cryptography

MDPI

Article

An Incentive-Compatible Semi-Parallel Proof-of-Work Protocol †

Mustafa Doger * and Sennur Ulukus

Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742, USA; ulukus@umd.edu

* Correspondence: doger@umd.edu

† This manuscript is an extended version of our conference paper, "Semi-Parallel Proof-of-Work: A Practical Protocol with Improved Incentive Compatibility," published in IEEE ICDCS 2026.

Abstract

Parallel Proof-of-Work (PoW) protocols have been suggested in the literature to improve the safety guarantees, transaction throughput, and confirmation latencies of Nakamoto consensus. In this work, we first consider the existing parallel PoW protocols and develop hard-coded incentive attack structures. Our theoretical results and simulations show that the existing parallel PoW protocols are more vulnerable to incentive attacks than Nakamoto consensus, e.g., attacks have a smaller profitability threshold, and they result in higher relative rewards. Next, we introduce a semi-parallel PoW protocol that outperforms both Nakamoto consensus and the existing parallel PoW protocols from most practical perspectives such as communication overheads, throughput, transaction conflicts, incentive compatibility of the protocol, as well as a fair distribution of transaction fees among the voters and the leaders. We use state-of-the-art analysis to evaluate the consistency of the protocol and consider Markov decision process (MDP) models to substantiate our claims about the resilience of our protocol against incentive attacks.

Keywords: Proof-of-Work (PoW); parallel PoW; blockchain consensus; incentive compatibility; incentive attacks; consensus safety guarantees

1. Introduction

Nakamoto consensus [1], a solution to the decentralized consensus problem, has a vulnerability to withholding attacks such as double-spending [2] and selfish mining [3] due to the fact that the arrival of the blocks in the longest chain protocol is subject to the Proof-of-Work (PoW) scheme, which has an exponential distribution. Since an exponential distribution with mean $\frac{1}{\lambda}$ has variance $\frac{1}{\lambda^2}$, adversaries can take advantage of the high variance of the block interarrival times by deviating from the honest behavior. More specifically, whenever it takes longer than usual for honest miners to mine a new block, the adversary can make profits from this by creating blocks and keeping them private until some predetermined conditions on the longest public chain are met [2–4].

The problem of high variance is not only pertinent to the withholding attacks. Mainstream cryptosystems which rely on Nakamoto consensus with high block interarrival times such as Bitcoin, which produces a block per 10 min, distribute the coinbase reward to the miner who finds a proper nonce. The other miners who could not find a proper nonce are not rewarded and have to start mining the next block to get a chance for the next reward. Although the reward a miner gets is proportional to its hashrate on average, the total number of blocks a miner has to work on until he/she gets a reward is distributed geometrically with a parameter $p = \frac{\text{miners hashrate}}{\text{total hashrate}}$ and has both a high mean $\frac{1}{p}$ and a high

Check for updates

Academic Editor: Josef Pieprzyk

Received: 11 August 2026
Revised: 28 September 2026
Accepted: 27 October 2026
Published: 31 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

variance $\frac{1-p}{p^2}$ for a miner who controls a small fraction of the hashpower in the system. Hence, it is no surprise that a home miner might have to spend decades before getting a reward in the Bitcoin system [5]. The presence of network delays and consequent forks exacerbates the problem of unfair reward distribution since blocks belonging to the miners with small fractions of hashpower are less likely to be included in the chain in the long run if they are in a fork race with other blocks [6–8]. As a result, miners who control small fractions of hashpower join their forces together and create pools, which distribute the block rewards among their members regularly, in order to provide a stable income. However, the formation of pools is not only a direct threat to the decentralization but also creates vulnerabilities such as different forms of block withholding attacks [9–12]. Moreover, as a pool size grows, the system becomes more vulnerable to selfish mining attacks from that pool. Since the throughput in Nakamoto consensus grows with the arrival of the blocks, it is also subject to limitations that originate from the same core principles.

Many protocols have been suggested to solve some of the above problems of Nakamoto consensus. By modifying how transactions are appended to the ledger, Fruitchain [13] aims to increase the throughput level and achieve a more fair distribution of transaction rewards among miners with less variance. Prism [14] tries to increase the throughput and reduce the cost of scalability and maintenance for miners through a voting mechanism that separates the transaction ledger and consensus votes. Bitcoin-NG [15] increases the throughput and reduces the variance of appending transactions to the ledger with a PoW leader election mechanism. The GHOST rule [16] allows miners of the orphan blocks to get partial rewards by replacing the longest chain rule and mitigates the unfair reward distribution. Modifications to the fork choice rule of Nakamoto consensus such as [3,17–19] try to mitigate the incentive attacks on the reward mechanism. PHANTOM [20] and SPECTRE [21] protocols allow blocks to have multiple parents creating Directed Acyclic Graphs (DAGs). As a result, higher block generation rates, λ, are allowed which in turn creates a fairer blockchain environment with high throughput. Multistage PoW [22] divides the PoW problem for each block into multiple sequential stages which decreases the variance of block interarrival times. Many of these protocols also come with limitations such as low fault tolerance, absence of rigorous security violation analysis, absence of an efficient, global and consistent ordering of transactions, communication overheads, and lack of resilience against possible withholding attacks [20,21,23–26].

Parallel PoW [27] has a similar principle where the system as a whole needs to solve L puzzles in parallel, which decreases the variance of reward distribution times and increases resilience against double-spending by decreasing the variance of block interarrival times. However, in their raw form, parallel PoW protocols are more vulnerable to incentive attacks [28–30]. Bobtail [28] can be seen as a modification to the raw form of the parallel PoW scheme to increase the resilience against proof withholding attacks. Tailstorm and DAG-style voting schemes are other modifications introduced in [29,30] that change the reward distribution rule between the parallel proofs which improves the resistance against the withholding attacks observed in the raw form of the parallel PoW scheme. It is worth noting that parallel PoW protocols where miners create parallel proofs on top of the same block are distinct from sharding techniques where miners create and maintain parallel chains. In this paper, we only consider parallel PoW protocols.

We first consider the existing variations of the parallel PoW protocols [28–30] and develop hard-coded incentive attacks that show shortcomings of the protocols. For Bobtail [28], drawing an analogy to the analysis of Zhang and Preneel [19] about smallest-hash-based fork choice rules, we combine incentive attacks with a Denial-of-Service attack and argue that the resulting profitability threshold, α_T, i.e., the minimum fraction of hashpower needed for a profitable attack, is zero. For Tailstorm [29], we develop another hard-

coded proof withholding attack which results in a profitability threshold of $\alpha_T \leq \frac{1}{L}$ and results in revenues as large as the upper bound of the profit of a selfish miner in Nakamoto consensus. For DAG-style voting [30], even when the adversary has no network influence, i.e., $\gamma = 0$, contrary to the authors' intention, when the adversarial fraction of hashpower $\alpha > \frac{1}{3}$, the punishment mechanism favors the adversary for a hard-coded attack we develop, and the resilience is worse than the resilience in Tailstorm in this parameter regime. For all three schemes, we verify our theoretical analysis and arguments with simulations.

Next, we introduce a voting-based semi-parallel PoW protocol that outperforms the existing models from most practical perspectives. Existing works on parallel PoW not only lack resilience against incentive attacks but also have communication overhead problems. Moreover, an arbitrary increase in throughput without conflicts among transactions is not possible. Our protocol offers a solution for all these problems where each PoW solution (proof) points to a proposal ledger through its Merkle root, to be voted in the next block. The communication overhead is decreased by requiring the proof to keep the proposal ledger hidden but specify the amount of transaction fees offered.

We introduce several rules for honest miners to make sure all miners of size $\alpha < 0.5$ pick the proposal ledger that offers the most fees as L increases as well as a fair distribution of transaction fees among the proposer and the subsequent miners that pick the proposed ledger. The introduction of the ledger choice rule and the split rule for the distribution of transaction fees makes sure that arbitrary manipulations of ledgers are not feasible for adversaries. As a result, only one hidden ledger is shared, and communication overheads as well as transaction conflicts are avoided. We introduce punishment mechanisms as in [17–19] to further reinforce incentive compatibility of the protocol. We adapt the state-of-the-art security latency analysis of Nakamoto consensus [31,32] to analyze the double-spending probability for our protocol. Further, we consider Markov decision process (MDP) models to substantiate our claims about the resilience of our protocol against incentive attacks.

In the rest of this paper, we assume a single colluding adversarial entity which holds α fraction of the hashpower in the system. We assume a network of n honest miners connected to each other with some network topology and large n. We denote the network influence of the adversary, i.e., the fraction of honest miners that prefer the adversarial chain in case of a fork, as γ. We abstract the mining puzzles with a collision-free hash function, f_H, which acts as a random oracle. In Section 2, we summarize the existing variations of the parallel PoW protocols and show their vulnerabilities against incentive attacks. In Section 3, we formally introduce our protocol and subsequently analyze it in Section 4. In Section 5, we conclude this paper by discussing how our protocol relates to the existing work on PoW-based protocols in general, as well as its limitations and possible future work.

2. Incentive Attacks on Existing Variations of Parallel PoW Protocols

In this section, we consider proof withholding attacks on the existing variations of parallel PoW protocols, where we pick the relative rewards as the relevant metric which is the fraction of the rewards an agent has in the system, as is commonly used in most of the literature. These attacks result in relative rewards higher than the honest mining for many parameter regimes and show that existing variations of parallel PoW protocols are even less resilient against incentive attacks than Nakamoto consensus in many realistic settings.

We start by defining the raw form of the parallel PoW protocol. In the raw form of the parallel PoW protocol, a block consists of L PoW solutions (simply referred to as proofs) that build upon the previous block by pointing to its hash. Each proof contains semantic information such as the previous hash, Merkle root of a ledger, the reward address, etc., which we ignore. From a practical point of view, we only care for the hash value of the proof

and the pointer to the previous hash. In the raw form, the proofs refer to the previous block, and the hash values of these proofs have to be below a certain threshold. All proofs that confirm the same parent block are eligible to be included in the next block, and L of them are needed to create the next block. As a result, given the most recent block, the proofs of the next block can be generated in parallel. This in turn allows a full cooperation between miners without the need for a pool.

In the honest protocol, essentially everyone works on top of the most recent block, and miners immediately release the proof they mine, which is essentially a vote (or a validation) on the previous block. In case of a fork, the miners pick the chain with the most aggregate votes to build upon. Thus, the first time L new proofs are mined on top of a previous block, a new block can be created, and revenues can be shared between L proofs. However, this usually requires an election rule to decide who will combine the proofs to create the new block. A simple way to do that is to let the miner of the proof with the lowest hash value be the leader and create the next block. For the purpose of the analysis we perform in this section, we believe that the details provided here are sufficient to build upon. For further details on the raw form of parallel PoW, we refer the reader to [27].

If the adversary does not release its proofs for the current block immediately as they are mined, while being able to observe the honest proofs immediately as they are mined, this informational advantage can potentially result in extra profits. In the context of parallel PoW protocols, these types of attacks are called proof withholding attacks. Next, we give an overview of the existing variations of parallel PoW protocols [28–30] and develop hard-coded proof withholding attacks that result in more relative rewards than the honest mining behavior.

2.1. Bobtail

In Bobtail, each proof has a hash value, and proofs building upon the same previous block can be created in parallel as usual, and any proof with a hash value lower than a certain threshold is a valid proof (in the actual paper of [28], the average hash value of L proofs needs to be below a threshold; however, for practical purposes and simplicity, we ignore proofs above a certain threshold and assume any proof below it is a valid proof as in [33]). The proof that has the lowest hash value is allowed to combine the current L proofs to create a block. To prevent the proof withholding attacks, each proof has a support value, which is the main difference from the raw form of the parallel PoW protocol. This *support value* is essentially a pointer to the lowest hash valued proof it has seen so far among the proofs for the current block height. The proof with the lowest hash value combines L proofs that have higher hash and support hash value to create a block. We note that [27] states that Bobtail is not a parallel PoW protocol. However, as both protocols allow parallel mining of the proofs and the essential difference between the raw form of the parallel PoW protocol and Bobtail is the support value that proofs in Bobtail contain, in this paper, we treat Bobtail as a variation of parallel PoW protocols.

In Figure 1, we show an example of how proofs point to each other according to their mining times and hash values. Each proof, denoted by p_i, has a hash value aligned with the y axis, a mining time aligned with the x axis, and a pointer to its support hash. For example, since p_5 has the smallest-hash value among all, if $L = 7$, then the miner of p_5 is allowed to create a block with references to all p_i. On the other hand, the miner of p_2 cannot use any of the red proofs to create a block since they have smaller hash values of their own or smaller support value.

https://doi.org/10.3390/cryptography10060080

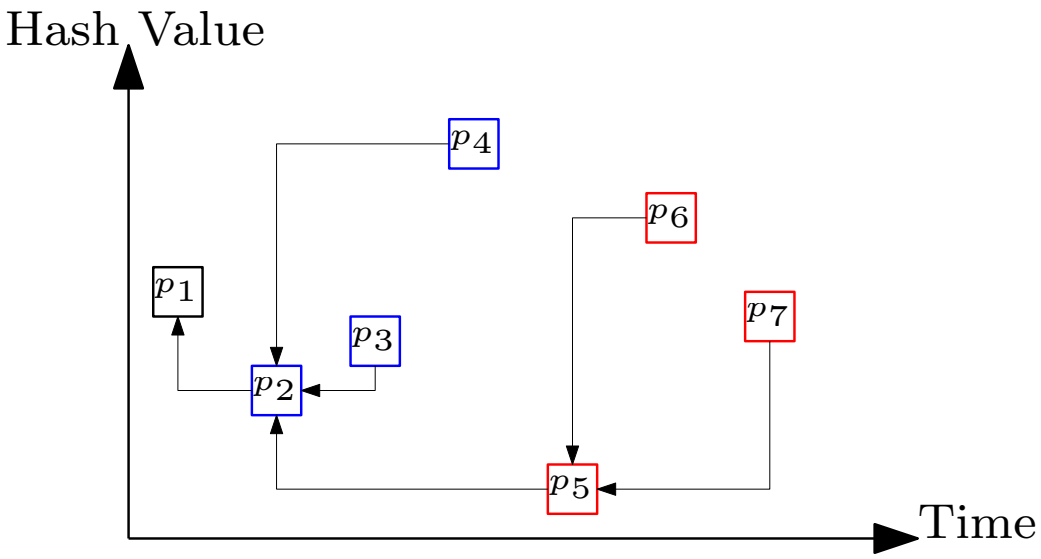


Figure 1. Bobtail protocol proof model.

In the raw form of parallel PoW, i.e., without the support value rule, the adversary cannot use the honest proofs that have lower hash value than its own lowest hash value while creating a block. Hence, the adversary can mine but hide all its proofs until it can form a proof set of size L containing all of its own proofs and all honest proofs that have higher hash value than its own proofs. With the support value rule in place, this proof withholding attack is mitigated, since in addition to the honest proofs that have lower hash value than the adversary's own lowest hash value, all the proofs that refer to them are not eligible either. For example, in Figure 1, if p_2 is mined by the adversary and it tries to create a block with p_2 having the lowest hash, it cannot use p_5, p_6, or p_7 when creating a block even though p_6 and p_7 have higher hash value than p_2 since p_2 is the support value of both p_6 and p_7.

To further strengthen the protocol, the authors consider a bonus reward rule where proofs whose support proof is the lowest hash valued proof which eventually created the final block get a bonus reward. The protocol also defines further incentive rules about prioritizing inclusion of proofs in case multiple sets of L proofs are possible; see [28] for further details. In Appendix A, we first provide details of the two attacks considered in the original paper of Bobtail and later create a stronger hard-coded attack for which pseudo-code is given. We provide both a theoretical overview as well as a simulation result which show Bobtail simply has a profitability threshold of $\alpha_T = 0$ due to the informational advantage that the adversary possesses. As a result, even a small home miner can mount an incentive attack on the protocol. It is also noted that the attack scales block interval times, decreases the throughput, and increases the latency.

2.2. Tree- and DAG-Style Voting

Tailstorm [29] is a tree-style parallel PoW protocol [30]. Tailstorm introduces two main innovations compared to the raw form of the parallel PoW protocol. First, even though the proofs for the current block height are generated in parallel, they can refer to each other, creating a tree of proofs for the current block. As long as the root of the tree is the same previous block, they are all eligible to be combined together to create the next block. Second, Tailstorm deviates from the raw form in how it rewards the proofs. Each block is rewarded proportional to the maximal depth of its proof tree, and the reward is distributed equally to each proof in the tree. As every proof in the same block gets rewarded equally, this means

they are punished equally for parallelness, since they can get more rewards if they build their proofs in a sequential chain, whereas parallel proofs decrease the rewards equally.

The addition of this tree structure and the new reward mechanism aims to mitigate the proof withholding attacks. DAG-style voting [30] introduces modifications along the same lines that aim to fix the unfair punishment mechanism in Tailstorm. Namely, proofs are allowed to refer to multiple proofs, which results in a DAG. In this case, the reward of each proof is proportional to the total number of ancestor and descendant proofs it has within the block, which punishes the non-sequentiality more fairly; see [30] for further details.

