-dependent, but baked into the protocol. A user cannot increase this to lower error probability. Can we avoid this drawback compared to traditional Nakamoto consensus?

Acknowledgment

We thank Lei Yang, Mohammad Alizadeh, Sundararajan Renganathan, David Mazières, 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] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *P2P*. IEEE, 2013, pp. 1–10.

[2] M. McSweeney. (2021) Solana experiences transaction stoppage as developers report 'intermittent instability'. [Online]. Available: https://www.theblockcrypto.com/linked/117624/solana-experiences-transaction-stoppage-as-developers-report-intermittent-instability

[3] B. Quarmby. (2022) Solana hit with another network incident causing degraded performance. [Online]. Available: https://cointelegraph.com/news/solana-hit-with-another-network-incident-causing-degraded-performance

[4] M. Millard. (2022) Solana restarted after seven-hour outage caused by surge of transactions. [Online]. Available: https://www.theblockcrypto.com/linked/144639/solana-restarted-after-seven-hour-outage-caused-by-surge-of-transactions

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008.

[6] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT (2)*, ser. LNCS, vol. 9057. Springer, 2015, pp. 281–310.

[7] B. David, P. Gazi, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *EUROCRYPT (2)*, ser. LNCS, vol. 10821. Springer, 2018, pp. 66–98.

[8] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *CCS*. ACM, 2018, pp. 913–930.

[9] R. Pass and E. Shi, "The sleepy model of consensus," in *ASIACRYPT (2)*, ser. LNCS, vol. 10625. Springer, 2017, pp. 380–409.

[10] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *Financial Cryptography*, ser. LNCS, vol. 11598. Springer, 2019, pp. 23–41.

[11] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Everything is a race and nakamoto always wins," in *CCS*. ACM, 2020, pp. 859–878.

[12] P. Gazi, A. Kiayias, and A. Russell, "Tight consistency bounds for bitcoin," in *CCS*. ACM, 2020, pp. 819–838.

[13] Bitcoin Project. (2020) Bitcoin developer guide – P2P network – initial block download – headers-first. [Online]. Available: https://web.archive.org/web/20230314181737/https://developer.bitcoin.org/devguide/p2p_network.html#headers-first

[14] IOHK. (2020) input-output-hk/ouroboros-network. [Online]. Available: https://github.com/input-output-hk/ouroboros-network/blob/92723084ce55a706d2b7e9c870861c55fb169491/ouroboros-network/src/Ouroboros/Network/BlockFetch/Decision.hs#L162

[15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *CRYPTO (1)*, ser. LNCS, vol. 10401. Springer, 2017, pp. 357–388.

[16] L. Ren, "Analysis of nakamoto consensus," Cryptology ePrint Archive, Paper 2019/943, 2019. [Online]. Available: https://eprint.iacr.org/2019/943

[17] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *EUROCRYPT (2)*, ser. LNCS, vol. 10211, 2017, pp. 643–673.

[18] L. Kiffer, R. Rajaraman, and A. Shelat, "A better method to analyze blockchain consistency," in *CCS*. ACM, 2018, pp. 729–744.

[19] J. Neu, S. Sridhar, L. Yang, D. Tse, and M. Alizadeh, "Longest chain consensus under bandwidth constraint," in *AFT*. ACM, 2022. [Online]. Available: https://eprint.iacr.org/2021/1545

[20] V. Bagaria, A. Dembo, S. Kannan, S. Oh, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Proof-of-stake longest chain protocols: Security vs predictability," 2019.

[21] A. Kiayias, S. Quader, and A. Russell, "Consistency of proof-of-stake blockchains with concurrent honest slot leaders," in *ICDCS*. IEEE, 2020, pp. 776–786.

[22] Bitcoin Project. Running a full node — Bitcoin. [Online]. Available: https://bitcoin.org/en/full-node#minimum-requirements

[23] M. Fitzi, P. Gaži, A. Kiayias, and A. Russell, "Proof-of-stake blockchain protocols with near-optimal throughput," Cryptology ePrint Archive, Paper 2020/037, 2020. [Online]. Available: https://eprint.iacr.org/2020/037

[24] L. Kiffer, A. Salman, D. Levin, A. Mislove, and C. Nita-Rotaru, "Under the hood of the ethereum gossip protocol," in *Financial Cryptography (2)*, ser. LNCS, vol. 12675. Springer, 2021, pp. 437–456.
[25] S. Sankagiri, S. Gandlur, and B. Hajek, "The longest-chain protocol under random delays," 2021.
[26] T. D. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," *Distributed Comput.*, vol. 33, no. 3-4, pp. 209–225, 2020.
[27] V. Saraph and M. Herlihy, "An empirical study of speculative concurrency in ethereum smart contracts," in *Tokenomics*, ser. OASIcs, vol. 71. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 4:1–4:15.
[28] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," in *CRYPTO (1)*, ser. LNCS, vol. 10401. Springer, 2017, pp. 291–323.
[29] C. Cachin and S. Tessaro, "Asynchronous verifiable information dispersal," in *DISC*, ser. LNCS, vol. 3724. Springer, 2005, pp. 503–504.
[30] J. Hendricks, G. R. Ganger, and M. K. Reiter, "Verifying distributed erasure-coded data," in *PODC*. ACM, 2007, pp. 139–146.
[31] L. Yang, S. J. Park, M. Alizadeh, S. Kannan, and D. Tse, "Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks," in *NSDI*. USENIX Association, 2022, pp. 493–512.
[32] K. Nazirkhanova, J. Neu, and D. Tse, "Information dispersal with provable retrievability for rollups," in *AFT*. ACM, 2022. [Online]. Available: https://eprint.iacr.org/2021/1544
[33] J. Li, D. Guo, and L. Ren, "Close latency-security trade-off for the nakamoto consensus," in *AFT*. ACM, 2021, pp. 100–113.
[34] M. Al-Bassam, A. Sonnino, V. Buterin, and I. Khoffi, "Fraud and data availability proofs: Detecting invalid blocks in light clients," in *Financial Cryptography (2)*, ser. LNCS, vol. 12675. Springer, 2021, pp. 279–298.
[35] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: DAG BFT protocols made practical," in *CCS*. ACM, 2022, pp. 2705–2718.
[36] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and tusk: a dag-based mempool and efficient BFT consensus," in *EuroSys*. ACM, 2022, pp. 34–50.
[37] M. Al-Bassam, "Lazyledger: A distributed data availability ledger with client-side smart contracts," 2019.
[38] D. Guo, J. Dong, and K. Wang, "Graph structure and statistical properties of ethereum transaction relationships," *Inf. Sci.*, vol. 492, pp. 58–71, 2019.
[39] T. Chen, Z. Li, Y. Zhu, J. Chen, X. Luo, J. C. Lui, X. Lin, and X. Zhang, "Understanding ethereum via graph analysis," *ACM Trans. Internet Techn.*, vol. 20, no. 2, pp. 18:1–18:32, 2020.
[40] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500830
[41] J. Duchi. Hoeffding's inequality. [Online]. Available: http://cs229.stanford.edu/extra-notes/hoeffding.pdf
[42] A. Lenz. Random walk with positive drift. [Online]. Available: https://math.stackexchange.com/q/4449213
[43] E. W. Weisstein, "q-Pochhammer Symbol – from Wolfram Mathworld," https://mathworld.wolfram.com/q-PochhammerSymbol.html, 2022, [Online; accessed 10-November-2022].

Appendix A
Simulation Details

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 target. 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. As preemption of downloads may cause nodes to alternate between downloads, we allow nodes to retain partial work in an LRU cache of size 10.

Except where we note otherwise, headers are assumed to propagate instantly in the simulations. To simulate an idealized bounded delay network, we set the header propagation delay to Δ and the capacity of each node to be ∞. Block headers in both PoW and PoS contain the relevant lottery information 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. Our simplified setting allows us to focus more on the congestion effects in isolation from the effects of topology and other P2P related issues.

Appendix B
Security Analysis Proofs

A. Combinatorial Pivots Stabilize

The following proposition is helpful for our proofs. The proof of Lem. 1 is similar to that of [9, Lem. 5], [11, Thm. 3.2], or [19, Lem. 11] and is hence deferred to App. B-A.

