Temporary Block Withholding Attacks on Filecoin's Expected Consensus

Tong Cao
Kunyao Academy
tongtian@kunyaokeji.com

Xin Li
Kunyao Academy
lvwang@kunyaokeji.com

Abstract

Filecoin is the most impactful storage-oriented cryptocurrency. In this system, miners dedicate their storage space to the network and verify transactions to earn rewards. Nowadays, Filecoin's network capacity has surpassed 15 exbibytes.

In this paper, we propose three temporary block withholding attacks to challenge Filecoin's expected consensus (EC). Specifically, we first deconstruct EC following old-fashioned methods (which have been widely developed since 2009) to analyze the advantages and disadvantages of EC's design. We then present three temporary block withholding schemes by leveraging the shortcomings of EC. We build Markov Decision Process (MDP) models for the three attacks to calculate the adversary's gains. We develop Monte Carlo simulators to mimic the mining strategies of the adversary and other miners and indicate the impacts of the three attacks on expectation. As a result, we show that our three attacks have significant impacts on Filecoin's mining fairness and transaction throughput. For instance, when honest miners who control more than half the global storage power update their tipsets (i.e., the collection of blocks in the same epoch that have the same parents) after the default transmission cutoff time, an adversary with 1% of the global storage power is able to launch temporary block withholding attacks without a loss in revenue, which could affect Filecoin's security and performance. Finally, we discuss the implications of our attacks and propose several countermeasures to mitigate them.

1 Introduction

Inspired by Bitcoin [26], many cryptocurrencies have been created to not only solve the Byzantine Generals problem [22] in the asynchronous and permissionless network, but also accomplish other goals, such as supporting smart contract [40], preserving user privacy [24, 30], and decentralizing storage [2,39,41]. In the past few years, many emerging technologies have been created and developed in these orientations.

Filecoin is not only a decentralized ledger, but also the leading decentralized storage platform. By combining an incentive mechanism (i.e., coin reward) and a low-level storage verification mechanism (i.e., proof of storage [17]), Filecoin has successfully integrated more than 15 exbibytes of decentralized storage. Broadly speaking, the core design of Filecoin's consensus layer is called Expected Consensus (EC), which comprises many mechanisms and protocols. EC mainly includes the Proof-of-Spacetime (PoSt) mechanism, the Directed Acyclic Graph (DAG)-based ledger extension mechanism (i.e., more than one block of transactions can be generated and confirmed at each round), and the Proof-of-Stake (PoS)-based leader selection (i.e., the probability that the participant can be elected as the leader to generate blocks is based on her storage power).

Despite the success of Filecoin in the last few years, there is a lack of formal analysis of its consensus layer. In this paper, we first conduct a novel study analyzing EC's unique design, with a focus on its impact on system security and performance. We follow traditional methods to evaluate EC's security bounds [4, 11, 18, 26, 31, 34]. We then propose three temporary block-withholding (TBW) attacks to challenge EC's security and performance. Our first attack, *TBW Attack 1.0*, anchors the threshold of breaking EC's mining fairness in the perfect implementation where every participant but the adversary obeys the default setting. Our second attack, *TBW Attack 2.0*, splits the honest miners and creates conflict between them by leveraging some miners' rationality. As a consequence, the adversary is able to lower the threshold for launching an attack. Our third attack, *TBW Attack 3.0*, is more threatening. It executes an action, **Front Epoch Prediction**, to further reduce the threshold of breaking the system's mining fairness. For instance, when more than half of the network storage is controlled by rational honest miners who update their tipsets after the default transmission cutoff time, an adversary with 1% of the network storage power is able to launch an attack, which is rare in proof-of-work (PoW) based blockchains. To evaluate the impacts of our three attacks, we use two well-established metrics, adversary's revenue share and stale block rate. We demonstrate that TBW attacks are threatening to EC's security and performance.

We make the following contributions:

- To the best of our knowledge, this is the first work to analyze Filecoin's consensus layer in a generic way by considering the properties that have been widely analyzed in PoW-based cryptocurrencies, which helps to improve our understanding of EC.
- Our three attacks are novel in Filecoin. With theoretical and experimental analyses, we indicate the impacts of TBW attacks in this novel decentralized consensus protocol (EC) and point out the vulnerability of EC's design.
- We present insights about the impacts of such attacks and provide mitigating countermeasures.

Disclosure. We have disclosed our findings to Protocol Labs. The time delay between Filecoin network and drand network has been shortened to mitigate *TBW Attack 3.0*[1]. The default cutoff time has been increased to the 15^{th} second of the epoch to mitigate *TBW Attack 2.0*.