Both the authors of [29,30] consider proof withholding attacks with a normalized reward metric which is the reward of an attacker up to the current block divided by the progress of the chain. They show the resilience of their algorithm according to this normalized reward metric; however, their normalization is only partial and does not take the hashpowers into account. In Appendix B, we reconsider normalization with respect to both chain progress and the hashpower of entities together, i.e., we focus on the reward per chain progress per hashpower, which in turn shows that the relative reward metric is the better choice when analyzing incentives in PoW protocols.

In Appendix C, we handcraft proof withholding attacks to show vulnerabilities of tree- and DAG-style voting protocols, where we consider the relative reward metric. The attacks we consider are quite similar for both tree- and DAG-style voting protocols, where the attacker simply withholds its own proofs which results in an informational advantage. This informational advantage simply results in a direct revenue advantage in Tailstorm, whereas the advantage in DAG-style voting protocol requires the adversarial power to be above some threshold. We explain the attack, analyze it theoretically, and then give pseudo-code for a practical implementation. To further strengthen our analysis, we simulate the attack to verify our theoretical results.

With the hard-coded proof withholding attack, the fraction of the adversarial proofs in the public chain reaches $\frac{\alpha}{1-\alpha}\frac{L-1}{L}$ even when $\gamma = 0$, for both tree- and DAG-style voting. Since this fraction directly equals the relative reward ratio in Tailstorm, an upper bound on the profitability threshold, i.e., an upper bound on the minimum adversarial power needed for a profitable attack, is $\alpha_T \leq \frac{1}{L}$. In other words, if the adversary holds more than $\frac{1}{L}$ fraction of the hashpower, then the attack is always profitable. For less adversarial power, better attack strategies may still exist. Although parallel PoW protocols become more secure against double-spending attacks as L increases and the income of the miners becomes more stable since even the small home miners [5] can get a reward as the nonce difficulties are easier, as the above attack implies, Tailstorm becomes more vulnerable to the incentive attacks as L increases. For DAG-style voting, the proof withholding attack is still in place; however, the profitability threshold is different since proofs get rewards proportional to the number of ancestors and descendants (see Appendix C.1).

2.3. Numerical Results for the Incentive Attacks

We plot the relative reward ratio of the attacks for the simulation of Bobtail and the theoretical results of Tailstorm [29] and DAG-style protocol [30] in Figure 2, where we pick $L = 50$ and $\gamma = 0$. Note that the choice of $\gamma = 0$ fully favors honest miners from the perspective of forks. For reference, we include the honest mining curve, i.e., when the adversary mines honestly and gets α rewards and the upper bound of selfish mining in Bitcoin, or more generally in Nakamoto consensus, where $\gamma = 1$, which is equal to $\frac{\alpha}{1-\alpha}$ (worst-case scenario for honest miners in Nakamoto consensus). We truncate the curves before they rise above the honest mining behavior for the sake of neatness.

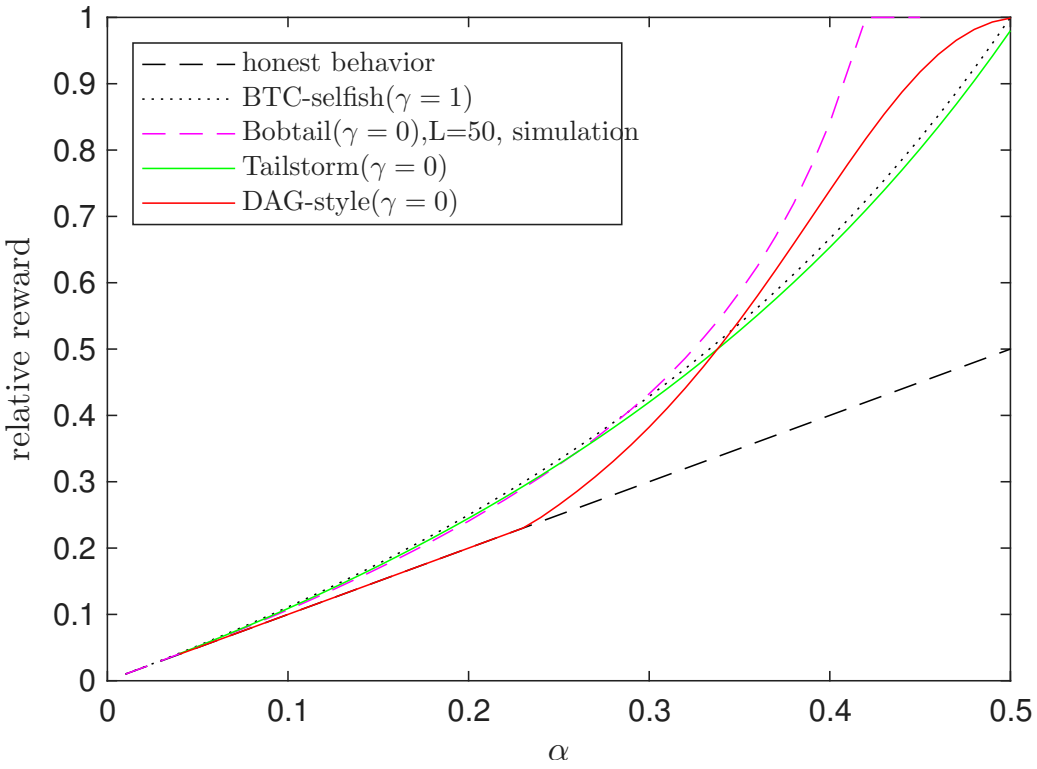


Figure 2. Relative revenues of withholding attacks on existing variations of parallel PoW protocols.

Bobtail has a profitability threshold of $\alpha_T = 0$, even though it is hard to discern the curves for small α in the figure. When $\alpha \geq \frac{1}{3}$, Bobtail gives even more profits than the upper bound of the selfish mining in Nakamoto consensus. We note that we added a prevention mechanism to our simulation of Bobtail that makes sure honest miners do not wait too long to find a small hash value as explained in Appendix A. Despite this, the hard-coded attack is stronger than the other attacks in the other protocols.

It is clear that for $\alpha > \frac{1}{L}$, the attack on Tailstorm is more profitable than honest mining as everyone including the honest miners is punished equally, and the limit as $\alpha \to 0.5$ is $\frac{L-1}{L}$. The behavior of the curve follows the upper bound of the selfish mining curve of Nakamoto consensus.

For DAG-style voting, the introduction of reward scaling according to the depth of the graph makes the protocol more resilient against incentive attacks than the Tailstorm only for small α and the profitability threshold in DAG-style voting is larger than Tailstorm. However, even when $\gamma = 0$, the DAG version of the attack becomes more profitable than the same withholding attack in Tailstorm for $\alpha > \frac{1}{3}$. This is due to the fact that the rewards in DAG-style voting are scaled according to the ancestor and descendant counts and the adversarial proofs have more ancestors and descendants than the honest ones have when $\frac{\alpha}{1-\alpha} > 0.5$. This is contrary to the intention of the authors as they introduced reward scaling to punish adversarial actions since they hold less than 0.5 fractional power.

It is worth noting that we did not perform an exhaustive search on all possible attacks and provided only a hard-coded attack. For example, the handcrafted attacks we consider work irrespective of the γ value (since we consider $\gamma = 0$, i.e., best-case scenario for honest miners); hence, better attacks exist that take advantage of γ. Further, attacks that better exploit the temporal variability of mining times may be possible. Hence, the actual profitability threshold might be even lower. However, our goal in this section is to show the existence of at least one type of attack against each parallel PoW protocol that proves their inferior resilience compared to the resilience of Nakamoto consensus. Hence, the attacks

and their analysis in Appendices A and C as well as the results presented here are sufficient for our purposes and the scope of this paper.

3. Proposed Protocol Description

The analysis and findings in the previous section show that full parallelization in PoW protocols results in vulnerability to incentive attacks due to the informational advantage of the adversary over the honest miners. Hence, we start by questioning the extent of parallelness needed in a parallel PoW protocol as well as the efficiency of creating transaction ledgers associated with the proofs.

If we think of each proof as a small block with some transactions attached to it as in [29,30], the propagation times associated with the ledgers of proofs would imply that more proofs are created during the network delays. Hence, proofs that are mined at around the same time should be valid and compatible to be combined together in such a protocol. But we have proven that the parallel PoW protocols in [29,30] are vulnerable to withholding attacks. Moreover, the protocol usually would have throughput loss due to the conflicts in ledgers of the proofs mined in parallel.

On the other hand, a protocol like Bobtail [28] separates the proofs from the ledgers and offers a leader election mechanism. This approach decreases communication overhead and solves the problem of conflicts between ledgers. However, it is well known that smallest-hash-based mechanisms used in Bobtail allow attackers to have an informational advantage which in turn can be converted into profits with withholding attacks as we have shown in the previous section. Further, smallest-hash proof withholding attacks also increase block interarrival times.

Notice that if a protocol separates the proofs from the ledgers, it decreases propagation times of proofs. As a result, fewer proofs will be mined during delay periods and thus the level of parallelness does not need to be high. This, however, requires a ledger election mechanism that is incentive-compatible. Further, in this case, ledger conflicts can be omitted with a proper mechanism. In this section, we introduce such a protocol that allows a reasonable amount of parallelness with small proof propagation times, an incentive-compatible leader election mechanism, less communication overhead, and no ledger conflicts. Further, it has superior variance properties compared to Nakamoto consensus, such as better safety guarantees and increased throughput.

3.1. High-Level Overview of the Protocol

For the sake of clarity, we start by explaining a high-level overview of the protocol with the help of Figure 3 before diving into the detailed mechanics in later subsections. We consider a series of blocks chained together as depicted in Figure 3. A valid block at height h, represented with red boxes in the figure, consists of L or $L+1$ valid PoW solutions (proofs) clustered together. Proofs are divided into two categories: initiators and incrementals. The orange circles represent the initiators whereas other circles are incrementals in the figure. Initiator proofs are the first proofs mined for a block whereas incrementals are the follow-up proofs mined on top of the initiator proofs and other incrementals.

Each proof contains a Merkle root pointer to a ledger it proposes but keeps that ledger hidden and only discloses the amount of transaction fees associated with it. As a result, proofs are virtually separated from the ledgers; thus, the level of parallelness does not need to be high. Hence, the level of parallelness we allow in this protocol is at most two per proof height, i.e., two proofs mined in parallel are compatible in the sense that a subsequent proof in the next proof height can refer to the two if they both agree on the history of proofs preceding them. In this way, the proofs are mined in a semi-parallel/semi-sequential manner until the block h contains L (or $L+1$) valid and compatible proofs, at which point

the ledger associated with the proof that offers most fees is released and verified, which we simply call the winning ledger, represented with the blue boxes in the figure. The next block height starts with an initiator proof (or two if they agree on the history), which points to the Merkle root of the winning ledger of the previous height (represented as the blue boxes). Subsequently, incremental proofs are mined as described earlier and the chain grows.

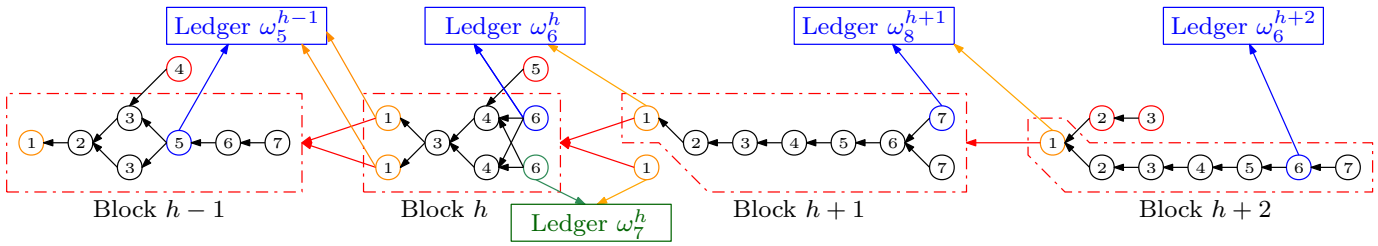


Figure 3. The proof model of the semi-parallel PoW protocol with $L = 7$. Orange and black circles represent the initiator and incremental proofs, and the numbers inside the circles represent the incremental heights. Each initiator proof refers to a ledger from the previous block height as the winning ledger. Blue boxes represent the winning ledgers based on the most aggregate work fork choice rule. Here, if the proof 7 of block $h+2$ is the most recent proof with most aggregate work and $b = 1$, the ledgers until block height $h+1$ are confirmed, since they are part of the most aggregate work chain. But ledger ω_6^{h+2} is not $b = 1$-confirmed since 1-confirmation requires a 1-complete block on height $h+3$, where the initiator of block $h+3$ needs to refer to ω_6^{h+2} as the winning ledger. Red dotted boxes (e.g., block h) together with a valid ledger (e.g., ω_6^h) are considered as a complete block. Note that block h with ledger ω_7^h is a complete but orphan block. To replace the block h and its associated ledger ω_6^h with the block h and its associated ledger ω_7^h in the view of honest miners, the adversary creates an initiator proof that refers to ω_7^h. The adversary has to create further proofs and share complete blocks at block height $h+1$ and $h+2$ to make the block h and its associated ledger ω_7^h as the fork choice rule (most aggregate work). Note that the adversary cannot use any proofs of block $h+1$ or $h+2$ in the figure since initiator proof of the block $h+1$ refers to ω_6^h as the winning ledger and all the subsequent proofs build on top of this initiator proof. Thus, to rewrite the history, the adversary has to work alone once it falls behind in the race of most aggregate work.

3.2. Proof Structure

Next, we explain the technical details regarding the content of the proofs, which are represented as ω_i^h, $i = 1, \dots, (L \text{ or } L+1)$ for a block height h. A valid initiator proof consists of six parts concatenated together, i.e., $\omega_1^h = (\omega_{1,1}^h, \dots, \omega_{1,6}^h)$, where, w.l.o.g., we assume the miner who mines ω_1^h is the miner $i = 1$. The sixth part of the proof, $\omega_{1,6}^h$, contains the nonce to make sure $f_H(\omega_1^h) < T_L$. The fifth part, i.e., $\omega_{1,5}^h$, is simply the hash of the coin address of the miner $i = 1$ to store its mining rewards. The fourth part, $\omega_{1,4}^h$, is the Merkle root of the transaction ledger proposed by miner 1, which will be explained in detail later. The third part, $\omega_{1,3}^h$, is simply the total accumulated transaction fees of the proposed ledger associated with $\omega_{1,4}^h$. The second part is a summary of (w.l.o.g. assume L) clustered proofs of preceding block height, i.e., $\omega_{1,2}^h = \omega^{h-1} = (f_H(\omega_1^{h-1}), \dots, f_H(\omega_L^{h-1}))$. If two initiator proofs at height h exist, say miner 1 and miner 2 created valid initiator proofs ω_1^h and ω_2^h, then they need to agree about the history to be clustered together in a block in order to prevent double-spending, i.e., $\omega_{1,2}^h = \omega_{2,2}^h$ ($\omega_{1,1}^h = \omega_{2,1}^h$ should also be satisfied, which will be explained in Section 3.3).

A valid incremental proof created by miner i consists of six parts concatenated together, i.e., $\omega_i^h = (\eta_{i,1}^h, \dots, \eta_{i,6}^h)$. Similar to the initiator proofs, the last four parts of the incremental proofs, i.e., $\eta_{i,3}^h$ to $\eta_{i,6}^h$, have the same rules and serve the same purpose. Thus, in this paper, we simply refer to these parts with ω instead of η for the sake of simplicity. However, the first part of the proof is simply a number representing the incremental work within the

cluster of proofs at block height h; hence, we call it the incremental height of the proof. The second part of the proof, related to that, is a list of summaries of proofs of the previous incremental height, with the condition that the list can contain at most two proofs and these proofs agree about the history, be it the ledger or the incremental proofs. An initiator has incremental height 1 by default, and all incrementals inherit $\omega^h_{1,1}$ and $\omega^h_{1,2}$ of the initiators within the cluster.

Example from Figure 3

As an example, consider the block at height h in Figure 3 with $L = 7$. Two valid initiator proofs are represented as orange circles created by miner 1 and 2, with $\omega^h_{1,2} = \omega^h_{2,2}$ and $\omega^h_{1,1} = \omega^h_{2,1}$, which were possibly created approximately at the same time, hence were not aware of each other, where we order them with respect to their summary, i.e., $f_H(\omega^h_1) < f_H(\omega^h_2)$.

In this figure, the numbers inside the circles represent the incremental heights of the proofs. The next incremental, let us call its creator miner 3, who is honest and was aware of both of the initiator proofs at the time of the mining, will have $\eta^h_{3,1} = 3$, $\eta^h_{3,2} = (f_H(\omega^h_1), f_H(\omega^h_2))$. Here, we note that although the proof height of ω^h_3, i.e., the height of the proof within block height h, is 2, the incremental height is 3 since the incremental work is 3.