Proposition 4. *For any $i < j$,*

$$L_{\min}(t_j + \nu) \geq L_{\min}(t_{i+1} - 1) + D_{(i,j]}. \tag{14}$$

Proof. By noting that if $D_k = 1$, then $t_{k+1} > t_k + \nu$, and adding the result of Prop. 2 for each index with $D_k = 1$. □

B. Probabilistic Pivots Are Abundant

These alternative characterizations of ppivots are insightful:

Proposition 5.

$$\begin{aligned} \mathsf{PPivot}(k) &\iff (\forall\,(i,j] \ni k\colon X_{(i,j]} > 0) && (15)\\ &\iff (\forall\,(i,j] \ni k\colon G_{(i,j]} > \overline{G}_{(i,j]}) && (16)\\ &\iff (X_k = 1) \wedge (\forall j \geq k : X_{(k,j]} \geq 0)\\ &\qquad \wedge (\forall i < (k-1) : X_{(i,k-1]} \geq 0) && (17) \end{aligned}$$

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

In particular, eqn. (17) 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. 11). The process $\{P_k\}$ counts ppivots, with increments $P_k \triangleq \mathbb{1}_{\{\mathsf{PPivot}(k)\}}$.

We build up to the proof of Lem. 2 through a series of propositions. Assume that $p_{\mathrm{G}} = \frac{1}{2} + \varepsilon_{\mathrm{G}}$ with $\varepsilon_{\mathrm{G}} \in (0, 1/2]$

Proposition 6. *With $\alpha_2 \triangleq 2\varepsilon_{\mathrm{G}}^2$, $\forall\,(i,j] : \forall \delta \geq 0$:*

$$\Pr\left[X_{(i,j]} \leq (1-\delta)2\varepsilon_{\mathrm{G}}(j-i)\right] \leq \exp(-\alpha_2 \delta^2 (j-i)). \tag{18}$$

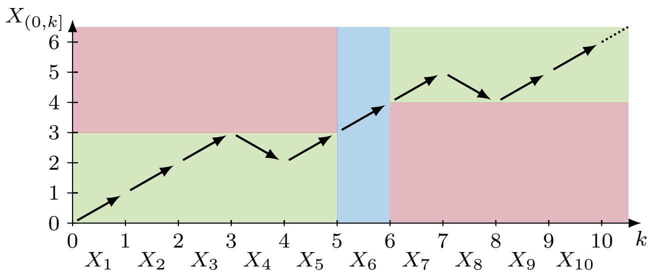

Fig. 11. Illustration of ppivot (eqn. (17)): A ppivot as an index k so that $X_k = 1$ (blue) and $X_{(0,.]}$ is strictly below $X_{(0,k]}$ left of k and weakly above $X_{(0,k]}$ right of k (green), elsewhere (red).

Proposition 7.

$$\forall k\colon \Pr[\mathsf{PPivot}(k)] \geq (2p_{\mathrm{G}}-1)^2/p_{\mathrm{G}} \triangleq p_{\mathrm{ppivot}} \quad (19)$$

Proposition 8. *With* $\alpha_3 \triangleq 2p_{\mathrm{ppivot}}^2$,

$$\forall (i,j] \asymp 2K_1K_2\colon \Pr\left[P_{(i,j]} \leq (1-\delta)p_{\mathrm{ppivot}}2K_1K_2\right] \leq 2K_1\exp(-\alpha_3\delta^2K_2) + K_{\mathrm{hrzn}}^2\exp(-\alpha_2K_1). \quad (20)$$

Proof of Prop. 6 is by Hoeffding's inequality [40] [41, Thm. 4]. Prop. 7 is proved using the probability of a random walk never returning to zero [42]. The proof of Prop. 8 is similar to that of [9, Thm. 5], except that we use a concentration bound to show that there are many ppivots in $(i,j]$ and not just one as shown in [9]. Details are in App. B-B.

Proof of Lem. 2. From Prop. 8 by setting $K_1, K_2 = \Omega(\kappa)$ and $K_{\mathrm{cp}} = 2K_1K_2$. □

C. Many Probabilistic Pivots Imply One Combinatorial Pivot

Proof of Prop. 3. 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 9.

$$\neg\mathsf{CPivot}(k) \implies \exists\,(i,j] \ni k\colon Y_{(i,j]} \leq 0. \quad (21)$$

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

$$\overline{D}_{(i,j]} \geq D_{(i,j]}, \quad (22)$$

$$G_{(i,j]} - D_{(i,j]} \geq \frac{1}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right). \quad (23)$$

Proof. Eqn. (22) is by the definition $Y_i = D_i - \overline{D}_i$. Then,

$$G_{(i,j]} + \overline{G}_{(i,j]} = D_{(i,j]} + \overline{D}_{(i,j]} \quad (24)$$

$$G_{(i,j]} + \overline{G}_{(i,j]} \geq 2D_{(i,j]} \quad (25)$$

$$2G_{(i,j]} - 2D_{(i,j]} \geq G_{(i,j]} - \overline{G}_{(i,j]}. \quad (26)$$

□

Proposition 11. *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.

□

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

$$\forall (i,j] \succeq K_{\mathrm{cp}}, i<K_{\mathrm{cp}}\colon \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) > Q_{(0,j]}, \quad (27)$$

$$\frac{\widetilde{C}}{4}P_{(0,K_{\mathrm{cp}}]} > Q_{(0,2K_{\mathrm{cp}}]}, \quad (28)$$

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

Proof. Due to eqn. (28), there is at least one ppivot in $(0,K_{\mathrm{cp}}]$ (otherwise $P_{(0,K_{\mathrm{cp}}]} = 0$). Suppose for contradiction that there is no cpivot in $(0,K_{\mathrm{cp}}]$. 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:

$$\bigcup_{I\in\mathcal{I}} I \supseteq \{k \in (0,K_{\mathrm{cp}}] : \mathsf{PPivot}(k)\} \quad (29)$$

$$\forall I \in \mathcal{I}\colon Y_I \leq 0 \quad \text{(from Prop. 9).} \quad (30)$$

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. (29) and (30)). 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. 3, 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 $I = (i, j]$,

$$Q_{(0,j]} \geq \widetilde{C}\left(G_{(i,j]} - D_{(i,j]}\right) \tag{31}$$

$$\geq \frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) \quad \text{(from Prop. 10)}. \tag{32}$$

This is a contradiction to eqn. (27).

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

$$G_I - D_I \geq \frac{1}{2}\left(G_I - \overline{G}_I\right) \quad \text{(from Prop. 10)} \tag{33}$$

$$\geq \frac{1}{2}P_I \quad \text{(from Prop. 11)}. \tag{34}$$

Consider the indices $k \in (0, 2K_{\text{cp}}]$ with $G_k = 1$ and $D_k = 0$. Let $\mathcal{I}_k = \{I \in \mathcal{I} \colon k \in I\}$ 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 $\mathcal{I}$ while maintaining eqns. (29) and (30) (see Fig. 12(a)). This process is repeated for all $k \in (0, 2K_{\text{cp}}]$ 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,

$$\sum_{k \in (0, 2K_{\text{cp}}] \colon G_k = 1, D_k = 0} |\mathcal{I}_k| \leq \sum_{k \in (0, 2K_{\text{cp}}] \colon G_k = 1, D_k = 0} 2 \tag{35}$$

$$= 2\left(G_{(0,2K_{\text{cp}}]} - D_{(0,2K_{\text{cp}}]}\right). \tag{36}$$

This sum can be rewritten as

$$\sum_{k \in (0, 2K_{\text{cp}}] \colon G_k = 1, D_k = 0} |\mathcal{I}_k| = \sum_{I \in \mathcal{I}} (G_I - D_I) \tag{37}$$

$$\geq \sum_{I \in \mathcal{I}} \frac{1}{2} P_I \tag{38}$$

$$\geq \frac{1}{2} P_{(0, K_{\text{cp}}]} \quad \text{(eqn. (29))}. \tag{39}$$

Therefore,

$$G_{(0,2K_{\text{cp}}]} - D_{(0,2K_{\text{cp}}]} \geq \frac{1}{4} P_{(0,K_{\text{cp}}]}. \tag{40}$$