2 Related Work

Block withholding was first proposed by M. Rosenfeld [33] in 2011. It was mainly recognized as the dishonest behaviors of sub miners in Bitcoin's mining pools. At that time, two types of block withholding behaviors were defined: "sabotage" and "lie in wait". The former means permanent block withholding, in which the sub miners never release the blocks in the target pool but enjoy the share from others, while the latter represents the temporary block withholding in which sub miners postpone block submission in order to increase their revenue.

Selfish mining [11] was proposed by Eyal and Sirer in 2013. Selfish mining considers more complicated temporary block withholding strategies (i.e., the adversary temporarily withholds leading blocks and felicitously releases them) of mining pools in Bitcoin. With different actions of selfish mining, the adversary with sufficient hash power may compromise Bitcoin's security.

For brevity, we consider only temporary block withholding of pools (or solo miners). Analyzing block withholding of sub miners in a pool is beyond the scope of the paper.

In the past decade, such temporary block withholding attacks have been widely studied in PoW blockchains [1, 11, 13, 18, 26, 27, 34]. Such attacks normally have significant impacts on system security and performance. Preventing temporary block withholding attacks has become the key challenge in designing decentralized consensus protocols [3, 20, 23, 32]. Recently, several temporary block withholding attacks have been proposed in Ethereum's PoS beacon chain [29, 35], which extend the scope of impact of such attacks.

[1] https://github.com/filecoin-project/lotus/pull/8606

3 Expected Consensus

This section introduces background knowledge about EC. We first build a model to formalize the legder's structure of EC, and then define a novel algorithm: Greedy Heaviest Observed Series-parallel Sub-graph (GHOSS) to generalize EC's fork selection rule.

3.1 Ledger's Structure

Ledger's structure is the core of cryptocurrencies, which is normally built and represented by a Directed Acyclic Graph (DAG). The consensus protocols' goal is actually to define the DAG's extension rule for every participant, which enables a common sub-graph (e.g., a chain, a sub-tree, or a sub-graph) in every participant's DAG. In this part, we use the model of series-parallel graph [10, 19] to formalize EC's ledger's structure. As a result, we can compare EC with previous consensus protocols in the model, which could help us to better understand EC's novelty.

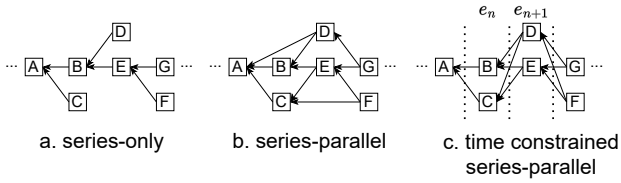


Figure 1: Illustration of three types of DAG in series-parallel model.

3.1.1 Formalization

Broadly speaking, the ledger extension rule of cryptocurrencies can be categorized under two types: single reference (i.e., the new block can only be linked with a single previous block), and multiple references (i.e., the new block can be linked with multiple previous blocks). In the former type, it only allows *series composition*[2], thus, it derives the series-only DAG as shown in Fig. 1.a (i.e., there only exist some disjoint sub two-terminal graphs: G_x, G_y that G_x's sink is G_y's source). Most of cryptocurrencies' ledgers (e.g., Bitcoin, Ethereum) are series-only DAG. In the latter type, due to the multiple references rule, *parallel composition* can be made, which derives the series-parallel DAG as shown in Fig. 1.b (i.e., there not only exist some disjoint sub two-terminal graphs: G_x, G_y that G_x's sink is G_y's source, but also exist some disjoint sub two-terminal graphs: G_m, G_n that have the same source and sink). Some cryptocurrencies's ledgers (e.g., IOTA [37]) are series-parallel.

The ledger's structure of Filecoin's EC can be defined as a time constrained series-parallel DAG, as shown in Fig. 1.c.

[2] Please see [10, 19] for more details about definitions of series composition and parallel composition.

Precisely, the multiple references rule is time constrained, meaning, the new block can only be linked with the previous blocks who have the some parents. This design is quite similar with *g-greedy* protocol [12]. It is very difficult to say which type of DAG should be used to design a cryptocurrency. Our intuition here is, the ledger's structure of EC is a novel design, which provides a possibility on designing cryptocurrency. Our main goal here is to explain how different it is so that we can explain our TBW attacks more clear, analyzing EC's other properties would be the future work.