Next, assume two more incremental proofs are created by miners 4 and 5, such that $\eta^h_{i,1} = 4$ and $\eta^h_{i,2} = f_H(\omega^h_3)$ for $i = 4, 5$. Since these two proofs agree about the history, the next miner who is aware of both, call miner 6, can combine them as well, i.e., $\eta^h_{6,1} = 6$, $\eta^h_{6,2} = (f_H(\omega^h_4), f_H(\omega^h_5))$, where they are ordered according to $f_H(\omega^h_4) < f_H(\omega^h_5)$. In a similar way any other proof i, say $i = 7$, which has $\eta^h_{i,1} = 6$, $\eta^h_{i,2} = (f_H(\omega^h_4), f_H(\omega^h_5))$ can be combined with the proof of miner 6 on the next incremental height. Note that ω^h_6 and ω^h_7 are represented with the blue and the green circle on block height h in the figure, respectively. On the other hand, a miner 5′, possibly an adversary or an honest miner who was aware of the proof ω^h_4 but not aware of the proof ω^h_5, may have created the proof $\omega^h_{5'}$ with $\eta^h_{5',1} = 5$, with $\eta^h_{5',2} = f_H(\omega^h_4)$ represented as a red circle. In this case, we do not allow the proof of the miner 5′ to be combined with the proof of the miner 6 and the next honest miner will prefer combining the proofs of 6 and 7 since they agree about the history and can be combined to result in more aggregate work. Thus, the proof of the miner 5′ becomes orphan and is not included in $\omega^{h+1}_{1,2}$.

In a similar manner, at block height $h - 1$, the proof represented with a red circle cannot be combined with the proof represented with a blue circle, since their parent sets differ. Regarding the red circles of block height $h + 2$, multiple scenarios are possible. For example, an adversary may have kept both private until two parallel honest proofs are generated. In this scenario, the red and black proofs with incremental height 3 cannot be combined since they have different parents, and the fork resolves later on according to the fork choice rule described in Section 3.7.

To summarize this rule, the two incremental proofs at incremental height i who have the same parents can be combined in the next proof height resulting in incremental height $i + 2$. This also explains why $L + 1$ PoW solutions for a block as in block $h + 1$ in Figure 3 are allowed, as the last incremental height can have two proofs which can be combined by the initiator of next block height. We call two proofs that can be combined *parallel proofs*.

3.3. Ledger Election

It remains to define $\omega^h_{i,1}$, i.e., the first part of an initiator PoW solution. Before doing so, we make some observations. It should be clear that for height h, each miner has a proposed ledger that does not necessarily agree with others. However, miners of the same cluster

agree with each other about the previous block since they either build on top of each other or $\omega^h_{i,2} = \omega^h_{j,2}$ for two initiator proofs of the same cluster.

Thus, it remains to pick a rule to decide about the ledgers. If we are to accept all the transactions of the proposed ledgers of a clustered proof at height $h-1$, we need to further decide about a rule in case of conflicts, which can be as simple as first ordering by incremental height within the cluster and accepting the ledgers in the order of their hashes (or Merkle roots) in order to ignore further conflicts. However, this will both increase the communication overhead and also create a necessity of a further rule in case a ledger is not shared by the miner, e.g., a ledger withholding attack by the adversary.

From $\omega^h_{i,4}$, it should be clear that within the cluster each miner has a proposed ledger, and the ledger pays $\omega^h_{i,3}$ in transaction fees in addition to the coinbase reward. If we only accept one ledger, we need each proof of a cluster at a height to agree with each other about the ledger of the previous block height, in order to prevent rewriting of the history, i.e., double-spending. Thus, we consider the following scenario: The transaction fees associated with $\omega^h_{i,4}$ are offered to be shared with the miners of the proofs for the next block height $h+1$. If this is the case, it is in the best interest of miners at any block height to pick the ledger of the previous height that pays the most fees.

Thus, for an initiator proof ω^h_1, the rule is to simply pick the ledger that pays the most among the valid ledgers of a cluster from the previous height, i.e., $\omega^h_{1,1} = \omega^{h-1}_{j^*,4}$, where $j^* = \arg\max_j\{v(\omega^{h-1}_{j,3})\}$. Here, $v(\omega^{h-1}_{j,3}) = 0$ if $\omega^{h-1}_{j,4}$ is not valid or withheld; otherwise, if v is simply the identity function. If parallel initiator proofs exist in the same cluster at block height h, i.e., $\omega^h_{1,2} = \omega^h_{2,2}$, then we require them to agree about their ledger pick for the previous height as well, i.e., $\omega^h_{1,1} = \omega^h_{2,1}$. As all incrementals build on top of the initiators, they necessarily inherit the pick of the initiators; hence, we have a consensus about the ledger of the previous block height among the proofs clustered together.

Example from Figure 3

In Figure 3, the proofs that offer the most fees are represented with blue circles which are pointed to by their creator as well as the initiator proof of the next block height. Further, notice that both initiator proofs at block h pick the ledger of the proof ω^{h-1}_5; hence, they can be combined. On the other hand, even though the initiator proofs at block height $h+1$ agree about the proofs in the previous block h, they pick different ledgers from the previous block; thus, they cannot be combined. Further, any incremental proof and blocks mined on top of one of these incremental proofs cannot be used by the other incremental proof since they inherit the ledger choice of the incremental proof. Thus, to rewrite the history, the adversary has to create a separate proof tree with more aggregate work.

3.4. Communication Overhead

As a valid PoW solution contains the amount of the fees the associated proposal ledger offers, i.e., $\max_j\{\omega^{h-1}_{j,3}\}$, there is an open competition about the leader (ledger) of a cluster, i.e., the miner whose proposed transaction ledger is going to be accepted. Thus, the leader of a cluster at block height $h-1$ is essentially decided by the winning cluster of the next block height h. Miners hide their proposed ledger until there are L proofs accumulated in the cluster in order to prevent others from replicating their proposal (if it pays a high fee). This in turn decreases the communication overhead associated with parallel PoW protocols, since honest miners only need to share their ledger if they pay the most fees and only when there are L proofs accumulated.

As an incremental proof contains at most 7×32-bytes, we can assume that it will be received fast and will not congest the network. Although initiator proofs contain a bit more,

i.e., at most $(L+6) \times 32$-bytes, they are much less frequent and are thus unlikely to have a considerable impact on the network congestion.

3.5. Throughput Increase

We assume that the ledger size is limited; however, to increase throughput, the maximum allowed size of the ledger associated with a proof of a miner i is proportional to the height of the proof within the cluster, which we call proof height. More specifically, the proposed ledger size of the PoW solution ω_i^h is limited by $\lceil \frac{C(\omega_i^h)}{M} \rceil \times B$-bytes, where $C(\omega_i^h)$ is the proof height. Note that $C(\omega_i^h)$ is different from $\eta_{i,1}^h$, as $\eta_{i,1}^h$ is the incremental height.

For example, if an incremental proof has two initiator proofs as its parent, then it has incremental height $\eta_{i,1}^h = 3$, but $C(\omega_i^h) = 2$. Thus, there are at least $\frac{L}{2M}$ and at most $\frac{L}{M}$ quantized throughput levels. As a result, miners are incentivized to share their proof as soon as they find a valid one to minimize the number of parallel proofs since it allows the cluster as a whole to increase the number of transactions that can be included in a proposal ledger.

Further, the proposed ledger is created by a single miner which makes sure that no conflict can occur, i.e., unlike the other parallel PoW protocols, throughput is not wasted. This rule also reinforces a decreased communication overhead as it is unlikely for a ledger of a proof with small $C(\omega_i^h)$ value to offer a high fee, $\omega_{i,3}^h$; hence, miners simply focus on creating proofs initially until high throughput levels are achieved, which is when they start to focus on creating candidate ledgers for the next block height.

3.6. Ledger Content

A valid ledger created at block height h needs to satisfy certain conditions related to the mining rewards and fees. As the winning ledger of a cluster at block height $h-1$ and other miners within that cluster are pointed to by the miners of height h, the coinbase rewards of height $h-1$ are distributed in the ledger of block height h. Hence, a ledger at block height h needs to contain the coinbase reward distribution of height $h-1$ in order to be considered as valid.

Further, as the transaction fees of the winning ledger of height $h-2$ are shared between the miner who created that ledger and miners of winning cluster at block height $h-1$, which are pointed to by miners of height h, those fee payments have to be finalized in the ledger of height h as well, which may require a smart contract algorithm involving the associated ledgers and the miners' addresses of the previous two heights. The fraction of transaction fees that will be paid to the ledger creator is denoted as r and is explained in Section 4.3 and analyzed in Appendix G from the perspective of incentive attacks.

In short, one can think of each proof on block height h as a vote on the proposed ledgers of height $h-1$. After a while as the proofs at height h are accumulated enough, the miners also start to focus on proposing ledger candidates for the next block height which come together with the proof. Each proof gets rewarded for their work through the mining rewards and transaction fees.

3.7. Difficulty Adjustment, Fork Choice, and Block Confirmation Rules

For difficulty adjustment, we simply assume that T_L is updated every two weeks to make sure that the expected number of blocks generated every two weeks, call it N, stays the same. Since each block has on average $\bar{L}$ PoW solutions with $L < \bar{L} < L+1$, this implies that the difficulty is adjusted such that, on average, $\bar{L} \times N$ proofs are generated every two weeks. Since the protocol requires a semi-parallel PoW, where at least $\frac{L}{2}$ proofs are sequential, the variance of block interarrival times decreases with $O(\frac{1}{L})$ compared to Nakamoto consensus [34].

At all times, the miners pick the chain that contains the most aggregate work (heaviest chain). The aggregate work of a proof is the total number of proofs, weighted by the respective nonce difficulties it has in its history, plus the work of a compatible parallel proof if it exists. Here, without loss of generality, we assume that all the weights are the same. For example, the aggregate work of an incremental proof ω_i^h is the total number of proofs until block height h plus the incremental height $\eta_{i,1}^h$ (plus one if there is another compatible proof ω_j^h with same history and $C(\omega_i^h) = C(\omega_j^h)$). We consider b-block confirmation rule where a ledger and the transactions inside are confirmed if the ledger is part of the heaviest chain and there are b-complete blocks mined on top of it.

Note that as the fork choice rule is the chain that contains the most aggregate work at all times, this implies that the ledger offering the most fees is only a temporary fork choice rule. More specifically, when a block with L proofs is created and the ledger offering the most fees is shared, honest miners try to mine an initiator proof for the next block height referring to this ledger. However, as soon as the next initiator proof is created, this proof contains the most aggregate work and is the fork choice rule no matter which ledger it refers to as long as that ledger is semantically valid. We consider such a situation in more detail in Appendix D and show that it does not create any splits in honest views.

3.8. Reward Distribution

The block at height h receives total mining rewards proportional to the total proof height, $C(\omega_L^h)$, which is distributed in the ledger of block at height $h+1$. Further, a proof i for block height h receives $\frac{1}{C(\omega_L^h)}$ of the total block reward if it is the sole proof at that proof height, i.e., $C(\omega_i^h) \neq C(\omega_j^h)$ for any $j \neq i$. To mitigate the effect of the selfish mining attacks, if there are two proofs which have the same proof height, i.e., $C(\omega_i^h) = C(\omega_j^h)$ for some miners i and j, then the two proofs split the reward of the proof height and each gets $\frac{1}{2C(\omega_L^h)}$ of the total block reward.

Note that each proof in a cluster has the same difficulty contribution to the chain although some might get half the reward for it (a punishment for parallelness). In total, the block at height h receives mining rewards proportional to the total proof height, $C(\omega_L^h)$, which is distributed between the miners of the block at height h. As a result, each proof height on any block height is rewarded the same.

Example from Figure 3

The block at height h in Figure 3 receives 4 units of reward, and each proof in this block gets $\frac{1}{2}$ units of reward except the proof with incremental height 3, which is the sole proof at proof height 2 and gets 1 unit of rewards. On the other hand, the block at height $h+2$ gets 7 units of reward and each proof gets 1 unit of rewards.

3.9. Adversarial Ledgers

To have a valid ledger, we need the ledger associated with $\omega_{i,1}^h$ of the initiator proof ω_i^h to be valid, i.e., the transaction ledger that an initiator proof accepts for the previous block height needs to be semantically correct with respect to the complete ledger history. Notice that even if the remaining proposed ledgers of block height $h-1$ are invalid, as long as the associated PoW solutions are valid, they can be used by an initiator proof of block height h as valid votes in the summary section, i.e., $\omega_{i,2}^h$. In other words, as long as $f_H(\omega_j^{h-1}) < T_L$, then ω_j^{h-1} can be used in $\omega_{i,2}^h$ even if the ledger of ω_j^{h-1} is not valid or withheld.

After the aggregate work at a cluster of block height $h-1$ reaches L proofs, the ledger of the proof that offers the most fees is expected to be shared with the network within a reasonable time (grace period) such as maximum network delays $\Delta + \delta$. Here, Δ is the maximum network delay when sending a transaction ledger of a block, whereas δ is the

maximum network delay of sending a proof. If the ledger that offers the most fees is not shared within the grace period, the ledger offering the second most fees (and the others if necessary) is shared. Moreover, in this case, the proof associated with the ledger that was not shared can still be included as a valid vote, and its aggregate work is counted towards the heaviest chain.

To decrease the effect of adversarial ledger not being shared in a grace period which slightly delays the transition to the next height, we could allow a bit more than $L + 1$ proofs, where the reward split for extra proofs could be omitted as well if the ledger offering the most fees is not shared. We leave these minor adjustments to specific protocol implementations. Note that, as it is common in the literature and the practice, we assume honest miners echo the messages they receive; hence, all honest miners eventually receive the adversarial ledger if one of them receives it.

4. Analysis of the Proposed Protocol

In this section, we first analyze the resilience against double-spending attacks. Then, we consider the proof withholding attacks and show the resilience using an MDP analysis. Finally, we consider the fairness of the transaction fee distribution rule $r = \epsilon$.

4.1. Double-Spending Attacks

For the purpose of analyzing the double-spending probability, here we make the same assumptions about the network model and the miners. Our assumptions resemble the state-of-the-art analysis of double-spending in bounded delay models for Nakamoto consensus [31,32,35–37]. As analysis of double-spending probability in the bounded delay models requires a good understanding of the possible adversarial attacks within the delay intervals, we assume sufficient familiarity with the models/attacks/approaches in the cited blockchain analysis papers. Here, we try to explain the analysis in as simple and concise terms as possible. Note that even though we do not directly prove the safety and the liveness properties under Δ-synchronous settings, we rigorously prove that the upper bound on double-spending decreases exponentially; hence, our analysis suggests that the two properties can be proven by further adopting the analysis of [35] to show the occurrence of Nakamoto blocks.

4.1.1. System Model

Assume that there are n honest miners (n is large) connected with an underlying network for communication, each holding an infinitesimal hashpower. Further, each proof generated by an honest miner experiences a network delay of $[0, \delta]$, whereas each block ledger experiences a network delay of $[0, \Delta]$. A single colluding adversarial miner is present in the system holding α fraction of the total hashpower. This adversarial entity experiences no communication delay and is aware of any proof as soon as it is generated. The adversary can delay the transmission of the honest proofs/ledgers by the maximally allowed δ/Δ for its own advantage, and the ties are broken in favor of the adversary.

4.1.2. Confirmation Rule

Here, we only consider the double-spending probability for parameters $L \geq 10$ and $b = 1$ and give an upper bound for it. To recap, our b-block confirmation rule states the following: a ledger together with the transactions inside is confirmed if it is the part of the heaviest chain and there are b-complete blocks mined on top of it. Notice that in Nakamoto consensus, a b-block confirmation rule also counts the block that contains the ledger; therefore, when we compare our results, we should take this difference into account. More formally, our $b = 1$ rule corresponds to $1 \leq b \leq 2$ in Nakamoto consensus, since the ledger is proposed in one block and gets voted in the next (even though it remains hidden in the former block).

https://doi.org/10.3390/cryptography10060080

We refer the reader to Appendix E for the description of the double-spending attack and the related analysis which relies on the state-of-the-art techniques for Nakamoto consensus; please see also [31,32,35–37].

4.1.3. Numerical Evaluation

To illustrate the strength of our protocol in the Bitcoin regime, we pick the following parameters for numerical evaluation of our theoretical results under the system model described in Section 4.1.1. We pick $\lambda_B = \frac{1}{600}$, $\Delta = 10$ s [38] as is usual in Bitcoin and $\alpha = 0.25$ fraction of adversarial power. We further pick $\delta = 1$ s. Notice that the $\delta = 1$ s upper bound on the transmission delay of a proof under 1 KB is extremely pessimistic. Even though we assumed $L \geq 10$, to display the behavior of the curve fully, we plot the double-spending curve for $L \geq 1$ in Figure 4. It is easy to verify that our protocol achieves an upper bound of double-spending probability around 10^{-4} before $L = 50$.