This can also be seen from Fig. 12(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. 3). This gives

$$Q_{(0,2K_{\text{cp}}]} \geq \widetilde{C}\left(G_{(0,2K_{\text{cp}}]} - D_{(0,2K_{\text{cp}}]}\right) \tag{41}$$

$$\geq \frac{\widetilde{C}}{4} P_{(0,K_{\text{cp}}]} \tag{42}$$

which is a contradiction to eqn. (28). □

Proof of Lem. 3. This is proved by induction. For the base case ($m = 0$), Lem. 6 shows that $\exists k_1^* \in (0, K_{\text{cp}}] : \mathsf{CPivot}(k_1^*)$. For $m \geq 1$, assume that $\exists k_{m-1}^* \in ((m-1)K_{\text{cp}}, mK_{\text{cp}}]$ such that $\mathsf{CPivot}(k_{m-1}^*)$. Now we want to show that $\exists k_m^* \in (mK_{\text{cp}}, (m+1)K_{\text{cp}}]$ such that $\mathsf{CPivot}(k_m^*)$. The proof for this follows the same steps as the proof of Lem. 6, except that we make use of Prop. 3 to note that honest nodes only download blocks proposed after the most recent cpivot. Then due to the induction assumption, during an interval of indices $(i, j]$ where $i < (m+1)K_{\text{cp}}$, honest nodes only download blocks produced in indices $(i - 2K_{\text{cp}}, j]$. The details of this proof are in App. E-D. □

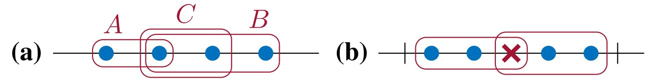

Fig. 12. Blue circles represent ppivots, red crosses represent indices with $G_k = 1$ and $D_k = 0$. (a) Given intervals A, B, C all containing the 2nd blue circle from left, interval C is redundant. (b) Given n blue circles, the adversary needs at least $n/4$ red crosses to draw a set of intervals satisfying eqns. (29) and (30). Here is a placement of red crosses relative to blue circles that achieves the minimum number of red crosses.

Appendix C
Proof-of-Work Security Proofs

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

$$\forall m \geq 0 \colon \exists k_m^* \in (mK_{\text{cp}}, (m+1)K_{\text{cp}}] : \mathsf{CPivot}(k_m^*), \tag{43}$$

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

$$\forall k \in \mathbb{N}, K \geq K_{\text{cp}} \colon t_{k+K} - t_k < \frac{K}{\lambda\tau(1-\delta)}, \tag{44}$$

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

The proof of Lem. 7 is similar to that of the PoS case (in fact, much simpler) which is shown in App. D-B. Eqn. (44) shows an event which occurs with overwhelming probability, and helps to determine the latency in *real time* from the analysis which is in terms of indices. Detailed proof for the PoW case is in App. E-E.

Briefly, to prove Thm. 1, we calculate for given $\Delta_{\text{h}}, C, \widetilde{C}$, the values of λ for which with overwhelming probability, the conditions of Lem. 3 hold. This ensures that cpivots occur and then Lem. 7 proves safety and liveness. Finally, since $\widetilde{C}$ is just an analysis parameter, we optimize over $\widetilde{C}$ to find the maximum λ. Details are in App. E-E.

Appendix D
Proof-of-Stake (SaPoS)

A. Pseudocodes for Equivocation Removal

See Algs. 5 and 6. Changes with respect to Algs. 1 and 2 are highlighted in green.

B. Security Proofs

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

$$\forall m \geq 0 \colon \exists k_m^* \in (mK_{\text{cp}}, (m+1)K_{\text{cp}}] : \mathsf{CPivot}(k_m^*), \tag{45}$$

$$\forall k \in \mathbb{N}, K \geq K_{\text{cp}} \colon t_{k+K} - t_k < \frac{K/(1-e^{-\rho})}{1-\delta}, \tag{46}$$

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

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

▷ *Global counter of time slots* $t \leftarrow 1, 2, \ldots$ *of duration* τ
▷ *Same as in Alg. 1:* INIT(genesis$\mathcal{C}$, genesisTxs), RECEIVEDHEADERCHAIN($\mathcal{C}$), RECEIVEDCONTENT($\mathcal{C}$, txs)
▷ $(\mathcal{C} \overset{BPO}{\equiv} \mathcal{C}') \triangleq (\mathcal{C} \neq \mathcal{C}') \wedge (\mathcal{C}.\mathsf{node} = \mathcal{C}'.\mathsf{node}) \wedge (\mathcal{C}.\mathsf{time} = \mathcal{C}'.\mathsf{time})$
at time slot $t \leftarrow 1, 2, \ldots$ ▷ *NC protocol main loop*
  $\mathsf{txs} \leftarrow \mathcal{Z}$.RECEIVEPENDINGTXS()
  ▷ *Construct equivocation proofs against headers in prefix not already proven*
  $\mathsf{eqProofs} \leftarrow \{(\mathcal{C}_1 \preceq \mathrm{d}\mathcal{C}, \mathcal{C}_2 \in \mathrm{h}\mathcal{T}) \mid (\mathcal{C}_1 \overset{\mathrm{BPO}}{\equiv} \mathcal{C}_2) \wedge (|\mathcal{C}_1| > |\mathrm{d}\mathcal{C}| - k_{\mathrm{epf}}) \wedge ((\mathcal{C}_1, \mathcal{C}_2) \notin \bigcup_{\mathcal{C} \preceq \mathrm{d}\mathcal{C}} \mathcal{C}.\mathsf{eqProofs})\}$
  ▷ *Produce and disseminate a new block if eligible*
  if $\mathcal{C}' \neq \perp$ **with** $\mathcal{C}' \leftarrow \mathcal{F}_{\mathrm{hdrtree}}$.EXTEND($\mathrm{d}\mathcal{C}$, txs, eqProofs)
    $\mathcal{Z}$.UPLOADCONTENT($\mathcal{C}'$, txs)
    $\mathcal{Z}$.BROADCASTHEADERCHAIN($\mathcal{C}'$)
  ▷ *Blank the content of blocks against which there was an equivocation proof*
  $\mathrm{blkTxs}'[\mathcal{C}] \leftarrow (\emptyset$ **if** $(\mathcal{C}, \_) \in \bigcup_{\mathcal{C}' \preceq \mathcal{C}^*} \mathcal{C}'.\mathsf{eqProofs}$, **else** $\mathrm{blkTxs}[\mathcal{C}])$
  ▷ *Confirm all but the last* k_{conf} *blocks on the longest downloaded chain*
  $\mathsf{LOG}^t \leftarrow \mathrm{txsLedger}(\mathrm{blkTxs}, \mathcal{C}^{\lceil k_{\mathrm{conf}}})$ ▷ *Ledger of node* p *at* t*:* LOG^t_p
do throughout
  ▷ Choose $\mathcal{C}$ from download rule (*e.g.* Alg. 4)
  if $\exists \mathcal{C}' \in \mathrm{h}\mathcal{T}\colon \mathcal{C}' \overset{\mathrm{BPO}}{\equiv} \mathcal{C}$
    $\mathrm{blkTxs}[\mathcal{C}] \leftarrow \emptyset$ ▷ *Do not download* $\mathcal{C}$*, query download rule again*
  else
    Download content for $\mathcal{C}$

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

▷ INIT(genesis$\mathcal{C}$, numNodes) *and* VERIFY($\mathcal{C}$) *same as in Alg. 2*
▷ ISLEADER(P, t) *same as in Alg. 3*
on EXTEND($t', \mathcal{C}$, txs, eqProofs) **from** $\mathcal{A}$ (adversary node P) or $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$
  ▷ *New header chain is valid if parent chain* $\mathcal{C}$ *is valid,* P *is leader for slot* t'*, and* t' *is later than the tip of* $\mathcal{C}$ *and is not in the future*
  if $(\mathcal{C} \in \mathcal{T}) \wedge \mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$.ISLEADER$(P, t') \wedge (\mathcal{C}.\mathsf{time} < t' \leq t)$
    ▷ *Check equiv. pfs. are valid, point to 'recent' headers, and do not repeat*
    if $\forall (\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{eqProofs}\colon (\mathcal{C}_1 \preceq \mathcal{C}) \wedge (\mathcal{C}_2 \in \mathcal{T}) \wedge (\mathcal{C}_1 \overset{\mathrm{BPO}}{\equiv} \mathcal{C}_2) \wedge (|\mathcal{C}_1| > |\mathcal{C}| - k_{\mathrm{epf}}) \wedge ((\mathcal{C}_1, \mathcal{C}_2) \notin \bigcup_{\mathcal{C}' \preceq \mathcal{C}} \mathcal{C}'.\mathsf{eqProofs})\}$
      ▷ *Produce a new block header extending* $\mathcal{C}$
      $\mathcal{C}' \leftarrow \mathcal{C} \| \mathrm{newBlock}(\mathsf{time}\colon t', \mathsf{node}\colon P, \mathsf{txsHash}\colon \mathrm{Hash}(\mathsf{txs}))$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{C}'\}$ ▷ *Register new header chain in header tree*
      return $\mathcal{C}'$
  return $\perp$
on EXTEND($\mathcal{C}$, txs, eqProofs) **from** node P (possibly adversarial) **at** time slot t
  return $\mathcal{F}^{\mathrm{PoS},\rho}_{\mathrm{hdrtree}}$.EXTEND($t, \mathcal{C}$, txs, eqProofs)