3.1.2 Block Structure

Each block contains a header and a body[3]. Here, we emphasize that blocks are linked via content identifier (CID). In EC, each block has an unique CID, which is calculated through the hash function:

$$CID(B_{e_i,j}) = Hash(CID(B_{e_{i-1},1}), CID(B_{e_{i-1},2}), ..., CID(B_{e_{i-1},k}), contents) \quad (1)$$

where, $B_{e_{i-1},1}, B_{e_{i-1},2}, ..., B_{e_{i-1},k}$ note $B_{e_i,j}$'s parent blocks, and *contents* represent other messages included in the block. As long as the miner is elected leader, s/he can produce the block. To do so, the miner needs to assemble all required messages to execute the hash function (Eq. 1) to calculate the block's Content Identifier. Afterward, the leader disseminates the block's CID and corresponding signed messages to the network so that other participants can verify if the block is valid (i.e., check the leader's legality and verify the block's contents).

3.1.3 *tipset*

In EC, a *tipset* is defined as a collection of blocks that have the same parent blocks, which is used to represent the accumulated weights of the time constrained sub-graphs in each epoch. The main idea of *tipset* is to aggregate the blocks in each epoch, which is very closed to the aggregators in Ethereum 2.0[4].

3.2 Leader Election

In EC, miners contribute their storage space to the network. In turn, their storage power is verified by a Proof-of-Storage mechanism [16, 17] and recorded in a power table (which is built to maintain membership; see Filecoin's specifications for more details[5]). Each round, miners request the randomness beacon from drand (an independent network that generates trustworthy random numbers periodically; see Filecoin's specifications for more details[6]) as the input of an election function 3. Let RB_i be the i^{th} randomness beacon. To decide whether they are elected, miners need to execute, first, a Verifiable Random function (VRF) $vrf.digest_i = VRF(RB_i)$[7]; and second, an SHA256 function $H(vrf.digest_i)$ for the purpose of calculating probability. The miner is elected as the leader only when the following condition is satisfied:

$$\frac{H(vrf.digest_i)}{2^{256}} < 1 - e^{-\mu * \beta} \quad (2)$$

where μ denotes the expected number of successful attempts to solve the puzzle that can be selected in each epoch. (Each successful attempt is counted as a winning event, which is named *WinCount*. This is different from normal blockchains, where a successful attempt represents a block in general). The right side of inequality 2 represents the difficulty of the miner being elected, which is determined by that miner's storage power share β. As long as the miner is elected leader, she is able to obtain more than one *WinCount*. After the miner is elected, the difficulty of obtaining *WinCounts* increases recursively, which is shown in inequiality 3.

$$\frac{H(vrf.digest_i)}{2^{256}} < 1 - \sum_{j=0}^{j=n-1} \left(\frac{(\mu * \beta)^j * e^{-\mu * \beta}}{j!}\right), n \geq 1 \quad (3)$$

where n denotes the number of *WinCounts* the miner can obtain. It is obvious that the mining difficulty increases as n increases.

3.3 Weighting Function

The weighting function is made to represent the accumulated weights of the time constrained sub-graphs. Broadly speaking, the weights of time constrained sub-graphs (or *tipsets*) are affected by two factors: accumulated storage power and blocks in the sub-graphs. In general, the storage power remains stable during a short time period. Thus, we mainly consider the impact of accumulated blocks (or *WinCounts*) in this paper. Based on Filecoin's official specification, the weighing function is defined as follows:

$$w(T_{e_t,j}) = w(T_{e_{t-1},p}) + (wpf(T_{e_t,j}) + wpf(T_{e_t,j}) * n_{wc}(T_{e_t,j}) * \frac{\tau}{\mu}) * 2^8 \quad (4)$$

where $w(T_{e_t,j})$ and $w(T_{e_{t-1},p})$ represent the weight of the j^{th} tipset in the t^{th} epoch and the weight of the p^{th} tipset in the $(t-1)^{th}$ epoch, and $T_{e_{t-1},p}$ is the parent of $T_{e_t,j}$. $wpf(T_{e_t,j})$ denotes the factor of the total storage power referenced in $T_{e_t,j}$, which is equal to the binary length of the total storage power minus 1. $wpf(T_{e_t,j})$ represents *WPowerFactor*, defined

[3] https://spec.filecoin.io/#section-systems.filecoin_blockchain

[4] https://stonecoldpat.substack.com/p/role-of-aggregators-and-subcommittees

[5] https://spec.filecoin.io/#section-glossary.power-table

[6] https://spec.filecoin.io/#section-libraries.drand

[7] https://spec.filecoin.io/#section-glossary.vrf

in EC[8]). $n_{wc}(T_{e_1,j})$ denotes the number of *WinCount* in $T_{e_t,j}$, τ is the ratio between *WPowerFactor* and *WBlocksFactor*, and μ denotes the number of expected *WinCount* at each round.