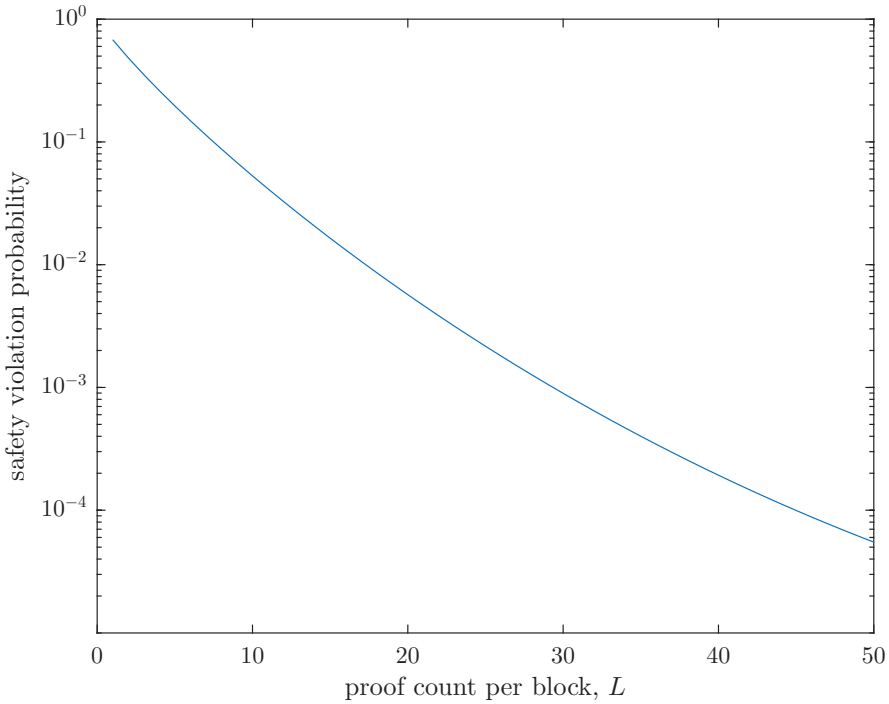


Figure 4. The 1-block confirmation rule with L proofs, $\alpha = 0.25$.

Note that the consistency guarantee of our model is similar to that of parallel PoW protocols [27]. This is an expected result, since by requiring multiple proofs for a block, whether parallel or sequential or a mix of them as ours, we decrease the chance of the adversarial ledger to be lucky with its attack and ahead of honest miners when the honest ledger is confirmed. This is simply a reduction-of-variance argument, which is in essence no different to increasing the b-block confirmation rule for Bitcoin.

The results indicate considerably superior security guarantees compared to Nakamoto consensus if we assume the same block sizes, generation times, and network delays. More specifically, with the 1-block confirmation rule and 10 min block interarrival time, under $L = 50$ and $\alpha = 0.25$, an upper bound of double-spending probability around 10^{-4} is reached within less than 20 min on average in our protocol. In comparison, Bitcoin achieves an upper bound of double-spending probability around 10^{-3} in 220 min on average [32].

4.2. Incentive Attacks

Next, we consider the incentive attacks on our protocol. Mainly, there are two different incentive attacks, one where the adversary creates a block with all the proofs being

adversarial and the other where the adversary hides its proofs for the current height and combines them with honest proofs of the current height to create a block.

The analysis of the former attack is similar to our consistency analysis with $b = 1$, which shows that the attack is less profitable than honest mining as L increases. For example, even if we ignore the potential adversarial block losses in pre-mining and post-confirmation phases but take the probability of adversarial advantage of M_{lead} and $M_{deficit}$ into account, the success probability of replacing 1 honest block is with probability less than 10^{-4} for $L = 50$ and $\alpha = 0.25$. Notice that if the attack fails, the adversary has no proofs in the honest block; hence, its relative rewards are upper bounded by the success probability of the attack, which is less than the relative rewards of honest strategy, i.e., $\alpha = 0.25$, for $L \geq 10$.

We believe it is trivial to do such an analysis, as it is done via Monte Carlo simulations in ([28], Section V, Figure 7), which essentially shows that PoW protocols become more resistant to entire block withholding as the number of proofs required for a single block increases; hence, we omit such trivial confirmation of expected results here.

4.2.1. Proof Withholding

The latter attack is what we call a proof withholding attack in this paper, and it is generally much harder to prevent in parallel PoW protocols. The analysis and discussion we provided in Section 2 show that the existing parallel PoW protocols fail to prevent such attacks for even small values of α, and they perform worse than Nakamoto consensus in general. Our protocol, on the other hand, is a hybrid protocol where a block consists of multiple proofs like parallel PoW [28–30] and multistage PoW protocols [22,25,34]; however, the order of these proofs is semi-sequential, i.e., we allow at most two parallel proofs at a proof height which is inspired by the k-cluster rule in PHANTOM protocol [20] with $k = 1$.

More specifically, since the proof propagation times are small, if we consider the DAG created by the proofs, for any proof within a block, at most one proof is allowed to be not an ancestor or descendant. Although this condition is stricter compared to DAG-style voting of [30], since the proofs we consider are small-sized and only refer to the ledger through the Merkle root, it is satisfied in most network conditions. Thus, the resilience against proof withholding attacks is increased compared to them while multiple proofs are still allowed for a block. Further, a punishment rule, i.e., reward splitting, is introduced to further solidify the resilience inspired by the DECOR+ [18] and RS protocols [19].

4.2.2. Numerical Evaluation

In Appendix F, we explain the MDP model with the appropriate state and action space as well as the objective function for our protocol. Note that we assume the conventional system model used in selfish mining literature with α fraction of the hashpower controlled by a single colluding adversary which has γ network advantage. In Figure 5, we present the results of the MDP models together with the results of Section 2, where we analyzed the relative rewards of the existing parallel PoW protocols. We pick $\gamma = 0.5$ for the MDP models and truncate the results of Section 2 before they rise above the honest behavior for the sake of neatness in the figures. For Nakamoto consensus, we run the MDP model of [4], and the modified model described in Appendix F is run for our protocol with 10^{-5} precision, where α values range from 0.02 to 0.48 with a 0.02 spacing between them.

Note that the effect of γ is ignored in the analysis of Section 2 for Tailstorm and DAG-style voting, which implies that the actual relative reward of the adversary is underestimated for these protocols, and the best possible rewards are even higher than the values we show here. Nevertheless, it is clear that our protocol performs better even when the adversary is underestimated in other protocols.

In our protocol, matching always gives a half reward when $(a, h, fork) = (1, 1, relevant)$ (see Appendix F), whereas the adversarial reward in Nakamoto consensus increases with increasing γ since matching succeeds more often. Thus, in comparison to Nakamoto consensus, our protocol is more resilient against selfish mining attacks when $\gamma \geq 0.3$ and resilience increases even more as γ increases further, which we plot in Appendix F. On the other hand, when $\alpha > 0.28$ as γ drops below 0.3, Nakamoto consensus becomes slightly more resilient. For the relative ratio in other values of γ, we refer the reader to Appendix F.

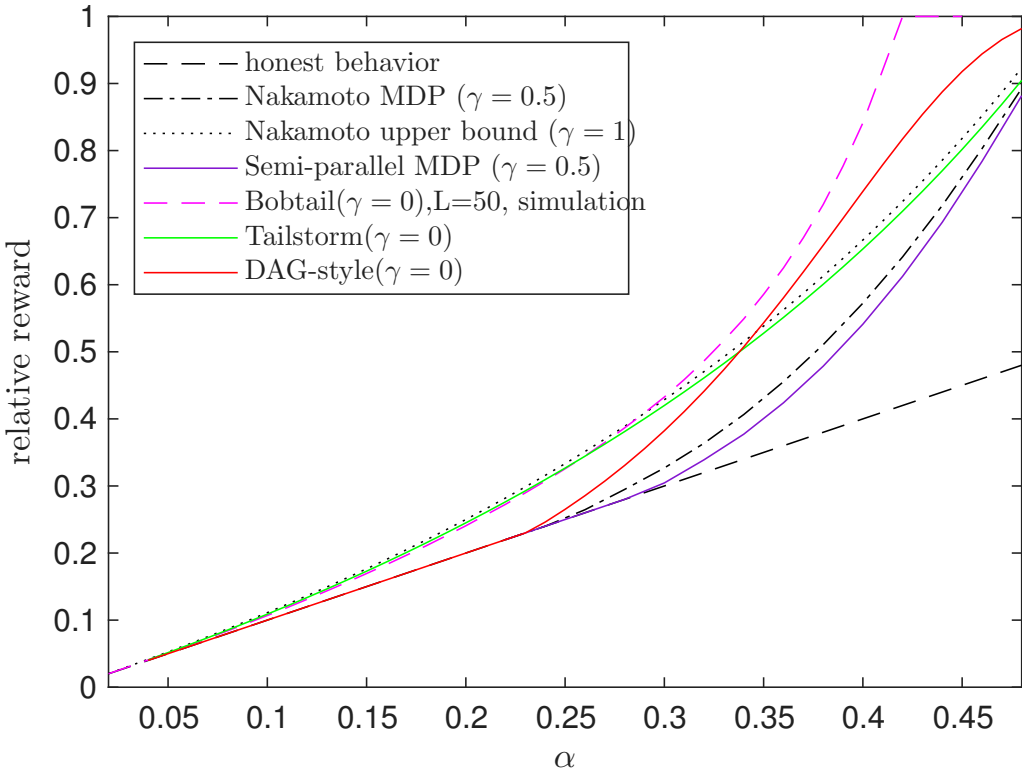


Figure 5. Relative revenues of withholding attacks.

4.3. Transaction Fee Split Rule

The transaction fees associated with a ledger at block height $h - 1$ are divided between the miner who proposed the ledger at block height $h - 1$ and the miners of block height h who voted for the ledger. The ledger creator (leader) receives r fraction of the fees, and the miners of the next block referring to it (voters) share $(1 - r)$ fraction as a whole, which are divided between them in the same way coinbase rewards are divided, i.e., each proof height gets $(1 - r)\frac{1}{C(\omega_L^h)}$ fraction of the total fees and is divided between two proofs if that proof height contains parallel proofs.

Note that we need to pick r such that it is in the interest of all the miners to pick the ledger offering the most fees. To determine an incentive-compatible r, in Appendix G, we consider an attacker that acts against the rule of picking the ledger offering the most fees to increase its expected transaction fee revenue. We show that as long as r is small enough, such an attack is usually not profitable. Moreover, even if profitable for the attacker, when r is small enough, the effect on the transaction fee rewards received by honest miners is minimal. Further, it is trivial to argue that as long as $r > 0$, the adversary should publish its ledger if it offers the most fees. Specifically, when multiple miners have a proposed ledger at block height $h - 1$, if the adversary offers the most fees and it does not share the ledger, another ledger will be shared after the grace period ends, and the adversary loses the potential profits. Thus, the transaction inclusion attack (corresponds to not publishing the ledger in our protocol) considered in Bitcoin-NG settings (Section 5.1) [15] and later

corrected in (Section 4.1) [24] which argues in favor of $\frac{\alpha}{1-\alpha} < r < \frac{1-\alpha}{2-\alpha}$ does not directly apply to our protocol when $L > 1$ since multiple miner are allowed to propose ledgers, and $r > 0$ implies that the adversary should publish its ledger if it offers the most fees. Hence, in our protocol, we choose $r = \epsilon$, i.e., the leader gets a tiny but non-zero fraction of the transaction fee rewards making the attack described in Appendix G essentially ineffective.

5. Conclusions and Limitations

Our protocol can be seen as a hybridization of sequential PoW mechanism with parallel PoW protocols. More specifically, in our protocol, each proof is both a vote for the previous block height and a proposal ledger for the current block height to be voted in the next block. However, the ledger is hidden under a hash summary which decreases the proof propagation times which in turn decreases the extent of parallelness needed in the protocol. Thus, we consider k-cluster rule of [20] with $k = 1$ and apply it to the parallel proofs of a block. In [20], a subset of blocks in a DAG is called a 1-cluster if for every block in the subset all other blocks except one are either an ancestor or descendant. Then, for any block, the block that is not an ancestor or descendant has to be a parallel block in the sense that it should have the same ancestors and descendants. The role of k in [20] essentially comes from the fact that multiple blocks can be generated during a block delay which are non-sequential blocks and would not be accepted in the original Nakamoto consensus. Thus, the authors of [20] propose a modification to allow k non-sequential DAG blocks. In our protocol, we pick $k = 1$ since, instead of blocks, the proofs are propagated through the network, and the size of each proof is very small.

On the other hand, when the Lth proof is generated, a block ledger is propagated through the network where the pick of $k = 1$ might be too small for that specific proof height if the throughput is set high. This could be readjusted in the protocol by allowing more than 2 parallel proofs in the last proof height. This in turn would require a different reward split rule for the last height. In this case, their ledgers could be ignored as well in order to prevent ledger replications. Here, we assume that such an event is negligible and leave it to the specific designers to consider these situations according to their throughput choice.

Note that completely ignoring parallel proofs or accepting but not rewarding them would be against the original intention of creating a parallel PoW protocol that is fair and has less variance and allows proofs to be mined in parallel. On the other hand, allowing too many parallel proofs allows the adversary to mount withholding attacks as considered in Section 2. Moreover, it is also unnecessary to allow too many parallel proofs if proof sizes are very small that can be propagated through the network fairly quickly as in our protocol. Thus, our pick of $k = 1$ is a compromise between them. Future work could consider fair rewarding mechanisms while allowing higher k-cluster proof DAGs, where k can be seen as a measure of parallelness.

From the perspective of incentive mechanism, refs. [17–19] consider rules for Nakamoto consensus that increase the resilience against selfish mining attacks where the protocols considered in these works are very similar to 1-cluster DAG of [20]; hence, we pick similar reward/punishment mechanisms as considered in those papers. We note that, according to [39], the orphan rewarding mechanisms make sure that the honest strategy is the best strategy in terms of the profitability; however, such an analysis ignores the dynamics of rational but honest miners who are free to leave when they are making more losses as explained in Appendix B. In our algorithm, we still resort to the orphan reward mechanism by rewarding parallel proofs. However, inherently, the goal in parallel PoW is to allow multiple proofs to be mined concurrently. Hence, it is not fair to reward one more than the other when two parallel proofs exist. Further, allowing orphans from the past into the main chain makes the parallel PoW protocols vulnerable to proof withholding attacks. Thus, we pick reward splitting mechanism as a compromise.

Notice that the authors of [19] acknowledge the dilemma of rewarding the bad or punishing the good in a protocol design. They show that punishment protocols such as RS achieve better resilience against double-spending and selfish mining attacks when $\gamma = 0.5$, but they suffer more against the feather-forking attacks [40]. However, the analysis of resilience against a censorship attacker in our protocol is different from the protocols considered in [19] since a feather-forking attacker would not be able to see the content of the transaction ledger associated with a proof until L-proofs are mined. Until then, the attacker will contribute to the same proof DAG as honest miners, and it only mounts the attack after L proofs are mined and if the ledger with maximum fee contains the transaction that attacker tries to censor. We leave the analysis of this type of attack to future work.

From the perspective of the throughput increase, our protocol is much less restricted compared to the existing work since the ledger is separated from the proofs. As a result, the ledger size can be increased arbitrarily with two things to bear in mind. First, as ledger size is increased, Δ increases which in turn leads to higher number of proofs that are mined during a grace period. These proofs should be treated differently than the parallel proofs on other proof heights and rewarded fairly, which we leave to specific implementations of designers. Second, although the ledger is created by a single miner for each block, which prevents transaction conflicts, when the ledger size is increased, a ledger creator gets more control of the transactions picked for a ledger. However, the effect of this is mitigated by the fact that the ledger offering the most fees is picked which prevents the ledger creators from acting at their own pleasure when creating a ledger.

Although we analyzed incentive attacks on the coinbase rewards as well as the transaction fees individually, there is still a need to develop an analysis considering both attacks at the same time to give stronger guarantees on the resilience against incentive attacks. However, we note that even for Nakamoto consensus, which has been thoroughly studied for over fifteen years, such a rigorous analysis is still missing [41].

Author Contributions: Conceptualization, M.D.; methodology, M.D.; formal analysis, M.D.; investigation, M.D.; experimental verification, M.D.; validation, M.D. and S.U.; writing—original draft preparation, M.D.; writing—review and editing, S.U.; supervision, S.U.; project administration, S.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Bobtail Smallest-Hash Attacks

Appendix A.1. DoS Attack in Bobtail

If the adversary has the lowest hash valued proof which is published together with $L - 1$ other valid proofs mined by everyone, it can simply refuse to publish the block, i.e., Denial-of-Service (DoS) attack. This attack, acknowledged in ([28], Section VIII.C) , requires honest miners to further mine until finding a lower hash value to create a block.