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 $b_i \in \mathrm{d}\mathcal{C}_p(t)^{\lceil k_{\mathrm{conf}}}$ which was produced in index $i \leq h - k_{\mathrm{conf}}$. From eqn. (45), every interval of $2K_{\mathrm{cp}}$ indices contains at least one cpivot. Therefore for any i, there exist cpivots j, k such that

$$i < j < k \leq i + 4K_{\mathrm{cp}}. \tag{47}$$

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

$$l \geq h - 2K_{\mathrm{cp}} \geq i + k_{\mathrm{conf}} - 2K_{\mathrm{cp}} > i + 4K_{\mathrm{cp}}. \tag{48}$$

From eqn. (47) and eqn. (48), we have

$$i < j < k \leq i + k_{\mathrm{epf}} < l < h. \tag{49}$$

These are shown in Fig. 13. Let b_j, b_k, b_l be the blocks corresponding to the respective cpivots (see Fig. 13). Due to Lem. 1 and $t \geq t_h > t_l + \nu$,

$$b_i \preceq b_j \preceq b_k \preceq b_l \preceq \mathrm{d}\mathcal{C}_p(t) \cap \mathrm{d}\mathcal{C}_q(t'). \tag{50}$$

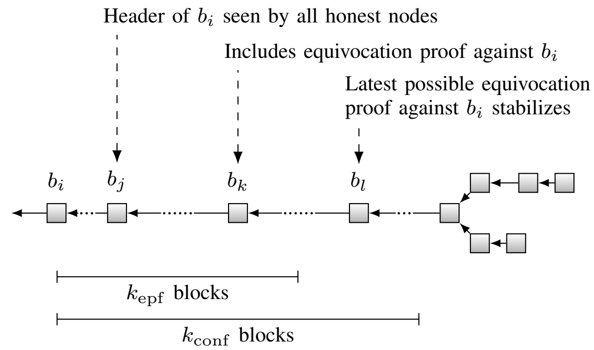

Fig. 13. Illustration for the proof of Lem. 8. Consider an arbitrary block b_i that is k_{conf}-deep in the longest chain of a node. Indices j, k, l are cpivots. Since cpivots stabilize, the corresponding blocks b_j, b_k, b_l are in all honest nodes' longest chains. At cpivot j, we know for sure that 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.

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

$$\mathrm{d}\mathcal{C}_p(t)^{\lceil k_{\mathrm{conf}}} \overset{\preceq}{\succeq} \mathrm{d}\mathcal{C}_q(t')^{\lceil k_{\mathrm{conf}}} \tag{51}$$

where $\mathcal{C}_1 \overset{\preceq}{\succeq} \mathcal{C}_2$ denotes $\mathcal{C}_1 \preceq \mathcal{C}_2$ or $\mathcal{C}_2 \preceq \mathcal{C}_1$. Due to eqn. (51), 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^t_p and $\mathsf{LOG}^{t'}_q$ are consistent, we only need to show that if the content of a block is blanked in LOG^t_p, it is also blanked in $\mathsf{LOG}^{t'}_q$, and conversely if it is not blanked in LOG^t_p, it is not blanked in $\mathsf{LOG}^{t'}_q$.

Suppose that the content of b_i is blanked in LOG^t_p. 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 $\mathrm{d}\mathcal{C}_p(t)$. The header of b_i must be seen by all honest nodes p before the end of slot $t_j + \nu$ (since $b_j \in \mathrm{d}\mathcal{C}_p(t_j + \nu)$). 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 $b_k \in \mathrm{d}\mathcal{C}_q(t')$, so the content of block b_i will be blanked in $\mathsf{LOG}^{t'}_q$ as well.

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

$$|b_l| \geq |\mathrm{d}\mathcal{C}_p(t)| - 2K_{\mathrm{cp}}. \tag{52}$$

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

$$|b_i| \leq |\mathrm{d}\mathcal{C}_p(t)| - k_{\mathrm{conf}}. \tag{53}$$

Together, we have

$$|b_l| \geq |b_i| + k_{\mathrm{conf}} - 2K_{\mathrm{cp}} > |b_i| + k_{\mathrm{epf}}. \tag{54}$$

Therefore b_l or any block extending it cannot contain an equivocation proof against b_i. Since $b_l \in \mathrm{d}\mathcal{C}_p(t)$ and $b_l \in \mathrm{d}\mathcal{C}_q(t')$, there cannot be any block in $\mathrm{d}\mathcal{C}_q(t')$ before b_l, that is not in $\mathrm{d}\mathcal{C}_p(t)$. Therefore, there is no equivocation proof against b_i in $\mathrm{d}\mathcal{C}_q(t')$. 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 $\mathsf{LOG}_q^{t'}$. We can thus conclude that either $\mathsf{LOG}_p^t \preceq \mathsf{LOG}_q^{t'}$ or $\mathsf{LOG}_q^{t'} \preceq \mathsf{LOG}_p^t$. Therefore, safety holds.

Next we 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 $\mathsf{CPivot}(k^*)$. 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,

$$D_{(k^*-1,k^*+2k_{\mathrm{conf}}-1]} > k_{\mathrm{conf}} \tag{55}$$

$$\implies D_{(k^*,k^*+2k_{\mathrm{conf}}-1]} > k_{\mathrm{conf}} - 1. \tag{56}$$

Then from Prop. 4,

$$L_{\min}(t_{k^*+2k_{\mathrm{conf}}-1} + \nu) - L_{\min}(t_{k^*+1} - 1) \geq D_{(k^*,k^*+2k_{\mathrm{conf}}-1]} \geq k_{\mathrm{conf}}. \tag{57}$$

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 $t' \geq t_{k^*+2k_{\mathrm{conf}}-1} + \nu$. Further, the content of an honest block will never be blanked out in any honest node's ledger. Finally, with $k^* \leq h + 2K_{\mathrm{cp}}$ and eqn. (44),

$$\begin{aligned} t_{k^*+2k_{\mathrm{conf}}-1} + \nu - t &\leq t_{h+2K_{\mathrm{cp}}+2k_{\mathrm{conf}}-1} + \nu - t_h \\ &\leq t_{h+2K_{\mathrm{cp}}+2k_{\mathrm{conf}}} - t_h \\ &< \frac{2K_{\mathrm{cp}} + 2k_{\mathrm{conf}}}{(1-e^{-\rho})(1-\delta)}. \end{aligned} \tag{58}$$

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

To prove Thm. 2, we calculate for given $\Delta_{\mathrm{h}}, C, \widetilde{C}$, the values of ρ, τ for which with overwhelming probability, the conditions of Lem. 3 hold. This ensures that cpivots occur and then Lem. 7 proves safety and liveness. Details are in App. E-F.