3.4 Fork Selection Rule

Due to the network delay, churn, or some uncertainties, multiple *tipsets* (i.e., multiple sub-graphs) can be composited, which causes fork. As shown in Fig. 2, block D,E,H have the same parents (which are block B,C), so they can be referenced by block J,G,F. However, due to some reasons, the proposer of block G was not able to receive block H. Thus, block G would not be combined with block J,F. In the next step, the new block proposer would face a choice: mine on block J,F or block G.

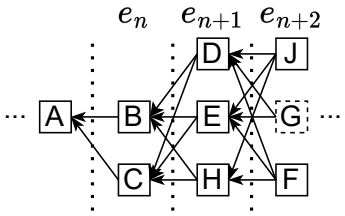


Figure 2: Example of fork in EC. The solid rectangle denotes the block that would be included in the main sub-graph as valid blocks. The dashed rectangle denotes the orphaned block.

Before we formalize EC's fork selection rule, let's have a look at the relationship between *tipset* and sub-graph. In practice, *tipset* is convenient to aggregate the blocks, and the link between *tipsets* is easy to be used to calculate the weights. However, it is very confusing for people who do not know Filecoin. The common questions, such as, what are *tipsets*? what is the chain of *tipsets*? what is it different with chain of blocks?, are frequently asked.

To better explain EC, we redefine *tipset* in EC's DAG: a *tipset* is a set of sinks of some sub-trees who follow the time constraint rule (i.e., they have the same parents). As shown in Fig. 2, in epoch e_n, *tipset* that includes block B,C is the set of sinks of *sub-tree* that includes block A,B and *sub-tree* that includes block A,C. It is obvious that *sub-graph* that include block A,B,C was selected in e_n. Consequently, the weights of *tipset* are the accumulated weights of the selected *sub-graph*.

Greedy Heaviest Observed Series-parallel Sub-graph (GHOSS). In graph theory, EC's fork selection rule can be summarized as follows:

- first, each participant initializes the genesis block;
- second, each participant selects the heaviest observed sub-graph that consists of several sub-trees following the time constraint (i.e., the sinks of sub-trees have the same parents).

Following this rule, we can find in Fig. 2, the selected sub-graphs in epoch e_n, e_{n+1}, e_{n+2} are sub-graph that consists of block A,B,C, sub-graph that consists of block A,B,C,D,E,H, and sub-graph that consists of block A,B,C,D,E,H,J,F. With this definition, we can find, the former question (should the next block proposers mine on block J,F or block G?) is actually equal to how to select the sub-graph. Let each block to have the same weight, it is easy to find sub-graph that consists of A,B,C,D,E,H,J,F is the heaviest one, thus, block J,F would be included by miners, while block G is orphaned.

4 Warm Up: Deconstructing EC Following Old-Fashioned Methods

This section analyzes EC through an angle that considers well-defined properties and challenges of PoW) based cryptocurrencies. For comparison, the advantages and disadvantages of EC are discussed. We provide an overview of leveraging EC's "Achilles' heel" to launch temporary block-withholding attacks at the end of this section.

4.1 Gambler's Ruin Problem on EC

Probabilistic consensus protocols have been widely studied in the past decade. They are keys to solving the Byzantine Generals' problem in asynchronous and permissionless networks. Nakamoto S. was the first to use the probabilistic Gambler's Ruin model to prove the weak/eventual consistency assuming an honest majority [26]. Precisely, the adversary with less than 50% of the global hash power who withholds her blocks would eventually fail to compete with the public chain.

In EC, the Gambler's Ruin model is more complicated than Bitcoin's mainly because 1) the round of competition is defined by epoch, not by the event when the next block is generated; and 2) the adversary's success at each round is based on not only the adversary's newly discovered blocks, but also the honest miners' newly discovered blocks.