Appendix A.2. Proof Withholding in Bobtail

The authors of Bobtail also consider Monte Carlo simulations of proof withholding attacks where a miner simply withholds all its proofs for the current block height until it finds the lowest hash valued proof or gives up the attack and releases all its proofs if it seems unlikely to do so before honest miners creating a block with all proofs honest ([28], Section VIII.B).

Appendix A.3. A DoS Proof Withholding Attack

Here, we combine the two attacks described above which results in a stronger attack with a profitability threshold of zero. In the case of the attack described in Appendix A.2, we note that the authors do not consider that if the adversary manages to mine the proof with the lowest hash value for the current block height, it can release this lowest hash proof while withholding the rest of the proofs, to build a lead for the proofs of the next blocks. This in turn becomes similar to the DoS attack described earlier, since honest miners will have to find a proof with lower hash value. We describe this hard-coded attack next and provide simulations to display the effectiveness of the attack.

Let us assume that everyone works on the same block and L proofs are needed. When honest miners create a proof, they release immediately, whereas the adversary keeps its proofs hidden; however, while picking a support value, it acts honestly, i.e., whenever it mines a proof, it simply picks the lowest hash value among all proofs mined so far as the support value, no matter if that proof is mined by honest miners or the adversary itself. When the system has L proofs in total, the adversary (which produced $L\alpha$ proofs on average) checks if it has the lowest hash proof among all L proofs:

1. If the adversary does not have the lowest hash value, it simply releases all its proofs, and the honest miner who has produced the lowest hash value creates the block including all the L proofs. The adversary does not lose any primary or bonus reward described in ([28], Section VIII).
2. If the lowest hash valued proof belongs to the adversary, it releases this proof, i.e., makes it public, while keeping the remaining proofs ($L\alpha - 1$ proofs on average) hidden. At that point, honest miners see $L(1-\alpha)$ honest proofs on average plus 1 adversarial proof which has the lowest hash value. They keep mining to complete L public proofs since they are not aware of the hidden adversarial proofs, where they use the adversarial proof as support until they mine a proof with lower hash value. The adversary, on the other hand, moves on to mine the next block to build a lead.
   When L public proofs are completed ($L-1$ honest and 1 adversarial):
   - If the honest miners already produced a proof with lower than the public adversarial proof, they can create a block. To prevent that, the adversary releases all its work for that block plus some additional work from the lead that it has created for the next block to undo all honest work. It simply loses no reward and undoes some honest work.
   - If the public adversarial proof is still the lowest hash value, the honest miners have to waste their time on the current block until they find a lower hash, while the adversary increases its lead further on the next block. It will simply undo the honest work when honest miners find a lower hash value. It simply loses no reward and undoes some honest work.

We also provide pseudo-code for the attack in Algorithm A1. None of the attack prevention mechanisms of Bobtail ([28], Rule I, II, Section VIII.B) are effective against this attack since the system follows most aggregate work; hence, the adversarial chain can undo the honest work, increasing the relative reward of the adversary. In fact, the bonus reward system introduced in ([28], Section VIII) will even worsen the situation for honest miners. If honest miners mine the lowest hash proof, the adversary gets the bonus reward as it acts honestly when referring to the lowest hash proof. If the adversary mines the lowest hash proof, honest miners become aware of the lowest hash valued proof after there are already L proofs in the system as the adversary keeps private before that point; hence, honest miners lose the bonus rewards.

Algorithm A1 Bobtail DoS Proof Withholding Attack

```
while attack do
    cur_blck_adv_prf_cnt := initial_released_lead + still_hidden_lead()
    cur_blck_tot_prf_cnt := cur_blck_adv_prf_cnt + cur_blck_hon_prf_cnt
    mine_hidden_until_L_total_proofs()
    if is_smallest_hash_adversarial() then
        release_smallest_hash_prf()
        while is_smallest_hash_still_adversarial() do
            mine_lead_for_next_block()
        end while
        if is_lead_enough_to_undo_honest_work() then
            release_cur_adv_block()
            release_initial_lead_for_next_block()
        else
            adv_loses_rewards_for_cur_block_and_lead()
        end if
    else
        release_all_proofs()
    end if
    continue_next_block_loop()
end while
return relative_reward_ratio
```

Note that the adversary has access to all public information and can track the public work and thus avoid the risk of losing its hidden work. This attack results in higher profits for the attacker even for small α and γ values, and the profitability threshold is $\alpha_T = 0$. The reasoning is no different from the selfish mining attack on smallest-hash tie-breaking (SHTB) rule which has a profitability threshold of $\alpha_T = 0$. The authors of [19] show that the informational advantage an attacker has in SHTB protocols allows it to gain more rewards than honest mining by withholding attacks in the smallest-hash regions. Same arguments hold here which shows that the profitability threshold is $\alpha_T = 0$, which is worse than Nakamoto consensus.

Notice that to succeed further in this attack, the adversary can further modify the attack to keep its proofs hidden (Step 2) only if the smallest hash it has produced is below a certain threshold. It could simply minimize its risk by picking a low enough smallest-hash region, as mentioned in [19]. For example, in Figure 1, if p_5 is mined by the adversary and the adversary is confident that the hash value of p_5 is low enough (in smallest-hash region), it can refuse to share the block it creates, and it can safely mine for the next block before honest miners can do so for a while, gaining an advantage. Although the authors state that DoS attacks can be thwarted by banning the attackers, such a defense mechanism is not straightforward in permissionless blockchain settings with easy-to-obtain sybil identities.

We first provide some theoretical intuition for the attack and then explain the simulation environment. Notice that the first time there are L proofs in total, i.e., counting both the adversarial and honest proofs, with probability α, the lowest hash proof belongs to the adversary. In other words, initially, Case 2 happens in every $\frac{1}{\alpha}$ blocks on average. Once this happens, the fraction of adversarial proofs for the next block will further increase since the adversary mines a lead for the next block before honest miners can find a smaller hash for the current block. Considering that the value of the hashes are essentially numbers in bits, which can be represented as uniform distribution on $[0, 1]$, this attack actually enters in vicious cycle where the adversary has practically infinite lead.

To understand why, consider the distribution of the minimum value of L i.i.d. uniform distributions, which has the following pdf and cdf

$$f(x) = L(1-x)^{L-1}, \tag{A1}$$

$$F(x) = 1 - (1-x)^{L}. \tag{A2}$$

Let H_{waste} denote the number of honest proofs that has to be created until the honest miners hit a lower hash value. Assuming the adversary has created the lowest hash value x,

$$\mathbb{E}[H_{waste}|x] = \frac{1}{x}. \tag{A3}$$

Now, considering the distribution of x, we have

$$\mathbb{E}[H_{waste}] = \int_0^1 L\frac{(1-x)^{L-1}}{x}\,dx \tag{A4}$$

$$= L\left[\ln x + \sum_{n=1}^{L-1}\binom{L-1}{n}(-1)^n\frac{x^n}{n}\right]_0^1, \tag{A5}$$

which diverges. This essentially means that the adversary is able to hit such low hash values that the honest miners have to waste virtually infinite amount of energy to replace it. From a practical perspective, if a DoS happens and the block is not created after the first L public proofs, honest miners could start over, ignoring the adversarial lowest hash proof. In this case, they will start to mine until they create L new proofs or a proof with lower hash than the adversarial, whichever comes first. Thus, H_{waste} becomes essentially the expectation of a truncated geometric distribution capped at $(L+N)$, where N denotes the number of hidden adversarial proofs that were created for the current block before the adversary started to build a lead for the next block. For the first block for which the adversary gets the lowest hash proof, we have

$$\mathbb{E}[H_{waste}] = \int_0^1 L(1-x)^{L-1} \times \left(\sum_{k=1}^{L+N-1} k(1-x)^{k-1}x + (L+N)(1-x)^{L+N-1}\right)dx \tag{A6}$$

$$= L\sum_{k=1}^{L+N-1}\frac{k}{(L+k)(L+k-1)} + \frac{L(L+N)}{2L+N-1}. \tag{A7}$$

If we assume $N \approx L\alpha - 1$, with $L \geq 10$, this in turn results in more than $\frac{2L}{3}$ honest proof waste for $\alpha > 0$. Meanwhile, the adversary creates the proofs for the next blocks, increasing the fraction of the adversarial proofs for the next blocks, which in turn can increase average honest waste further for the following blocks if the adversary gets the lowest hash again.

We note that more sophisticated attacks that account for the lead in the next blocks are possible. Further, one can analyze the scenarios with different network latency and more heterogeneous topologies for Bobtail by adopting selfish mining models developed for Nakamoto consensus such as ([6], Section V) to estimate γ. We leave these considerations to the reader if interested and move on to simulate the attack with $\gamma = 0$, which favors the honest miners in case of forks and the rule where the honest miners start over the mining process for the current block with ignoring the adversarial lowest hash proof if the adversary has it and refuses to publish a block.

Results with the practical rule are presented in Figure A1, where we simulate the attack for 10^6 blocks for $L = 10, 25, 50$. Even with the extra protection that we add with this rule in favor of the honest miners, it is clear that the profitability threshold is zero, and the adversarial relative reward is on par with the upper bound of the selfish mining attack in Nakamoto consensus. As α rises above $\frac{1}{3}$, the attack becomes even more powerful than the upper bound of the selfish mining attack in Nakamoto consensus. Moreover, without the practical rule we introduced, our simulations do verify our theory that the explosion happens (honest miners essentially waste practically close to infinite number of proofs) when we simulate a sufficiently large number of blocks.

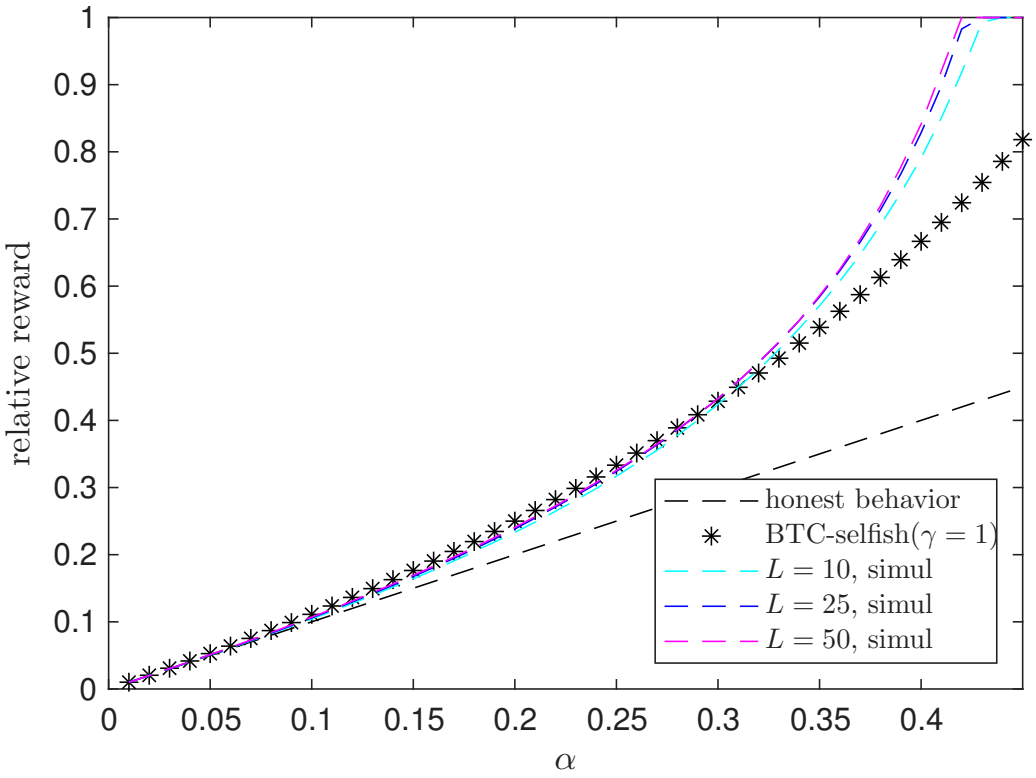


Figure A1. Bobtail relative reward of DoS proof withholding attack.

Appendix B. Why Relative Reward Metric?

To understand why a relative reward metric is the correct choice and why the normalized reward metric of [29,30] is an illusion, consider the adversarial (honest resp.) miners who hold α ($1-\alpha$ resp.) fraction of the hashpower in the system. Let the total adversarial (honest resp.) reward per chain progress be denoted as μ (ν resp.). Clearly, an incentive-compatible system that is not under an attack should satisfy

$$\frac{\nu}{1-\alpha} = \frac{\mu}{\alpha}, \tag{A8}$$

which is an equality between the rewards per chain progress per computational power of the adversarial and honest miners.

When the adversary is employing an incentive attack, let μ_{attack} and ν_{attack} denote the new rewards per chain progress, and let ρ denote the fraction of the total rewards the adversary gets. Assume that the adversary gets more relative rewards than its fair share, i.e., $\rho > \alpha$, but the total adversarial reward per chain progress decreases compared to not employing the attack i.e., $\mu > \mu_{attack}$.

We will analyze this system in more detail. However, to give the gist of our argument in a few words, if such a system that is under attack is safe as claimed in [29,30], then honest mining should be sustainable even if the adversary launches the attack. However, if honest mining is sustainable under the attack given the mining rewards and costs, then the adversarial mining is not only sustainable but also more profitable since the adversary gets more relative rewards compared to its costs.

Specifically, an incentive-compatible system that is safe against the attack should have

$$\frac{\nu_{attack}}{1-\alpha} \geq \frac{\mu_{attack}}{\alpha}, \tag{A9}$$

which compares the rewards per chain progress per computational power under the attack which is not satisfied when $\alpha < \rho$, since

$$\frac{\nu_{attack}}{1-\alpha} < \frac{\nu_{attack}}{1-\rho} = \frac{\mu_{attack}}{\rho} < \frac{\mu_{attack}}{\alpha}. \tag{A10}$$

Notice that in this situation, if the adversary is getting less reward per chain progress in an incentive attack, (A10) implies that the honest miners are affected even worse since relative reward ρ is in favor of the adversary. In other words, under the attack, the normalized honest loss is larger, i.e.,

$$\frac{\nu - \nu_{attack}}{1-\alpha} > \frac{\mu - \mu_{attack}}{\alpha}. \tag{A11}$$

As a result, if attacking is not profitable for the adversary despite getting higher relative reward than its fair share, then for honest miners the situation is worse. In an open economy model, if the honest miners are rational and they are making losses (greater than the adversary per computational power) and their mining costs are not compensated by the rewards, they will leave the system before the adversary, which breaks down the whole blockchain system. This is even more severe than having a system that rewards miners unfairly.

On the other hand, if the system is secure under the attacks (as [29,30] claim) and honest miners are not leaving the system in an open economy model, it implies that the coin exchange value compared to their mining costs dynamically changed, and honest mining became sustainable, which in turn implies that the adversary is making unfair profits as it gets more than its fair share. In other words, under the attack, if the honest reward per chain progress per computational power $\frac{\nu_{attack}}{1-\alpha}$ is sustainable when taking the mining costs into account, then $\rho > \alpha$ implies that adversarial reward per chain progress per computational power is not only sustainable but also more profitable, i.e.,

$$\frac{\mu_{attack}}{\alpha} > \frac{\mu_{attack}}{\rho} > \frac{\nu_{attack}}{1-\alpha}, \tag{A12}$$

where ρ, i.e., the relative reward metric, can be compared to α to measure this extra profit.

In short, if the total reward issued per chain progress is $\mu + \nu$, then an incentive-compatible system should satisfy $\mu = (\mu + \nu)\alpha$. If the total rewards is $c(\mu + \nu)$, then an incentive-compatible system should satisfy $c\mu = c(\mu + \nu)\alpha$. If the adversarial fraction of the rewards satisfies $\rho > \alpha$, the system is not incentive-compatible. The specific value of c does not matter, as the number of coins issued will not change the market cap of the system and only changes the value of a single coin.

Hence, the normalization of [29,30] is ignoring the hashpower of the entities and simply focusing on the reward per chain/time progress, whereas a true normalization should compare the reward per chain/time progress per hashpower for each entity, i.e., $\frac{\mu_{attack}}{\alpha}$ and $\frac{\nu_{attack}}{1-\alpha}$, i.e., check if (A9) is satisfied or not, which is equivalent to checking if $\rho \leq \alpha$ or not. We note that the sustainability analysis here focuses on the long run market dynamics, and in the short-term both honest and adversarial miners make relative losses compared to before the attack. However, it is well known that this short-term loss is present in general for selfish mining attacks [39,42–44]. Further, from a macroeconomic perspective, minting more or less coins in a cryptocurrency environment is not relevant either, as eventually the share of the coins held by an agent determines its purchasing power, not the amount of coins minted [45].

Appendix C. Proof Withholding Attack for Tailstorm and DAG-Style Voting

We start by explaining the attack structure. Assume that the adversary withholds its proofs at the current block height, call it h, and combines them with the honest proofs of the current block height to create the block associated with these proofs but withholds it. In this situation, this adversarial block contains α fraction of the adversarial proofs on average.