Appendix E
Proof Details

A. Proof Details for Sec. IV-B

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

$$\Pr[G_k = 1] = \Pr[\mathsf{Good}(t_k) \mid \neg\mathsf{Empty}(t_k)] \tag{59}$$

$$= \frac{\Pr[\mathsf{Good}(t_k)]}{\Pr[\mathsf{Empty}(t_k)]} = \frac{(1-\beta)\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}}. \tag{60}$$

Take an iid random process $\{T_k\}$ with $\Pr[T_k = t] = (1 - p_{\mathrm{E}})p_{\mathrm{E}}^t$ for $t \geq 0$ where $p_{\mathrm{E}} = \Pr[H_t + A_t = 0]$. 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 $\Pr[H_t = 1 \wedge A_t = 0 \mid H_t + A_t > 0]$ 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 $\{(T_k, G'_k)\}$. □

B. Proof Details for Sec. IV-C1

Proof of Lem. 1. Note that $\mathrm{d}\mathcal{C}_p(t)$ is a valid chain at slot t and $|\mathrm{d}\mathcal{C}_p(t)| = L_p(t) \geq L_{\min}(t)$. 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, $|\mathrm{d}\mathcal{C}_q(t_i - 1)| = L_q(t_i - 1) \geq L_{\min}(t_i - 1)$. 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,

$$L_{\min}(t_i + \nu) \geq |b'|. \tag{61}$$

By assumption,

$$|\mathcal{C}'| \geq L_{\min}(s). \tag{62}$$

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,

$$L_{\min}(s) \geq L_{\min}(t_j + \nu) \tag{63}$$

$$\geq L_{\min}(t_{i+1} - 1) + D_{(i,j]} \quad \text{(from Prop. 4)} \tag{64}$$

$$\geq L_{\min}(t_i + \nu) + D_{(i,j]}. \tag{65}$$

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

$$|\mathcal{C}'| \leq |b'| + \overline{D}_{(i,j]}. \tag{66}$$

From eqns. (61), (62), (65) and (66), we derive

$$D_{(i,j]} \leq \overline{D}_{(i,j]} \implies Y_{(i,j]} \leq 0 \implies Y_{(0,i]} < Y_{(0,j]} \tag{67}$$

where $i < k \leq j$.

2) Case 2: $t_j \leq s < t_j + \nu$ 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 $D_l = 1$, $t_l + \nu < t_j \leq s$, 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, $t_j > t_l + \nu$, and $D_{(i,l]} = D_{(i,j-1]}$.

$$\begin{aligned} L_{\min}(s) &\geq L_{\min}(t_j) && (68)\\ &\geq L_{\min}(t_l + \nu) && (69)\\ &\geq L_{\min}(t_{i+1} - 1) + D_{(i,l]} \quad \text{(from Prop. 4)} && (70)\\ &\geq L_{\min}(t_i + \nu) + D_{(i,j-1]}. && (71) \end{aligned}$$

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

$$|\mathcal{C}'| \leq |b'| + \overline{D}_{(i,j-1]}. \tag{72}$$

From eqns. (61), (62), (68) and (72),

$$\begin{aligned} D_{(i,j-1]} \leq \overline{D}_{(i,j-1]} &\implies Y_{(i,j-1]} \leq 0 \\ &\implies Y_{(0,i]} < Y_{(0,j-1]}. \end{aligned} \tag{73}$$

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

In either case, eqn. (67) or eqn. (73) contradict the assumption $\mathsf{CPivot}(k)$ (Def. 3). □

C. Proof Details for Sec. IV-C2

Proposition 12 (Hoeffding's inequality [40] [41, 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, $\forall t \geq 0$:*

$$\Pr\left[\left(\sum_{i=1}^{n} Z_i\right) \geq \mathbb{E}\left[\sum_{i=1}^{n} Z_i\right] + tn\right] \leq \exp\left(\frac{-2nt^2}{(b-a)^2}\right) \tag{74}$$

$$\Pr\left[\left(\sum_{i=1}^{n} Z_i\right) \leq \mathbb{E}\left[\sum_{i=1}^{n} Z_i\right] - tn\right] \leq \exp\left(\frac{-2nt^2}{(b-a)^2}\right) \tag{75}$$

Proof of Prop. 7. Eqn. (17) characterizes $\mathsf{PPivot}(k)$ as the intersection of three independent events:

$$\begin{aligned} \mathcal{E}_1 &\triangleq \{X_k = 1\} && (76)\\ \mathcal{E}_2 &\triangleq \{\forall \ell\colon X_{(k,k+\ell]} \geq 0\} && (77)\\ \mathcal{E}_3 &\triangleq \{\forall \ell\colon X_{(k-1-\ell,k-1]} \geq 0\} && (78) \end{aligned}$$

Their probabilities are easily calculated [42]:

$$\Pr[\mathcal{E}_1] = p_{\mathrm{G}} \qquad \Pr[\mathcal{E}_2] = \Pr[\mathcal{E}_3] = (2p_{\mathrm{G}} - 1)/p_{\mathrm{G}} \tag{79}$$

□

Proof. Let $\mathcal{E} \triangleq \{\forall (i, j] \succeq K_1\colon X_{(i,j]} > 0\}$. From Prop. 6 with $\delta = 1$, and a union bound over all intervals ($\leq K_{\mathrm{hrzn}}^2$ many), we get

$$\Pr[\neg\mathcal{E}] \leq K_{\mathrm{hrzn}}^2 \exp(-\alpha_2 K_1). \tag{80}$$

For any given index k, we can partition the intervals of eqn. (15) into 'long' and 'short' intervals (length at least vs. less than K_1):

$$\begin{aligned} \mathcal{E}_k &\triangleq \{\mathsf{PPivot}(k)\} = \mathcal{E}_k^{\mathrm{L}} \wedge \mathcal{E}_k^{\mathrm{S}} && (81)\\ \mathcal{E}_k^{\mathrm{L}} &\triangleq \{\forall k \in (i, j] \succeq K_1\colon X_{(i,j]} > 0\} && (82)\\ \mathcal{E}_k^{\mathrm{S}} &\triangleq \{\forall k \in (i, j] \prec K_1\colon X_{(i,j]} > 0\}. && (83) \end{aligned}$$

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

$$\begin{aligned} \forall \ell \in \{1, ..., 2K_1\}\colon\ I_\ell \triangleq \{&i + 0 \cdot 2K_1 + \ell, ... \\ &..., i + (K_2 - 1) \cdot 2K_1 + \ell\}. \end{aligned} \tag{84}$$

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

$$\begin{aligned} \mathcal{E}^* &\triangleq \{P_{I^*} \leq (1-\delta) p_{\mathrm{ppivot}} 2K_1K_2\} && (85)\\ \mathcal{E}_\ell &\triangleq \{P_{I_\ell} \leq (1-\delta) p_{\mathrm{ppivot}} K_2\}. && (86) \end{aligned}$$

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

$$\Pr[\mathcal{E}^* \mid \mathcal{E}] \leq \sum_{\ell=1}^{2K_1} \Pr[\mathcal{E}_\ell \mid \mathcal{E}]. \tag{87}$$

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

$$\begin{aligned} \Pr[\mathcal{E}_\ell \mid \mathcal{E}] &= \Pr[P_{I_\ell} \leq (1-\delta) p_{\mathrm{ppivot}} K_2 \mid \mathcal{E}] && (88)\\ &\overset{\text{(a)}}{\leq} \Pr[P_{I_\ell} \leq (1-\delta)\mu_\ell \mid \mathcal{E}] && (89)\\ &\overset{\text{(b)}}{\leq} \exp(-2\delta^2\mu_\ell^2/K_2) \overset{\text{(c)}}{\leq} \exp(-2p_{\mathrm{ppivot}}^2 \delta^2 K_2), && (90) \end{aligned}$$

where (a) and (c) use

$$\begin{aligned} \mu_\ell = K_2 \mathbb{E}\left[\mathbb{1}_{\{\mathsf{PPivot}(k)\}} \,\middle|\, \mathcal{E}\right] &\geq K_2 \mathbb{E}\left[\mathbb{1}_{\{\mathsf{PPivot}(k)\}}\right] \\ &\geq K_2 p_{\mathrm{ppivot}} \end{aligned} \tag{91}$$

(Prop. 7), 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. 12).