Let α be the adversary's storage power; μ be the expected number of *WinCounts* that can be generated at each epoch; $X,Y \in [0,n]$ be the number of *WinCounts* that the adversary and the honest miners can obtain at an epoch; $p_{a.win}, p_{h.win}$ be the adversary and honest miners' respective success rates; and p_{draw} be the probability of the draw. Then we can obtain:

[8] https://spec.filecoin.io/#section-algorithms.expected_consensus.chain-selection

$$
\begin{aligned}
p_{a.win} &= \sum_{X>Y} \left(\frac{(\mu*\alpha)^X e^{-\mu*\alpha}}{X!} * \frac{(\mu*(1-\alpha))^Y e^{-\mu*(1-\alpha)}}{Y!}\right) \\
p_{h.win} &= \sum_{X<Y} \left(\frac{(\mu*\alpha)^X e^{-\mu*\alpha}}{X!} * \frac{(\mu*(1-\alpha))^Y e^{-\mu*(1-\alpha)}}{Y!}\right) \\
p_{draw} &= \sum_{X=Y} \left(\frac{(\mu*\alpha)^X e^{-\mu*\alpha}}{X!} * \frac{(\mu*(1-\alpha))^Y e^{-\mu*(1-\alpha)}}{Y!}\right)
\end{aligned} \tag{5}
$$

To solve the fork when two chains have equal weight, each participant selects the chain of *tipsets* with the smallest final *ElectionProof ticket*, which gives the adversary a 50% probability of winning. Therefore, by solving the draw, we can obtain: $p_{a.win} = p_{a.win} + p_{draw} * 0.5$, $p_{h.win} = p_{h.win} + p_{draw} * 0.5$. We calculate $p_{a.win}, p_{h.win}$ on EC and indicate the result in Fig. 3, where we compare $p_{a.win}$ between EC and NC. It is clear that EC limits the adversary's success rate at each epoch when $\alpha < 50\%$ due to the Poisson process-based *WinCounts* generation. In the following sections, we only consider the adversary with less than 50% of global storage power, and we calculate $p_{a.win}$ at each state to evaluate the adversary's impacts.

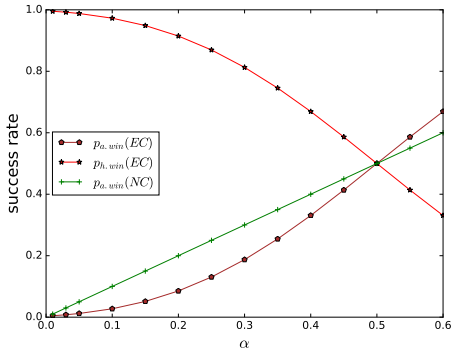


Figure 3: The probability that the adversary/honest miners find more *WinCounts* (e.g., the adversary/honest miners win) at each epoch on EC.

$$
\begin{cases}
q_z = 1 & if \quad p_{a.win} \geq p_{h.win} \\
q_z = \left(\frac{p_{a.win}}{p_{h.win}}\right)^z & if \quad p_{a.win} < p_{h.win}
\end{cases} \tag{6}
$$

Insight 1. Putting Eq. 6 into Nakamoto's evaluation [26], we find that EC makes double-spending attacks more difficult upon the assumption of honest majority.

Insight 2. Considering that 200 confirmations (approximately 100 minutes) are requested in EC, the success rate of double spending in EC is rare.

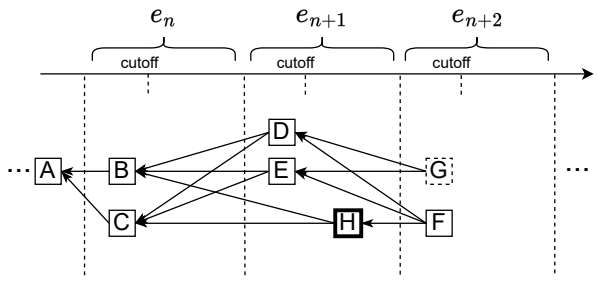


Figure 4: Illustration of the temporary block withholding in EC. e_n, e_{n+1}, e_{n+2} represent the 3 adjacent epochs, and related blocks in each epoch are denoted by rectangles. "cutoff" denotes the default time for miners to stop combining the blocks that were generated in the previous epoch. Please be aware that the blocks that are generated in epoch e_n will be released in epoch e_{n+1}. Thus, publicly visible blocks in e_{n+1} are always the blocks that were generated in e_n.