Next, it starts creating proofs for the next height, $h+1$, while honest miners keep mining on the previous height h since the adversary did not share its own proofs for block height h. When honest miners find the final (Lth) proof to create a block at block height h, the adversary simply releases its block for height h, plus one additional proof for height $h+1$ to make sure the public part of its chain has the most aggregate work and some honest work is undone. Hence, everyone starts to create proofs for block height $h+1$ on top of the public block at height h released by the adversary recently. Notice that at this point there is some initial advantage that the adversary has for proofs at height $h+1$ since it started mining on them as soon as it created the block at height h. The adversary keeps these proofs and the others it creates later private again until honest miners reach L proofs for block height $h+1$, at which point the adversary releases its block at height $h+1$ and one additional proof on height $h+2$ and keeps the rest of the proofs private again. The attack continues in a similar manner for the next blocks.

In Figure A2, we visualize the proof withholding attack in Tailstorm protocol with $L=7$. The circles and rhombus are proofs created by the honest and adversarial miners, respectively, and the numbers inside represent the rewards they get. Black circles represent the honest work that does not end up in the longest chain due to the attack. At block h, initially only the red proof is created and shared by the adversary, and honest miners create 4 proofs (blue) which they share while the adversary creates 2 more proofs (green) that it hides. Since honest miners are not aware of the 2 adversarial proofs, they create 2 more proofs (black). At this point, the adversary shares the green proofs at block h together with the red proof at block $h+1$, which makes sure the black circled honest proofs are replaced in the longest chain. The attack continues in a similar manner for next heights.

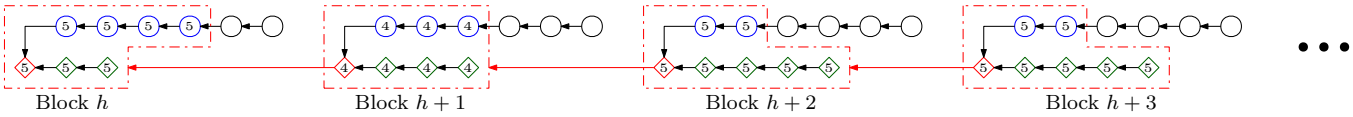


Figure A2. Tailstorm withholding attack.

We first prove the fraction of the adversarial proofs in the public chain in the long run under the attack.

Lemma A1. *For both tree- and DAG-style voting, in the long run, under the proof withholding attack, the fraction of the adversarial proofs in the public chain reaches* $\frac{\alpha}{1-\alpha}\frac{L-1}{L}$ *even when* $\gamma = 0$.

Proof. Let a_n be the fraction of adversarial proofs in the public block at height n. Let b_{n+1} be the fractional advantage of the adversary for block height $n+1$, i.e., the fraction of the proofs it already had for block at height $n+1$ when it released the complete block at height n. Then, according to the proof withholding strategy described above, the following holds,

$$\mathbb{E}[b_n|a_{n-1}] = \frac{\alpha}{1-\alpha}\left(a_{n-1} - \frac{1}{L}\right), \tag{A13}$$

$$\mathbb{E}[a_n|b_n, a_{n-1}] = b_n + (1-b_n)\alpha = \alpha(1+a_{n-1}) - \frac{\alpha}{L}. \tag{A14}$$

The reasoning is as follows: Given a_{n-1}, honest miners were not aware of $a_{n-1} - \frac{1}{L}$ fraction of the proofs created by the adversary at height $n-1$. Note that 1 adversarial proof of the height $n-1$ was released initially to make sure the switch to adversarial chain happens, which explains $\frac{1}{L}$ in (A13). Then, while honest miners try to mine $a_{n-1} - \frac{1}{L}$ fraction of the proofs to create the block at height $n-1$, the adversary mines on height n. For every honest proof at height $n-1$, the adversary creates $\frac{\alpha}{1-\alpha}$ proofs at height n on average, which explains (A13). Given b_n, α fraction of the rest of the proofs at height n are mined by the

adversary, which explains (A14). Clearly $\mathbb{E}[a_1] = \alpha$ which is for the first block where the attack started, which together with (A13) and (A14) implies

$$\lim_{n\to\infty} \mathbb{E}[a_n] = \frac{\alpha}{1-\alpha}\frac{L-1}{L}. \tag{A15}$$

Hence, in tree- and DAG-style voting protocols, the fraction of the proofs created by the adversary reaches $\frac{\alpha}{1-\alpha}$ in expectation as $L \to \infty$, which is the worst-case scenario of selfish mining in Nakamoto consensus, i.e., when $\gamma = 1$ [3,4]. Note that there can be better attack strategies. For example, the attack did not take advantage of γ; hence, the adversary always released 1 extra proof which it does not have to do for higher γ values. Thus, $\frac{\alpha}{1-\alpha}\frac{L-1}{L}$ is essentially a lower bound. □

We note that the analysis above is valid for $\alpha \geq \frac{1}{L}$ with L being large enough. A more rigorous analysis is needed that has to take into account the cases where the adversary becomes extremely lucky and has $b_{n+1} > 1$ for some block heights n as well as extremely unlucky cases and has $a_n < \frac{1}{L}$ for some block heights n. Although these cases can slightly change the final result of the analysis, for the purposes and claims of our results, the above analysis that considers the expected values is sufficient.

With the hard-coded proof withholding attack, the fraction of the adversarial proofs in the public chain reaches $\frac{\alpha}{1-\alpha}\frac{L-1}{L}$ even when $\gamma = 0$, for both tree- and DAG-style voting. Note that in Tailstorm [29], although the reward a block gets is scaled by the depth of the proof tree, everyone is punished and rewarded equally. Hence, the fraction of adversarial proofs we derive is the fraction of the rewards the adversary gets. We note that the authors of [29] consider a protocol variant, TS/const, where the relative reward of the adversary equals the reward per chain progress. Their simulations include an incentive attack called minor delay against TS/const, which seems similar to the attack we describe here. However, the claims of the authors in ([29], Figure 4) regarding the attack are not verifiable given our theoretical analysis. Hence, in Appendix C.2, we provide simulations of a practical implementation of the attack for 10^5 blocks and verify our theoretical results.

Appendix C.1. DAG Version of the Attack

The fraction of the adversarial proofs under the attack does not equal the relative rewards of the adversary in DAG-style voting since we need to count the number of ancestors and descendants for each proof. To make the explanation easier to follow, we refer to Figure A3 whenever suitable. The colors and shapes have the same meaning as in Figure A2, but we additionally have violet colored adversarial proofs that indicate that those proofs were mined before the red proof was shared. The number inside the proofs represent the reward each proof gets. We note that in the DAG version of the attack, the adversary points to all other adversarial proofs and all public honest proofs even though it does not release its own proofs immediately.

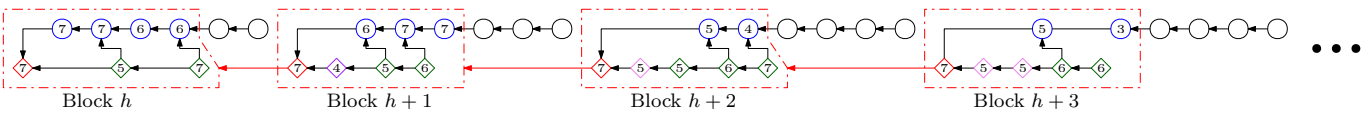


Figure A3. DAG version of the withholding attack.

In short, for each block, the red proof is the advantage proof shared initially by the adversary and seen by everyone and hence gets the full reward. All adversarial proofs get rewards proportional to the total number of adversarial proofs in the block. The greens (adversarial) additionally get rewards for the honest proofs they can see when they were mined. Blue proofs (honest) also get rewards for each green proof that refers to them in addition to the total number of blue and red proofs.

Lemma A2. *In the DAG version of the proof withholding attack, the fraction of the adversarial rewards reaches*

$$\rho = \frac{\mathbb{E}[A_R]}{\mathbb{E}[A_R] + \mathbb{E}[H_R]}, \tag{A16}$$

where

$$\mathbb{E}[A_R] = 1 \cdot L + (L\mathbb{E}[b_n] - 1) \cdot L\mathbb{E}[a_n] + L(\mathbb{E}[a_n] - \mathbb{E}[b_n]) \cdot L\frac{1 + \mathbb{E}[a_n]}{2}, \tag{A17}$$

$$\mathbb{E}[H_R] = L(1 - \mathbb{E}[a_n]) \cdot (1 + L(1 - \frac{\mathbb{E}[a_n] + \mathbb{E}[b_n]}{2})). \tag{A18}$$

Proof. The adversarial proof released initially to make sure the switch to adversarial chain happens is seen by everyone and simply gets L rewards. This is represented as the red rhombuses in Figure A3. The number of advantage proofs, i.e., the proofs that the adversary mined before honest miners finished the previous block height, equals $Lb_n - 1$ according to Lemma A1. These proofs are represented as the violet rhombuses in Figure A3 and are seen by only the other adversarial proofs and do not refer to any honest proof. This is because, while they were being mined, honest miners were still working on the previous height. Hence, there was no honest proof at that block height to refer to when the violet proofs were mined. As a result, these proofs get La_n rewards, i.e., the number of adversarial proofs at that height.

It remains to deal with the rest of the adversarial proofs, colored green in Figure A3. It is easy to see that there are $L(a_n - b_n)$ of them. However, the reward they get is slightly more complicated than the other ones because we have to count how many honest proofs they can refer to, i.e., the number of honest proofs that were already mined at their creation time. Notice that the first green adversarial proof is likely to see almost no honest proofs, whereas the last adversarial proof sees almost all honest proofs. As the mining time of the proofs are uniformly distributed, on average, each of these green adversarial proofs sees $L\frac{1-a_n}{2}$ honest proofs at their block height, i.e., on average half of the honest proofs at their height. Of course, they also see all the adversarial proofs, i.e., La_n of them.

Combining the counts and rewards of each color mentioned above, where we multiply the expectation of the count of each proof color and their expected rewards, the reward of the adversary equals (A17) on average from each block.

For the honest proofs, colored blue, their total count is simply $L(1 - a_n)$, and similar reasoning applies. Honest proofs, in addition to seeing each other, also see the red adversarial proof but no other adversarial proof. Thus, their reward coming from blue and red proofs is $1 + L(1 - a_n)$ in total. However, they are also referred to by green proofs and hence also get rewarded from them. Note that the first blue proof is likely to be seen by almost all green proofs, whereas the last one is likely to be seen by almost none. We can make the argument about uniform distribution of mining times again; hence, each honest proof gets an average reward $\frac{L(a_n - b_n)}{2}$, coming from the green proofs that refer to it.

Combining the counts and rewards of each color mentioned above, the reward of the honest miners is equal to (A18) on average from each block. Note that both $\mathbb{E}[a_n]$ and $\mathbb{E}[b_n]$ are derived in Lemma A1. □

Appendix C.2. Simulations

As the results claimed by [29] are not compatible with our theoretical analysis, we simulate the attacks in both tree- and DAG-style voting where we apply some practical modifications to the implementation of the attack since L is finite. The modification is as follows: The first time when there are L proofs including the hidden adversarial ones for block n, the adversary releases all its proofs if the number of hidden proofs it has is not

more than $\frac{1}{\alpha}$. This is because, in expectation, it takes $\frac{1}{\alpha}$ proofs to encounter an adversarial proof, which will be the initial lead for the next block that undoes the honest work. If the adversary has more than this, it is reasonable to take the risk; otherwise, if it does not release the hidden proofs, it will simply lose them since there will be more newly mined honest proofs (black proofs in Figures A2 and A3). This situation happens because L is finite and we have to deal with integers. The pseudo-code of the practical version of the attack is provided in Algorithm A2.

The results are provided in Figure A4, where we run the simulation for 10^5 blocks for each parameter. As we mentioned in our theoretical analysis, the accuracy of our analysis depends on L as we use expected values. Hence, as L increases, the theoretical and simulation values converge. Further, the attack does not need any network advantage γ, and both protocols become less resistant to incentive attacks than the original Nakamoto consensus. We note that we did not dig through all possible ways to find better practical implementations for which the simulation results could converge faster even for smaller L. We leave this to interested readers, since our purpose is to show the vulnerability of the existing parallel PoW protocols.

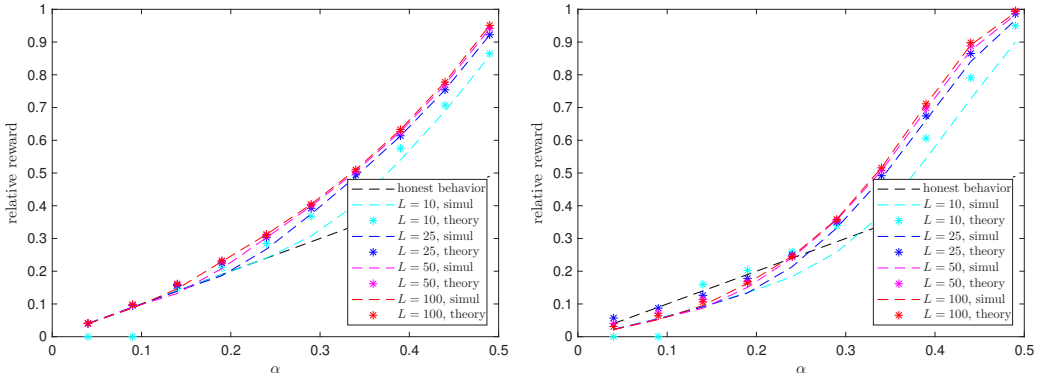


Figure A4. Withholding attack simulation, $\gamma = 0$. (**a**) Tailstorm version; (**b**) DAG version.

Algorithm A2 Practical Withholding Attack for [29,30]

```
while attack do
    cur_blck_adv_prf_cnt := initial_released_lead + still_hidden_lead()
    cur_blck_tot_prf_cnt := cur_blck_adv_prf_cnt + cur_blck_hon_prf_cnt
    mine_hidden_until_L_total_proofs()
    cur_blck_public_prf_cnt := initial_released_lead + cur_blck_hon_prf_cnt
    if cur_blck_hidden_adv_prf_cnt > 1/α then
        while cur_blck_public_prf_cnt < L do
            mine_lead_for_next_block()
        end while
        if is_lead_enough_to_undo_honest_work() then
            release_cur_adv_block()
            release_initial_lead_for_next_block()
        else
            adv_loses_rewards_for_cur_block_and_lead()
        end if
    else
        release_all_proofs()
    end if
    continue_next_block_loop()
end while
return relative_reward_ratio
```

Appendix D. Fork Choice Rule

The fork choice rule is to pick the chain that contains the most aggregate work at all times. Thus, if L proofs are mined for a block and the adversary owns the valid ledger offering the most fees but shares the ledger with only some of the honest miners in the grace period, these honest miners try to create an initiator proof of the next block height referring to the adversarial ledger, while other honest miners try to create an initiator proof referring to another ledger. However, this is not a threat to the consensus mechanism since as soon as someone creates and shares the next initiator proof, every honest miner will pick that newly created proof and the ledger it refers to as the fork choice since it contains the most aggregate work. In such a situation, by not sharing the ledger with all honest miners in time, the adversary risks any potential extra income due to owning the ledger offering the most fees since the first initiator proof created next might refer to another ledger as the winning ledger.

On the other hand, if an honest miner owns the ledger offering the most fees and the adversary creates the first initiator proof for the next block height, honest miners will accept the adversarial initiator proof since it is the chain with most aggregate work. Notice, the adversarial initiator proof might refer to another valid ledger instead of the ledger offering the most fees. Such a situation does not create any issue from the perspective of the consensus since the fork choice rule is always the same, i.e., pick the chain that contains the most aggregate work. From the perspective of the transaction fee revenues, in Section 4.3 and Appendix G, we show that such an attack is not profitable for the adversary.

Appendix E. Analysis of the Double-Spending Attack

Appendix E.1. The Attack

Essentially, we have to analyze the following. At block $h+1$, the miners start mining (voting) on ledgers that are created by miners of the proofs at block h. When an honest ledger of block h receives L votes at block $h+1$ and is part of the heaviest chain, the ledger is confirmed. See for example ω_6^h in Figure 3, which is confirmed when block $h+1$ is fully created. Any time after the honest ledger is confirmed, an adversarial ledger from block h; for example, ω_7^h in Figure 3 would be confirmed and replace the honest ledger, if a heavier chain is published. Such a replacement would be an instance of double-spending in our analysis.

We first recall that a double-spending analysis is more complicated than simply analyzing a private attack when network delays are present. In other words, when network delays are present, the views of honest miners differ and some honest miners can be lured into working for the adversary temporarily during delays, increasing its hashrate, known as balanced attacks [35] (Lemma F. 1). This in turn complicates the analysis.