Thus, we complete the proof by observing, that with $\alpha_3 = 2p_{\mathrm{ppivot}}^2$,

$$\begin{aligned} \Pr[\mathcal{E}^*] &= \Pr[\mathcal{E}^* \cap \mathcal{E}] + \Pr[\mathcal{E}^* \cap \neg\mathcal{E}] && (92)\\ &\leq \Pr[\mathcal{E}^* \mid \mathcal{E}] + \Pr[\neg\mathcal{E}] && (93)\\ &\leq 2K_1 \exp(-\alpha_3 \delta^2 K_2) + K_{\mathrm{hrzn}}^2 \exp(-\alpha_2 K_1). && (94) \end{aligned}$$

□

D. Proof Details for Sec. IV-C3

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

For $m \geq 1$, assume that $\exists k_{m-1}^* \in ((m-1)K_{\mathrm{cp}}, mK_{\mathrm{cp}}]$ such that $\mathsf{CPivot}(k_{m-1}^*)$. Now we want to show that $\exists k_m^* \in$

$(mK_{\rm cp},(m+1)K_{\rm cp}]$ such that $\mathsf{CPivot}(k_m^*)$. Suppose for contradiction that there is no cpivot in $(mK_{\rm cp},(m+1)K_{\rm cp}]$. As in the proof of Lem. 6, there is a set of intervals $\mathcal{I}$ such that:

$$\bigcup_{I\in\mathcal{I}} I \supseteq \{k\in(mK_{\rm cp},(m+1)K_{\rm cp}] : \mathsf{PPivot}(k)\} \quad (95)$$

$$\forall I\in\mathcal{I}\colon Y_I\leq 0. \quad (96)$$

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

First, consider the large intervals with $|I|\geq K_{\rm cp}$. Consider indices $k\in I$ for which $G_k=1$ (good) but $D_k=0$ ($\overline{D}$-slot). From Prop. 3, for each such index k, all honest nodes download $\widetilde{C}$ blocks that are produced in the interval $\left(k_{m-1}^*,k\right]$. 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 $I=(i,j]$,

$$Q_{\left(k_{m-1}^*,j\right]}\geq \widetilde{C}\left(G_{(i,j]}-D_{(i,j]}\right) \quad (97)$$

$$\geq \frac{\widetilde{C}}{2}\left(G_{(i,j]}-\overline{G}_{(i,j]}\right) \quad \text{(from Prop. 10)}. \quad (98)$$

But $k_{m-1}^*>(m-1)K_{\rm cp}$ and $i<(m+1)K_{\rm cp}$. Therefore $Q_{\left(k_{m-1}^*,j\right]}\leq Q_{(i-2K_{\rm cp},j]}$. Then we have a contradiction to eqn. (10).

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

$$G_I-D_I\geq\frac{1}{2}\left(G_I-\overline{G}_I\right)\geq\frac{1}{2}P_I \quad \text{(Props. 10 and 11)} \quad (99)$$

Consider the indices $k\in((m-1)K_{\rm cp},(m+1)K_{\rm cp}]$ 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,

$$\sum_{k\in((m-1)K_{\rm cp},(m+1)K_{\rm cp}]\colon G_k=1,D_k=0}|\mathcal{I}_k|$$
$$\leq 2\left(G_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}-D_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}\right). \quad (100)$$

This sum can be rewritten as

$$\sum_{k\in((m-1)K_{\rm cp},(m+1)K_{\rm cp}]\colon G_k=1,D_k=0}|\mathcal{I}_k| \quad (101)$$
$$=\sum_{I\in\mathcal{I}}(G_I-D_I) \quad (102)$$
$$\geq\sum_{I\in\mathcal{I}}\frac{1}{2}P_I \quad (103)$$
$$\geq\frac{1}{2}P_{(mK_{\rm cp},(m+1)K_{\rm cp}]}. \quad (104)$$

Therefore,

$$G_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}-D_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}$$
$$\geq\frac{1}{4}P_{(mK_{\rm cp},(m+1)K_{\rm cp}]}. \quad (105)$$

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 $(m-1)K_{\rm cp}$. By induction assumption, we have a cpivot $k_{m-2}^*\in((m-2)K_{\rm cp},(m-1)K_{\rm cp}]$. This gives

$$Q_{((m-2)K_{\rm cp},(m+1)K_{\rm cp}]}$$
$$\geq\widetilde{C}\left(G_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}-D_{((m-1)K_{\rm cp},(m+1)K_{\rm cp}]}\right) \quad (106)$$
$$\geq\frac{\widetilde{C}}{4}P_{(mK_{\rm cp},(m+1)K_{\rm cp}]} \quad (107)$$

which is a contradiction. □

E. Proof Details for Sec. V

Proposition 13.

$$\forall k,K\in\mathbb{N}\colon \Pr\left[\tau(t_{k+K}-t_k)\geq\frac{K}{\lambda(1-\delta)}\right]\leq e^{\frac{-K\delta^2}{2(1+\delta)}}. \quad (108)$$

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. □

Proof of Lem. 7. Safety: For an arbitrary slot t, let k be the largest index such that $t_k\leq t$. From eqn. (43), every interval of $2K_{\rm cp}$ indices contains at least one cpivot. Therefore, there exists $k^*\in(k-2K_{\rm cp}-1,k-1]$ 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 $k^*\geq k-k_{\rm conf}$, so the block b^* cannot be $k_{\rm conf}$-deep in any chain at slot t Therefore, LOG_p^t is a prefix of b^* which in turn is a prefix of $\mathrm{d}\mathcal{C}_q(t')$. We can thus conclude that either $\mathsf{LOG}_p^t\preceq\mathsf{LOG}_q^{t'}$ or $\mathsf{LOG}_q^{t'}\preceq\mathsf{LOG}_p^t$. 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 $k^*\in(k,k+2K_{\rm cp}]$ 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,

$$D_{(k^*-1,k^*+2k_{\rm conf}-1]}>k_{\rm conf} \quad (109)$$
$$\implies D_{(k^*,k^*+2k_{\rm conf}-1]}>k_{\rm conf}-1. \quad (110)$$

Then from Prop. 4,

$$L_{\min}(t_{k^*+2k_{\rm conf}-1}+\nu)-L_{\min}(t_{k^*+1}-1)\geq D_{(k^*,k^*+2k_{\rm conf}-1]}$$
$$\geq k_{\rm conf}. \quad (111)$$

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 $k_{\rm conf}$-deep in $\mathrm{d}\mathcal{C}_p(t')$ for all honest nodes p and all $t'\geq t_{k^*+2k_{\rm conf}-1}+\nu$. Finally, with $k^*\leq k+2K_{\rm cp}$ and eqn. (44),

$$t_{k^*+2k_{\rm conf}-1}+\nu-t\leq t_{k+6K_{\rm cp}+1}+\nu-t_k$$
$$\leq t_{k+6K_{\rm cp}+2}-t_k$$
$$<\frac{6K_{\rm cp}+2}{\lambda\tau(1-\delta)}. \quad (112)$$

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

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

$$\mathcal{E}_1 = \left\{\forall\,(i,j] \succeq K_{\mathrm{cp}} \colon P_{(i,j]} > (1-\delta)p_{\mathrm{ppivot}}(j-i)\right\} \quad (113)$$

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

$$\begin{aligned}\forall\,(i,j] \succeq K_{\mathrm{cp}} \colon \frac{\widetilde{C}}{4}P_{(i,j]} &> \frac{\widetilde{C}}{4}(1-\delta)p_{\mathrm{ppivot}}(j-i) && (114)\\ &> 4(j-i) && (115)\\ &\overset{\text{(a)}}{=} Q_{(i-2K_{\mathrm{cp}},j+K_{\mathrm{cp}}]} && (116)\end{aligned}$$

where (a) is because as $\tau \to 0$, each non-empty slot has exactly one BPO. This satisfies eqn. (11) in Lem. 3. Further,

$$\frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) \geq \frac{\widetilde{C}}{2}P_{(i,j]} > 3(j-i) > Q_{(i-2K_{\mathrm{cp}},j]} \quad (117)$$

which satisfies condition eqn. (10) in Lem. 3. Therefore there is one cpivot in every interval of the form $(mK_{\mathrm{cp}},(m+1)K_{\mathrm{cp}}]$. Also suppose the following event occurs:

$$\mathcal{E}_2 = \left\{\forall k \in \mathbb{N}, K \geq K_{\mathrm{cp}} \colon t_{k+K} - t_k < \frac{K}{\lambda\tau(1-\delta)}\right\}. \quad (118)$$

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

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

$$\begin{aligned}p_{\mathrm{G}} = (1-\beta)\frac{\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}} &\to (1-\beta)e^{-\lambda\left(\Delta_{\mathrm{h}}+\widetilde{C}/C\right)}, && (119)\\ \frac{\widetilde{C}}{16}\frac{(2p_{\mathrm{G}}-1)^2}{p_{\mathrm{G}}}&(1-\delta) > 1 && (120)\end{aligned}$$

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 $\delta \to 0$ as we can increase the latency through increasing K_{cp} to still satisfy the error bounds. Maximizing over $\widetilde{C}$ from eqns. (120) and (119) gives the resulting threshold. □

F. Proof Details for Sec. VI

The proposition below helps to derive the confirmation latency in units of real time from the analysis.