Without network delays, a private attack can be shown to succeed if any other strategy succeeds in Nakamoto consensus [35] (Theorem 5.1). This result is easily extendable to the heaviest chain rule here if we had no network delays. On the other hand, to analyze double-spending under network delays, we will give the adversary certain advantages that simplify our analysis and give an upper bound for the double-spending probability. In other words, since pinning down the exact worst-case scenario adversarial attack vector is much harder, by overestimating the capabilities of the adversary, we will provide achievable consistency results from the honest miners' perspective which are upper bounds on the worst-case scenario adversarial attack vector.

Here, transaction fee fork choice rule does not matter as the most aggregate rule is always in place, and we break ties in favor of the adversary. The Δ considered here can be seen as 2 times the original Δ value in the protocol description since the adversary can

avoid releasing its ledger if it offers the most fees resulting in doubling of a valid ledger propagation time. We ignore such a triviality here as it does not affect our analysis.

Appendix E.2. Analysis

Assume at time τ that the block h has L proofs and a ledger of these proofs are starting to be disseminated through the network. At this point, the honest miners need Δ time to receive an honest ledger of block h. From this point onward, honest miners will have to create a complete block at height $h+1$ with L (or $L+1$) proofs, each of which will experience a δ delay. When these proofs are mined and dispersed, another Δ time is needed for the proposed ledger of block $h+1$ to be received by honest miners. We refer to the time the ledger of block $h+1$ is received by all honest miners as τ_c. Notice that at τ_c, the ledger of the block h is confirmed according to 1-block confirmation rule since block $h+1$ is complete. Thus, the interval $[\tau, \tau_c]$ is called the confirmation interval.

First, note that at time τ, the adversary might have a private chain that has more work than the public work observable by the honest miners. This initial adversarial advantage is called the lead. We refer to the interval where the adversary mines to try to build a lead as the pre-mining phase as in [31,32]. At τ_c, when the ledger is confirmed, if the adversary has a private chain with more work, it can release it to convince honest miners to switch to the adversarial ledger, which will undo the honest work. Otherwise, it can still keep mining on its own chain in the hopes of catching the honest work and replacing their ledger. This interval after τ_c is called the post-confirmation race as in [31,32].

We start by assuming that every honest proof experiences maximal δ delay (and the ledger experiences maximal Δ delay), and any honest proof generated within that delay is converted to an adversarial proof, where the adversary can decide where to put it. In other words, when an honest proof is generated, within the following δ time, honest miners are assumed to stop mining, and the adversarial hashpower rises from $\alpha \cdot HP$ to $1 \cdot HP$ for this δ-time, where we denote full hashpower of the system as HP. This additional adversarial advantage is inspired by the rigged model of [32] which is a refined version of the rigged model considered in [31] that gives the adversary additional advantage to upper bound the double-spending probability.

Notice that when we assume the rigged model in our protocol, there will not be any honest parallel proofs in the system as the parallel proofs are only generated due to the network delay honest miners experience. As a result, we need L sequential proofs for the block confirmation, which is no different from the L-block confirmation rule in Nakamoto consensus. For example, in Figure 3, the ledger ω_8^{h+1} is confirmed when block $h+2$ is created which has L sequential proofs. This is no different to asking L blocks to be mined on top of the block to be confirmed in Bitcoin. As the honest miners work for the adversarial favor during the δ and Δ delays, all honest proofs will be on different heights (AHPODH, see AHBODH in [31,32]).

Lemma A3 (Guo-Ren [31])**.** *Under AHPODH, if any attack succeeds in violating a transaction's safety, then the private mining attack also succeeds in violating that transaction's safety.*

We skip proving this result as it essentially follows the same steps as bounded Δ-delay model, by simply replacing blocks with proofs and Δ with δ in [31] (Lemma 5). We note that since our protocol allows two parallel proofs to be combined, the balanced attacks are less likely to succeed. Further, refs. [6–8] show that more than two proofs being mined within a δ time is extremely unlikely; hence, adversarial balanced attacks are less effective in our protocol. Thus, the rigged model assumption we make here and the resulting double-spending upper bound probability is an extremely pessimistic one.

Let $\lambda_B = \frac{1}{T_B}$ be the block generation rate where T_B denotes the block interarrival time. Let λ_P denote the proof generation rate. Let β denote the fraction of honest hashpower in the system ($\beta = 1 - \alpha$). As honest miners need L sequential proofs in the rigged model, we assume

$$\lambda_P = L\lambda_B, \tag{A19}$$

and

$$p = 1 - q = \beta e^{-\lambda_P \delta} > \frac{1}{2}. \tag{A20}$$

Let $S_L \sim NegBin(L, \beta)$ denote the sum of L i.i.d. geometric random variables with success probability β. Let $S_{\delta_L} \sim \sum_{i=1}^{L} Poisson(\lambda_P \delta)$ denote the sum of L i.i.d. Poisson random variables with rate $\lambda_P \delta$, whereas $S_{\Delta_2} \sim \sum_{i=1}^{2} Poisson(\lambda_P \Delta)$.

Lemma A4. *Under the private attack with the rigged model, the adversarial proofs generated in the confirmation interval are bounded by the distribution of*

$$S_L + S_{\delta_L} + S_{\Delta_2}. \tag{A21}$$

Proof. At τ, Δ time is needed for the proposed honest ledger of block h to be received by the honest miners, during which they work for the adversary. Hence, the total number of proofs created during the delay is $Poisson(\lambda_P \Delta)$. Next, before the arrival of each honest proof, the adversary creates $Geo(\beta)$ proofs, where $Geo(\beta)$ is a geometric random variable with success probability β. After the arrival, the honest proof suffers δ delay during which $Poisson(\lambda_P \delta)$ proofs are generated, all of which belonging to the adversary since honest miners work in its favor. Once L such proofs are generated, a proposed ledger for block $h + 1$ will be shared in Δ time for the block to be considered complete, during which honest miners work for the adversary. □

Note that before τ, the adversary might also build some initial lead for the attack [46], which we call the pre-mining attack. The analysis of such a lead normally does not require a rigged model since balanced attacks are only possible after τ [36]; however, we simply take the rigged model of [31], giving another additional power to the adversary here as the lead is insignificant. We also assume that the starting point of the adversarial lead is somewhere in the proof tree of block h, and the proof tree of block h is of infinite length.

This mild assumption essentially allows us to ignore the ledger delay of the block $h - 1$ and incorporate all proofs before τ into a single proof tree. In any case, it seems nearly impossible for the adversary to sustain a lead for more than L proofs such that the starting point of this lead is more than 1 block away. We could consider this additional detail by considering a more general ruin theoretical analysis introduced in [37]; however, it would merely change the insignificant digits in the final result. We refer the reader to [31,37] and simply note the following result.

Lemma A5. *The adversarial lead at τ can be upper bounded as*

$$M_{lead} \sim Geo\left(1 - \frac{q}{p}\right). \tag{A22}$$

Proof. We refer the reader to ([31], Section 5.1) for the details of the proof. Essentially, we can assume τ to be large, and the lead becomes the steady-state distribution birth–death process since it cannot go below zero, where the probability of a step forward is $q = 1 - p$, i.e., an adversarial arrival. □

https://doi.org/10.3390/cryptography10060080

At τ_c, when the honest ledger is confirmed, the adversary has $M_{lead}+S_L+S_{\delta_L}+S_{\Delta_2}$ proofs. If this is greater than L proofs the honest miners have for the confirmation of the honest ledger, then the adversary can simply release the private proofs and undo the honest ledger, i.e., double-spend. Otherwise, it still has a chance to catch up in the post-confirmation race.

Again, for simplicity, we assume the same rigged model of [31], even though it is not possible for the adversary to mount a balanced attack after τ_c anymore [36]. Making the same mild assumption of ignoring the ledger delays after block $h+2$ for simplicity and incorporating all the proofs to be mined after τ_c into a single proof tree, the number of proofs that the adversary can make up for its deficit is denoted as $M_{deficit} \sim M_{lead}$; see [31,32,36,37].

Thus, ignoring the Δ ledger delays before block $h-1$ and after $h+2$ (we still consider the δ delays of all proofs for all heights), we have the following theorem [37].

Theorem A1. *Given proof mining rate λ_P, honest fraction β, ledger delays of Δ, proof delays of δ, and 1-block confirmation rule, the confirmed ledger of block h with L votes cannot be discarded with probability greater than*

$$P(M_{lead} + S_L + S_{\delta_L} + S_{\Delta_2} + M_{deficit} \geq L). \tag{A23}$$

Appendix F. MDP Models for the Proposed Protocol

In this section, we assume the reader is familiar with the ϵ-optimal selfish mining strategies of [4]. We avoid presenting the same average reward ratio MDP model and the results. We simply refer the reader to the relevant parts of this paper ([4], Sections 2 and 4.1). Here, we assume the conventional system model used in selfish mining literature with α fraction of the hashpower controlled by a single colluding adversary which has γ network advantage, i.e., γ fraction of honest miners prefer the adversarial view when the adversary rushes to release its private branch that results in a fork.

To consider proof withholding attacks in our protocol, first note that there are two types of proofs, initiator and incremental proofs. We need to take the different proof types into account when considering an MDP model. Assume a last common proof (or two proofs if parallel) present in both the adversarial and honest chain. Right after this proof, when the adversary creates a private branch, the analysis depends on the first proof height that was withheld, until the adversary releases and overrides the honest branch or adopts the honest branch.

Let us assume the withholding attack, i.e., adversarial private branch, starts on an incremental proof. This means the previous proof height is a common proof between the adversarial and honest branches. Here, we have to consider what changes compared to [4] when a private branch is released by the adversary in order to understand how we can modify the MDP of [4] properly. If the adversary releases the incremental proofs right after,

1. Honest miners mine only 1 incremental proof:
   (a) If the released adversarial branch had 1 proof, the honest miners can combine the first adversarial proof and the honest proof as they are on the same proof height and share the same ancestor, and rewards will be split. In such a scenario, the public most aggregate work length essentially increases by one but also contains a fraction of adversarial rewards. Subsequently, the adversary either mines on its released branch that is 1 aggregate work behind the public most aggregate work or adopts the public most aggregate work branch which also contains a fractional adversarial reward.

(b) If the released adversarial branch had 2 proofs, a network split happens since the first honest and the first adversarial proof can be combined to have aggregate work of 2 proofs as well.
(c) If the released adversarial branch had more than 2 proofs, everyone accepts the adversarially released branch (override happens).

2. Honest miners mine more than 1 proof. Similar to the subcases above:
   (a) The adversary can override the honest branch if it has more than the honest proofs in its recently released branch.
   (b) If both the honest branch and the released adversarial branch have the same number of proofs, a network split of γ happens between honest miners.
   (c) If the adversary only has 1 proof in its released branch, the parallel combination of the first adversarial and honest proof can create a network split if the honest branch has 2 proofs.

First assume $L \to \infty$, i.e., virtually all proofs are incremental proofs. Notice that in this situation, i.e., when the proofs with the same heights can always be combined together to increase aggregate chain work and rewards are split, we only need to reconfigure the MDP model of [4] to analyze the resilience against incentive attacks where we can ignore the concept of blocks since each proof height gets the same reward independent of the blocks, and rewards are split between parallel proofs of the same height if a proof height contains parallel proofs. Hence, we modify the average reward ratio MDP of [4] to incorporate the effect of the reward splitting, which we call reward split MDP.

We consider the following modified objective function representing the *relative payoff*,

$$\alpha_{rel} = \frac{\sum_t A_t - \frac{A'_t}{2}}{\sum_t A_t + H_t - A'_t}, \tag{A24}$$

where A_t and H_t are the number of the adversarial and honest proofs that make it into the heaviest chain at step t, and A'_t is the number of proof heights that contain parallel proofs, i.e., those proofs whose coinbase rewards are split between adversarial and honest parallel proofs. As in [3,4,47,48], we consider γ network influence of the adversary. In other words, whenever honest miners mine a proof, the adversary can match it if it has a proof with the same aggregate work, and the honest network will split between the two if these two proofs cannot be combined, i.e., when they do not have the same parents.

We consider the same action space as in [4] but expand the state space to $(a, h, fork, p)$, with p representing if parallel proofs exist. The extra parameter p represents the parallel proofs at each step and can be restricted to $p \in \{0, 1\}$ since parallel proofs only happen if the adversary and honest miners work on top of the same proof/proofs of the previous proof height, which implies an *adopt* happened in the last step. When $(a, h, fork) = (1, 1, relevant)$, a match action will make $p = 1$ since the role of γ becomes irrelevant as both proofs can be combined by honest miners, and $h + p$ becomes the aggregate work of the honest chain that the adversary has to race with if it does not adopt the combination of the two parallel proofs immediately. Finally, we modify the reward function of [4] as

$$\omega_\rho(x, y, p) = (1 - \rho)x - \rho y + (-0.5 + \rho)p, \tag{A25}$$

according to (A24), where x and y are the number of adversarial and honest proofs that make it into the heaviest chain at the end of each step, and $p \in \{0, 1\}$ represents if the reward of the first proof is split between parallel proofs.

Next, when L is finite, to consider the effect of the initiator proofs, assume the withholding attack, i.e., adversarial private branch, starts on an initiator proof. This means that the previous proof height is the last proof height of the previous block. This in turn implies that even though the previous proofs of the previous proof height is a common ancestor for both the adversarial and honest branches, the initiators can only be combined if they vote on the same ledger for the previous block. Then,

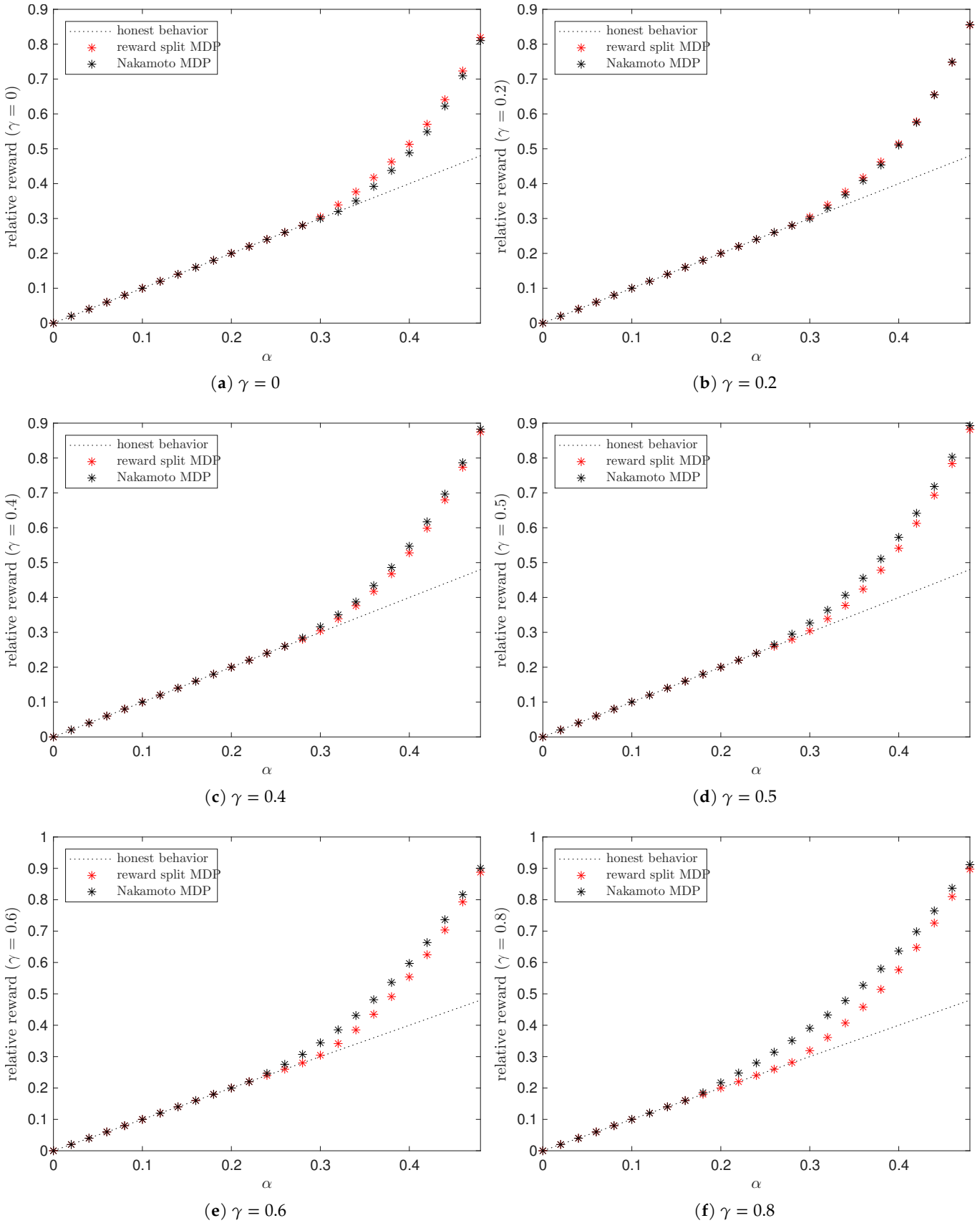
1. If the honest and adversarial initiators vote on the same valid ledger (does not matter if the ledger is created by the adversary as long as it is valid), the situation is no different from the previous analysis with incremental heights.
2. If the adversary and honest miners vote on different ledgers, the adversarial initiator cannot be combined with the honest initiator. Thus, until an override or adopt action happens, no parallel proofs can be combined, and the effect of p introduced in our model is suspended until the next private branch. In other words, the system would temporarily have the same rules as Nakamoto consensus, and the MDP model of [4] could be used directly. We can expand the state space to $(a, h, fork, p, l)$ with l keeping track of the last common proof height in order to check if this situation happens.