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

$$\Pr\left[t_{k+K} - t_k \geq \frac{K/(1-e^{-\rho})}{1-\delta}\right] \leq e^{-2K(1-e^{-\rho})\delta^2}, \quad (121)$$

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

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.

$$\forall t, T \in \mathbb{N} \colon \Pr\left[\sum_{r=t}^{t+T}(H_t + A_t) \geq \rho T(1+\delta)\right] \leq e^{\frac{-\rho T\delta^2}{2(1+\delta)}}. \quad (122)$$

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.

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

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

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

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

$$\begin{aligned}\forall\,(i,j] \succeq K_{\mathrm{cp}} \colon \quad Q_{(i,j]} &\triangleq \sum_{k=i+1}^{j}(H_{t_k} + A_{t_k}) && (123)\\ \text{with } T = \frac{j-i}{(1-e^{-\rho})(1-\delta)}, \quad &\leq \sum_{t=t_i}^{t_i+T}(H_t + A_t) && (124)\\ &< \frac{\rho(j-i)(1+\delta)}{(1-e^{-\rho})(1-\delta)} && (125)\\ &\leq \frac{\rho(j-i)}{(1-e^{-\rho})(1-2\delta)}. && (126)\end{aligned}$$

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

$$\begin{aligned}\forall\,(i,j] \succeq K_{\mathrm{cp}} \colon \quad \frac{\widetilde{C}}{4}P_{(i,j]} &> \frac{\widetilde{C}}{4}(1-2\delta)p_{\mathrm{ppivot}}(j-i) && (127)\\ &> \frac{4\rho(j-1)(1-2\delta)}{(1-e^{-\rho})(1-4\delta)} && (128)\\ &> \frac{4\rho(j-1)}{(1-e^{-\rho})(1-2\delta)} && (129)\\ &> Q_{(i-2K_{\mathrm{cp}},j+K_{\mathrm{cp}}]}. && (130)\end{aligned}$$

This satisfies eqn. (11) in Lem. 3. Further,

$$\begin{aligned}\frac{\widetilde{C}}{2}\left(G_{(i,j]} - \overline{G}_{(i,j]}\right) &\geq \frac{\widetilde{C}}{2}P_{(i,j]} && (131)\\ &> \frac{3\rho(j-1)}{(1-e^{-\rho})(1-2\delta)} && (132)\\ &> Q_{(i-2K_{\mathrm{cp}},j]} && (133)\end{aligned}$$

which satisfies condition eqn. (10) in Lem. 3. Therefore there is one cpivot in every interval of the form $(mK_{\mathrm{cp}},(m+1)K_{\mathrm{cp}}]$. Then by Lem. 8, the protocol achieves safety and liveness with appropriately chosen $k_{\mathrm{conf}}, k_{\mathrm{epf}}, T_{\mathrm{live}}$.

By using Lem. 2, $K_{\mathrm{cp}} = \Omega(\kappa^2)$, $K_{\mathrm{hrzn}} = \mathrm{poly}(\kappa)$, 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 $(\nu+1)\tau \geq \Delta_{\mathrm{h}} + \widetilde{C}/C$, $p_{\mathrm{G}} = (1-\beta)\frac{\rho e^{-\rho(\nu+1)}}{1-e^{-\rho}}$, $\frac{C}{16}p_{\mathrm{ppivot}} > \frac{\rho}{1-e^{-\rho}}$, and $p_{\mathrm{ppivot}} = (2p_{\mathrm{G}}-1)^2/p_{\mathrm{G}}$,. As in the case of PoW, $\widetilde{C}$ is a free parameter that can be optimized to find the best set of parameters. □

Appendix F
Transaction Validity Proofs

Proof of Lem. 4. In Sec. VI-A, 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 $\mathcal{T}_{\mathsf{acc}(k_{\mathrm{epf}})}$ 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 $\mathcal{T}_{\mathsf{acc}(k_{\mathrm{epf}})}$, 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. □

Appendix G
Protocol Algorithms Reference

A. Helper Functions for Pseudocode

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

B. 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 $\mathcal{Z}.\mathrm{blkTxs}$ from block headers to the block content (transactions). This mapping is referred to as the 'cloud' in Sec. III and Fig. 9. $\mathcal{Z}$ 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:

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

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

Appendix H
Congestion-Based Attacks

A. Forking Attack

The teasing attack relied strongly on the fact that the attacker could entice nodes with a long header chain that is

[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.

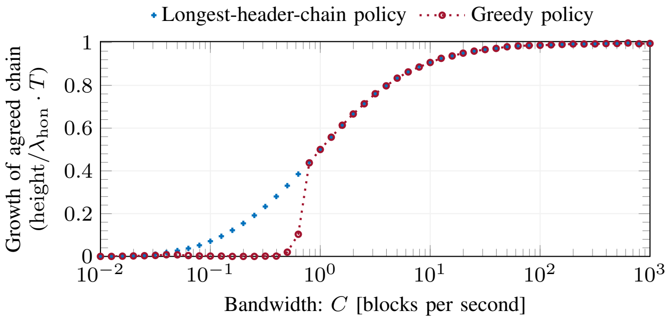

Fig. 14. The rate nodes grow the agreed chain after the network splits into two sets of 50 nodes for 15 secs, when the download rule is "longest-header-chain" (•) or "greedy" (···•···). Nodes using the greedy policy prioritize downloads on their current chain. Under low bandwidth, they do not recover 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)).

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.

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.

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.[6] 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 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. 14) 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.

B. *Equiv-Teasing Attack*

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 NC protocol without an equivocation removal policy. For concreteness, we demonstrate this attack on PoS NC 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, [19, Figure 3] showed one attack and the attack in this section generalizes that. On the other hand, [19] proved PoS NC 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 $1 - \operatorname{negl}(\kappa)$. 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 [19, Figure 3] required that the PoS NC protocol rejects blocks with invalid transactions after downloading them, this attack does not require that. Therefore, this attack works even if the PoS NC 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. VII, 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. [19, 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 ([19, Alg. 4]), the node first filters the tree

[6]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.

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.

C. Attack Strategy

The attack works in two phases. See Fig. 15 for reference. C is the bandwidth constraint (in blocks per second), τ is the slot duration, and κ is the security parameter.

a) Setup phase: At time slot t_0, the adversary creates a chain $\mathcal{C}$ which forks off the honest chain $\mathcal{C}_0$ by at least $L = \log(\kappa)$ blocks, and is at least as long as $\mathcal{C}_0$. The prefix length L is chosen so that the setup succeeds with non-negligible probability. The adversary initially keeps $\mathcal{C}$ private.

b) Execution phase: The adversary creates different chains $\mathcal{C}_1, \mathcal{C}_2, \ldots$ which contain equivocations of the blocks in $\mathcal{C}$, 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 $\mathcal{C}_i$. 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 $\mathcal{C}$. 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 $\mathcal{C}$. The adversary makes the following updates,
   - $\mathcal{C}_0 \leftarrow$ chain ending in b^*,
   - $\mathcal{C} \leftarrow$ chain ending in b',
   - $t_0 \leftarrow t^*$,
   - $L \leftarrow L + 1$,

   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.

D. Analysis Overview

The analysis below reuses notation defined in Sec. IV-B. For the attack to succeed, we assume the following:

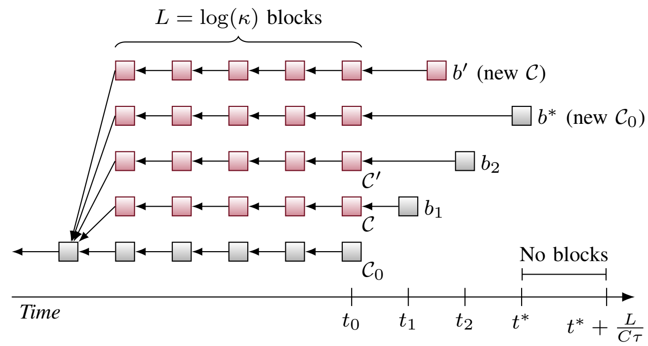

Fig. 15. Illustration of the new attack of Section H-D. At time t_0, $\mathcal{C}_0$ is the longest downloaded chain of all honest nodes, and the adversary produces a chain $\mathcal{C}$ that forks off $\mathcal{C}_0$ by $L = \log(\kappa)$ blocks. Blocks produced in time slots $t_1, t_2, \ldots$ (whether honest or adversarial) extend the chain $\mathcal{C}$ or a chain $\mathcal{C}'$ containing equivocation of the blocks in $\mathcal{C}$, 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 $\mathcal{C}_0$ and the chain ending in b' as new $\mathcal{C}$, and repeats the attack.