To consider if this last case of expanded state space helps the adversary or not, we can first compare the relative rewards of the MDP model of [4] and the reward split MDP we explained earlier. If the reward split MDP results in higher relative rewards for an (α, γ) pair, it makes no sense for the adversary to withhold the ledger with the most fees. On the other hand, if the reward split MDP results in less relative rewards, then it is in the interest of the adversary to launch this type of attack for initiator heights if it has a ledger offering the most fees.

We assume the adversary can always create a ledger with the most fees offered, which is more than what the adversary could achieve. We implement this complicated situation where we have a state space of the form $(a, h, fork, p, l)$ with l keeping track of the last common proof height. Even in the scenario where there is no reward loss, the result is approximately a weighted average between the reward ratio of the work in [4] (with approximate weight $1/L$) and the reward split MDP (with approximate weight $(L-1)/L$). This is expected since the last case arises in approximately $\frac{1}{L}$ of the proof heights, i.e., the initiator heights. On the other hand, for $\gamma < 0.3$, the result is the same as the reward ratio of reward split MDP since it is more profitable to have parallel proofs in this case. As the gain is minor even when the adversarial losses are ignored, we omit the simulation with expanded state space, as the computational burden becomes very large with increasing L. In any case, initiator proof heights are only $\frac{1}{L}$ of the total proof heights in the system.

Simulations

To evaluate the performance of the reward split MDP and compare it with the Nakamoto consensus MDP of [4], we run the MDP models for $\gamma = 0, 0.2, 0.4, 0.5, 0.6, 0.8$, and α values range from 0.02 to 0.48 with 0.02 spacing between them with 10^{-5} precision. From Figure A5, it is clear that reward splitting helps the adversary when γ is low, as matching always gives a half reward when $(a, h, fork) = (1, 1, relevant)$, whereas the adversarial relative reward in Nakamoto consensus increases with increasing γ since matching succeeds more often. Thus, even when $\gamma = 0.8$, the profitability threshold in our protocol stays around $\alpha \approx 0.28$.



(**a**) $\gamma = 0$ (**b**) $\gamma = 0.2$

(**c**) $\gamma = 0.4$ (**d**) $\gamma = 0.5$

(**e**) $\gamma = 0.6$ (**f**) $\gamma = 0.8$

Figure A5. Reward split MDP vs. Nakamoto MDP [4].

https://doi.org/10.3390/cryptography10060080

Appendix G. Transaction Split Rule

Here, we consider an attacker that wishes to increase its rewards from the transactions fees by picking its own proposed ledger in the previous block instead of the ledger offering the most fees. We show the conditions for the attack to be profitable. Further, we show that such an attack does not have a big impact on the transaction fee rewards received by other miners. For the simplicity of the analysis, here we consider zero network delay. Even though non-zero network delay can affect the transaction fee reward distribution in the case of proof heights that contain parallel proofs, our ultimate goal is to show that the adversarial advantage in this type of attack decreases as r gets smaller which is independent of whether there are parallel proof heights or not.

More concretely, assume a miner M holds α fraction of the hashpower in the system. Further, assume that L proofs are created for the last block height h and the miner M was lucky enough to create at least one proof for block height h (hence, a proposed ledger for next block). Let F_M denote the maximum offered transaction fee by the miner M's proposed ledger, whereas $F_{M'}$ denotes the maximum offered transaction fee overall by any proposed ledger for the next block height. Ignoring the small δ delays of the proofs, we have the following result.

Lemma A6. *It is in the interest of miner M to create an initiator proof referring to its own proposed ledger instead of the ledger offering the most fees if*

$$\frac{F_M}{F_{M'}} \geq 1 - \frac{r}{r + \frac{1-r}{L}(1 + (L-1)\alpha)}. \tag{A26}$$

Proof. Assume the miner picks its own proposed ledger while trying to mine an initiator proof for the next block height:

1. If it is successful to mine an initiator referring to the ledger offering F_M fees before others, which happens with probability α, it shares the initiator immediately. Other miners accept the initiator of M and mine on top of it since it is the chain with the heaviest work.
2. If others mine an initiator before him, referring to the ledger offering $F_{M'}$ fees, which happens with probability $1-\alpha$, the miner M accepts the ledger and mines on top of it as everyone else.

For the rest of the $L-1$ incremental proofs after the initiator, the miner M creates $(L-1)\alpha$ more proofs on average. The expected transaction fee rewards is

$$F_M \cdot \alpha\left(r + \frac{1-r}{L} + \frac{1-r}{L}(L-1)\alpha\right) + F_{M'} \cdot (1-\alpha)\left(\frac{1-r}{L}(L-1)\alpha\right). \tag{A27}$$

On the other hand if it simply tries to mine an initiator proof referring to the ledger offering $F_{M'}$ fees, the expected transaction fees rewards is

$$F_{M'}(1-r) \cdot \alpha. \tag{A28}$$

Comparing the two strategies, we get (A26). □

Notice that the threshold in (A26) is increasing in α and decreasing in r. Hence, when the attacker has more fraction of hashpower, its own ledger has to offer similar fees as the ledger offering the most fees. For example, if we picked an equal split rule between the proposer and voters, i.e., $r = \frac{1}{C(\omega_L^h)+1}$, (A26) becomes

$$\frac{F_M}{F_{M'}} \geq \frac{(L-1)\alpha + 1}{(L-1)\alpha + 2}. \tag{A29}$$

https://doi.org/10.3390/cryptography10060080

If $\alpha = 0.1$ and $L = 50$, the threshold in (A29) becomes 0.85, i.e., the ledger of M should offer at least $0.85F_{M'}$; otherwise, the attack is not profitable. On the other hand, with an equal split rule as $\alpha \to 0$, if M is lucky enough to propose a ledger, it is in its own interest to mine an initiator referring to its own ledger as long as it offers at least half the rewards compared to the ledger offering the most fees. In other words, the ledger of M should offer at least $0.5F_{M'}$. Hence, we do not pick equal split rule.

If we pick smaller values for r compared to equal split rule, for example we pick $r = \frac{1}{2L}$, for $L = 50$, if the ledger of M offers at least $0.85F_{M'}$, it needs to hold less than $\alpha \leq 0.038$ fraction of hashpower in the system. Hence, as we decrease r, the attack becomes unprofitable even for a small miner. On the other hand, even when it is profitable, as long as r is small, the transaction fee rewards for the other miners that create incrementals on top of this ledger do not lose much, for example, with $r = \frac{1}{2L}$ and $\alpha = 0.1$, they only lose $0.08F_{M'}\frac{1-r}{L}$.

We also note that the above analysis assumes that the attacker was lucky enough to mine a proof with a proposed ledger for the previous block. Even though the threshold in (A26) decreases as α decreases, the frequency of such a lucky situation also decreases. Finally, when network delays are present, the results above can be trivially verified by replacing L's in the right side of (A26) with the expected number of proof heights, $C(\omega_L)$, which does not change the behavior of the threshold with respect to r and α.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 26 October 2026).
2. Rosenfeld, M. Analysis of Hashrate-Based Double Spending. *arXiv* **2014**, arXiv:1402.2009. [CrossRef]
3. Eyal, I.; Sirer, E.G. Majority is Not Enough: Bitcoin Mining is Vulnerable. *Commun. ACM* **2018**, *61*, 95–102.
4. Sapirshtein, A.; Sompolinsky, Y.; Zohar, A. *Optimal Selfish Mining Strategies in Bitcoin*; Springer FC: Berlin/Heidelberg, Germany, 2017.
5. Swanson, E. Available online: http://alloscomp.com/bitcoin/calculator (accessed on 18 January 2025).
6. Chen, C.; Chen, X.; Yu, J.; Wu, W.; Wu, D. Impact of Temporary Fork on the Evolution of Mining Pools in Blockchain Networks: An Evolutionary Game Analysis. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 400–418. [CrossRef]
7. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. Modeling the Impact of Network Connectivity on Consensus Security of Proof-of-Work Blockchain. In *Proceedings of the IEEE INFOCOM, July 2020*; IEEE: Piscataway, NJ, USA, 2020.
8. Sakurai, A.; Shudo, K. Model-Based Calculation Method of Mining Fairness in Blockchain. *IEEE Open J. Comput. Soc.* **2026**, *7*, 129–141. [CrossRef]
9. Courtois, N.T.; Bahack, L. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. *arXiv* **2014**, arXiv:1402.1718. [CrossRef]
10. Eyal, I. The Miner's Dilemma. In *Proceedings of the IEEE S&P, May 2015*; IEEE: Piscataway, NJ, USA, 2015.
11. Kwon, Y.; Kim, D.; Son, Y.; Vasserman, E.; Kim, Y. Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin. In *Proceedings of the ACM SIGSAC CCS, October 2017*; ACM: New York, NY, USA, 2017.
12. Gao, S.; Li, Z.; Peng, Z.; Xiao, B. Power Adjusting and Bribery Racing: Novel Mining Attacks in the Bitcoin System. In *Proceedings of the ACM SIGSAC CCS, November 2019*; ACM: New York, NY, USA, 2019.
13. Pass, R.; Shi, E. FruitChains: A Fair Blockchain. In *Proceedings of the ACM PODC, July 2017*; ACM: New York, NY, USA, 2017.
14. Bagaria, V.; Kannan, S.; Tse, D.; Fanti, G.; Viswanath, P. Prism: Deconstructing the Blockchain to Approach Physical Limits. In *Proceedings of the ACM SIGSAC CCS*; ACM: New York, NY, USA, 2019.
15. Eyal, I.; Gencer, A.E.; Sirer, E.G.; van Renesse, R. Bitcoin-NG: A Scalable Blockchain Protocol. In *Proceedings of the USENIX NSDI, March 2016*; USENIX Association: Berkeley, CA, USA, 2016.
16. Sompolinsky, Y.; Zohar, A. Secure High-Rate Transaction Processing in Bitcoin. In Proceedings of the Financial Cryptography, San Juan, Puerto Rico, 26–30 January 2015.
17. Zhang, R.; Preneel, B. Publish or Perish: A Backward-Compatible Defense Against Selfish Mining in Bitcoin. In *Proceedings of the CT-RSA, February 2017*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 277–292.
18. Lerner, S.D. DECOR + HOP: A Scalable Blockchain Protocol. 2015. Available online: http://scalingbitcoin.org/papers/DECOR-HOP.pdf (accessed on 26 October 2026).
19. Zhang, R.; Preneel, B. Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security. In *Proceedings of the IEEE S&P*; IEEE: Piscataway, NJ, USA, 2019.

20. Sompolinsky, Y.; Wyborski, S.; Zohar, A. PHANTOM GHOSTDAG: A scalable generalization of Nakamoto consensus: September 2, 2021. In *Proceedings of the ACM AFT, September 2021*; ACM: New York, NY, USA, 2021.
21. Sompolinsky, Y.; Lewenberg, Y.; Zohar, A. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *Cryptol. Eprint Arch.* **2016**, *Preprint*.
22. Sarkar, P. A New Blockchain Proposal Supporting Multi-Stage Proof-of-Work. *J. Blockchain Res.* **2022**, *1*, 37–50. [CrossRef]
23. Hu, J.; Jiang, Z.; Xu, C. Greedy-Mine: A Profitable Mining Attack Strategy in Bitcoin-NG. *Int. J. Intell. Syst.* **2024**, *2024*, 9998126. [CrossRef]
24. Yin, J.; Wang, C.; Zhang, Z.; Liu, J. Revisiting the Incentive Mechanism of Bitcoin-NG. In *Proceedings of the Springer ACISP*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 706–719.
25. D'Arco, P.; Ansaroudi, Z.E.; Mogavero, F. Multi-stage Proof-of-Works: Properties and vulnerabilities. *Theor. Comput. Sci.* **2023**, *976*, 114108. [CrossRef]
26. Chang, D.; Hasan, M.; Jain, P. Spy Based Analysis of Selfish Mining Attack on Multi-Stage Blockchain. *Cryptol. Eprint Arch.* **2019**, *Preprint*. Available online: https://eprint.iacr.org/2019/1327 (accessed on 26 October 2026).
27. Keller, P.; Böhme, R. Parallel Proof-of-Work with Concrete Bounds. In *Proceedings of the ACM AFT, September 2022*; ACM: New York, NY, USA, 2022.
28. Bissias, G.; Levine, B.N. Bobtail: Improved Blockchain Security with Low-Variance Mining. In Proceedings of the NDSS Symposium, San Diego, CA, USA, 23–26 February 2020.
29. Keller, P.; Glickenhaus, B.; Bissias, G.; Griffith, G. Tailstorm: A Secure and Fair Blockchain for Cash Transactions. In *Proceedings of the ACM AFT, October 2023*; ACM: New York, NY, USA, 2023.
30. Keller, P. Parallel Proof-of-Work with DAG-Style Voting and Targeted Reward Discounting. *arXiv* **2023**, arXiv:2312.03111. [CrossRef]
31. Guo, D.; Ren, L. Bitcoin's Latency–Security Analysis Made Simple. In *Proceedings of the ACM AFT, September 2023*; ACM: New York, NY, USA, 2023.
32. Doger, M.; Ulukus, S. Refined Bitcoin Security-Latency Under Network Delay. *IEEE Trans. Inf. Theory* **2025**, *71*, 3038–3047. [CrossRef]
33. Bazzanella, D.; Gangemi, A. Bitcoin: A new proof-of-work system with reduced variance. *Financ. Innov.* **2023**, *9*, 91. [CrossRef]
34. Anada, H. Theoretical Analysis of Serial/Parallel Variations of Hash-Mining for Smaller Variance of Confirmation Time. In *Proceedings of the IEEE ICCE, August 2024*; IEEE: Piscataway, NJ, USA, 2024.
35. Dembo, A.; Kannan, S.; Tas, E.; Tse, D.; Viswanath, P.; Wang, X.; Zeitouni, O. Everything is a Race and Nakamoto Always Wins. In *Proceedings of the ACM CCS, November 2020*; ACM: New York, NY, USA, 2020.
36. Cao, S.J.; Guo, D. Security, Latency, and Throughput of Proof-of-Work Nakamoto Consensus. *IEEE Trans. Inf. Theory* **2025**, *71*, 4708–4731. [CrossRef]
37. Doger, M.; Ulukus, S.; Akar, N. PoW Security-Latency Under Random Delays and the Effect of Transaction Fees. In *Proceedings of the IEEE ITW, November 2024*; IEEE: Piscataway, NJ, USA, 2024.
38. DSN Bitcoin Monitoring. Available online: https://dsn.tm.kit.edu/bitcoin/ (accessed on 11 February 2026).
39. Grunspan, C.; Pérez-Marco, R. Block withholding resilience. *Digit. Finance* **2025**, *7*, 43–60. [CrossRef]
40. Miller, A. Feather-Forks: Enforcing a Blacklist with Sub-50% Hash Power. 2013. Available online: http://bitcointalk.org/index.php?topic=312668.0 (accessed on 26 October 2026).
41. Finkbeiner, C.; Najd, M.E.; Guskind, J.; Almashaqbeh, G. SoK: Time to be Selfless?! Demystifying the Landscape of Selfish Mining Strategies and Models. In *Proceedings of the IEEE ICBC, June 2025*; IEEE: Piscataway, NJ, USA, 2025.
42. Negy, K.A.; Rizun, P.R.; Sirer, E.G. Selfish Mining Re-Examined. In *Proceedings of the Springer FC*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 61–78.
43. Grunspan, C.; Pérez-Marco, R. On profitability of selfish mining. *arXiv* **2019**, arXiv:1805.08281. [CrossRef]
44. Grunspan, C.; Pérez-Marco, R. Profit Lag and Alternate Network Mining. In *Proceedings of the Springer MARBLE*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 115–132.
45. Mankiw, N.G. *Macroeconomics*, 7th Edition; Worth Publishers: New York, NY, USA, 2010.
46. Sompolinsky, Y.; Zohar, A. Bitcoin's Security Model Revisited. *arXiv* **2016**, arXiv:1605.09193. [CrossRef]
47. Nayak, K.; Kumar, S.; Miller, A.; Shi, E. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In *Proceedings of the IEEE EuroS&P, March 2016*; IEEE: Piscataway, NJ, USA, 2016.
48. Zur, R.; Eyal, I.; Tamar, A. Efficient MDP Analysis for Selfish-Mining in Blockchains. In *Proceedings of the ACM AFT, October 2020*; ACM: New York, NY, USA, 2020.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.