- 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 $e^{-O(L)} = 1/\operatorname{poly}(\kappa)$ 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 $\operatorname{poly}(\kappa)$.

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 $L \leftarrow L+1$. 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 App. H-C (3) occurs.

E. Analysis Details

Building up on the definitions from Sec. IV-B, 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

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

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 $\mathcal{C}$ 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 $|\mathrm{d}\mathcal{C}_p(r)| = |b|$ 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 App. H-C 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 Lem. 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 $t_\mathrm{a} > t_0 + T_\mathrm{conf}$, t_a is a successful time slot, and there are no blocks produced in slots $(t_\mathrm{a}, t_\mathrm{a} + L'/(C\tau)]$ where L' is the value of the attacker's parameter L at time slot $t_0 + T_\mathrm{conf}$. If the attacker does not terminate before slot t_a, then there is a safety violation.*

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

Lemma 13. *For all t,*

$$\Pr\left[F_t\right] \geq p_\mathrm{H} \left(1 - 2e^{-L/9}\right) \frac{1}{\sqrt{8L}} e^{-4\alpha_4 L}, \tag{134}$$

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

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

$$\begin{aligned}
&\Pr\left[F_t\right] \\
&= \Pr\left[\exists r < t\colon (H_r > 0) \wedge (\mathcal{A}_{(r,t]} \geq \mathcal{H}_{(r,t]}) \wedge (\mathcal{A}_{(r,t]} \geq L)\right] \\
&\geq \Pr\left[H_s > 0 \wedge \mathcal{A}_{(s,t]} \geq \mathcal{H}_{(s,t]} \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&= \Pr\left[H_s > 0\right] \Pr\left[\mathcal{A}_{(s,t]} \geq \mathcal{H}_{(s,t]} \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&\geq \Pr\left[H_s > 0\right] \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge \mathcal{A}_{(s,t]} \geq L\right] \\
&\overset{(a)}{\geq} \Pr\left[H_s > 0\right] \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge \mathcal{B}_{(s,t]} \geq 2L\right] \\
&\geq p_\mathrm{H} \Pr\left[\mathcal{H}_{(s,t]} \leq L \wedge 2L \leq \mathcal{B}_{(s,t]} \leq 2L(1+2\epsilon)\right] \\
&\geq p_\mathrm{H} \Pr\left[2L \leq \mathcal{B}_{(s,t]} \leq 2L(1+2\epsilon)\right] \\
&\quad \Pr\left[\mathcal{H}_{(s,t]} \leq L \mid \mathcal{B}_{(s,t]} = 2L(1+2\epsilon)\right]
\end{aligned} \tag{135}$$

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

$$\Pr\left[\mathcal{B}_{(s,t]} < p(t-s)(1-\delta)\right] \leq \exp\left(-\frac{p(t-s)\delta^2}{2}\right), \tag{136}$$

$$\Pr\left[\mathcal{B}_{(s,t]} > p(t-s)(1+\delta)\right] \leq \exp\left(-\frac{p(t-s)\delta^2}{3}\right). \tag{137}$$

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

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

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

$$\Pr\left[2L \leq \mathcal{B}_{(s,t]} \leq 2L(1+2\epsilon)\right] \geq 1 - 2\exp\left(\frac{-2L\epsilon^2}{3(1+\epsilon)}\right). \tag{138}$$

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

$$\begin{aligned}
&\Pr\left[\mathcal{H}_{(s,t]} \leq L \mid \mathcal{B}_{(s,t]} = 2L(1+\epsilon)\right] \\
&\geq \frac{1}{\sqrt{4L(1+2\epsilon)}} \exp\left(-2\alpha_4 L(1+2\epsilon)\right)
\end{aligned} \tag{139}$$

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

$$D(x||y) = x\ln\left(\frac{x}{y}\right) + (1-x)\ln\left(\frac{1-x}{1-y}\right). \tag{140}$$

Putting these together,

$$\Pr\left[F_t\right] \geq p_\mathrm{H}\left(1 - 2e^{\frac{-2L\epsilon^2}{3(1+\epsilon)}}\right) \frac{1}{\sqrt{4L(1+2\epsilon)}} e^{-2\alpha_4 L(1+2\epsilon)}. \tag{141}$$

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

Corollary 1. *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

$$\Pr\left[\mathrm{Loner}_T(t) \mid H_t + A_t > 0\right] = (1-p)^T. \tag{142}$$

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 App. H-C is successful and that the adversary can updates it state and continue the attack. Let $t_1, t_2, ...$ 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).

$$\begin{aligned}
&\Pr\left[\exists i \leq N\colon A_{t_i} > 0\right] \\
&= \sum_{k=1}^{\infty} \Pr\left[N = k\right] \Pr\left[\exists i \leq k\colon A_{t_i} > 0 \mid N = k\right].
\end{aligned} \tag{143}$$

Here,

$$\begin{aligned}\Pr\left[N=k\right] &= \prod_{i=1}^{k-1}\Pr\left[\neg\mathrm{Loner}_T(t_i)\mid H_{t_i}+A_{t_i}>0\right]\\ &\quad \Pr\left[\neg\mathrm{Loner}_T(t_k)\mid H_{t_k}+A_{t_k}>0\right]\\ &= \left(1-(1-p)^T\right)^{k-1}(1-p)^T. \end{aligned} \tag{144}$$

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

$$\begin{aligned}&\Pr\left[\exists i\le k\colon A_{t_i}>0\mid N=k\right]\\ &= 1-\prod_{i=1}^{k}\Pr\left[A_{t_i}=0\mid A_{t_i}+H_{t_i}>0\right]\\ &= 1-\left(\frac{e^{-\beta\rho}(1-e^{-(1-\beta)\rho})}{(1-e^{-\rho})}\right)^k\\ &= 1-\left(1-\frac{1-e^{-\beta\rho}}{1-e^{-\rho}}\right)^k\\ &\ge 1-(1-\beta)^k \end{aligned} \tag{145}$$

Putting them together,

$$\begin{aligned}&\Pr\left[\exists i\le N\colon A_{t_i}>0\right] \qquad (146)\\ &\ge \sum_{k=1}^{\infty}\left(1-(1-p)^T\right)^{k-1}(1-p)^T\left(1-(1-\beta)^k\right)\\ &= 1-\frac{(1-p)^T(1-\beta)}{1-(1-(1-p)^T)(1-\beta)}\\ &\ge 1-\frac{(1-p)^T(1-\beta)}{\beta}. \end{aligned} \tag{147}$$

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 Cor. 1, 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

$$\begin{aligned}\Pr\left[\text{attack continues forever}\right] &\ge \prod_{l=L}^{\infty}\left(1-\frac{(1-\beta)e^{-\frac{\rho l}{C\tau}}}{\beta}\right)\\ &\ge \prod_{l=L}^{\infty}\left(1-e^{-\frac{\rho(l-L)}{C\tau}}\right)\\ &= \prod_{l=1}^{\infty}\left(1-e^{-\frac{\rho l}{C\tau}}\right)\\ &= \left(e^{-\frac{\rho}{C\tau}};e^{-\frac{\rho}{C\tau}}\right)_\infty. \end{aligned} \tag{148}$$

Here, $(x;x)_\infty$ is called the q-Pochhammer symbol and $(x;x)_\infty\in(0,1)$ for all $x\in(0,1)$ [43]. The condition $\frac{\rho}{\tau}>\frac{C}{L}\log\frac{1-\beta}{\beta}$ is derived from the condition in Lem. 14 with $T=\frac{L}{C\tau}$. □

Corollary 2. *For the protocol $\Pi^{\rho,\tau,k_{\mathrm{conf}}}$ 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 $\frac{1-e^{-\rho}}{\tau}\le\frac{\rho}{\tau}\le\frac{C}{\log\kappa}\log\frac{1-\beta}{\beta}$. If this is not true, then the attacker never terminates, hence there is a safety violation as per Lem. 12. The maximum block production rate $\lambda=\frac{\rho}{\tau}$ calculated from Lem. 15 is plotted in Fig. 1(b).