4.2 EC's "Achilles' heel"

As long as the adversary is elected as leader in an epoch, s/he can withhold the block. By doing so, s/he always has an advantage at the first withholding window, which favors the adversary to win the conflict at the next withholding window. As shown in Fig. 4, the adversary generates and withholds block H in e_{n+1}. In the next epoch, the honest miners' block G can only be linked with block D, E, and the adversary's block F can be linked with block D, E, H. Thus, the fork happens in e_{n+2}. Following GHOSS protocol, sub-graph that consists of block A, B, C, D, E, H, F would be selected as the heaviest one, while block G is orphaned.

Over the short term, i.e., two consecutive withholding epochs, the adversary is able to increase her success rate to replace the honest miners' blocks. Precisely, the adversary withholds her blocks in the first withholding epoch to generate the fork. The adversary then goes to the second consecutive withholding epoch with some leading *WinCounts*, which gives a natural advantage to the adversary who has more hidden *WinCounts*. There is a non-negligible probability that honest miners' blocks would be replaced by the adversary's hidden *WinCounts* (we analyze the probability in Sections 5, 6, 7). If such attacks happen frequently, the system's security and performance would be affected (we show the impacts in Section 8).

4.3 Length 2 Selfish Mining

Similar with GHOST protocol [38] and HLMD-GHOST protocol [5], EC's time constrained GHOSS protocol was made to prevent double spending attacks when block interval is shortened to seconds. However, double spending attack is not the only threat to blockchain system's security. Recently, selfish mining style attacks have been proposed in Ethereum

2.0 [36], which have the significant impact on Ethereum 2.0's security.

In this paper, we propose *TBW attacks* that are like one-block reorg attack [36] in Ethereum 2.0, which can be formalized as a short-term selfish mining strategy. As shown in Fig. 5, this type of attack can be summarized as: 1), the adversary generates and withholds the block whenever s/he is elected as the leader (i.e., with probability p to go to state 1); 2), the adversary immediately releases the private block/blocks in the next epoch (i.e., generate artificial fork and go to state $0'$); 3), with a probability, the adversary is able to win the fork selection.

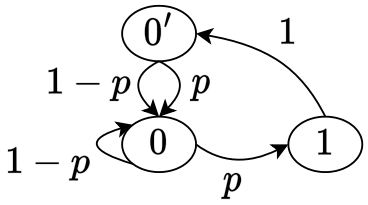


Figure 5: The Markov decision process of length 2 selfish mining attack.

5 Temporary Block Withholding Attack for Replacing Honest Miners' Blocks on EC (*TBW Attack 1.0*)

This section introduces the strategy of *TBW Attack 1.0* alongside the adversary's actions and algorithm. The Markov Decision Process (MDP) model is built to evaluate the impact of *TBW Attack 1.0* on an adversary's revenue share.

5.1 Intuition

As we indicated in Section 4.2, the adversary always has an accumulated advantage in epoch e_{t+1} when she discovers and withholds the block in epoch e_t. This advantage is eliminated in the subsequent epochs $e_{t+2}, e_{t+3}, \ldots$ if the adversary persists in withholding the blocks, because the adversary's success rate would vastly decrease in the Gambler's Ruin model (as analyzed in Section 4.1). Therefore, we only consider two consecutive block withholding epochs not only in *TBW Attack 1.0*, but also *TBW Attack 2.0* and *TBW Attack 3.0*. Our intuition is that this accumulated withholding advantage over the short term can help the adversary to use her private *tipsets* to replace the public *tipsets*. If such replacement happens frequently, the system's stability and performance would be affected.

5.2 Model

5.2.1 Notations

- $\mathcal{A}$ denotes the adversary and $\mathcal{H}$ denotes the honest miners;
- *WinCounts* denotes the units that miners can earn in each epoch following the official specification of EC[9];
- $TChain_{pub}, TChain_{private}$ represent the public and private chain of *tipsets* maintained by honest miners and the adversary. We introduce $TChain_{private}$ as an abstraction to evaluate temporary block withholding attacks following old-fashioned methods [11, 18, 27, 34];
- μ denotes the expected number of *WinCounts* that can be generated in each epoch. Currently, μ is hard-coded and equal to 5 in EC;
- n denotes the maximum number of *WinCounts* that can be accepted in each epoch. Currently, n is hard-coded and equal to 15 in EC[10];
- $\mathbb{N}_a(e_t), \mathbb{N}_h(e_t)$ denote the number of *WinCounts* that the adversary and honest miners can obtain in epoch e_t;
- ω_a denotes the number of accumulated hidden *WinCounts* and ω_h denotes the number of accumulated public *WinCounts* that conflict with the private chain of tipsets.

5.2.2 Storage Power Distribution

In *TBW Attack 1.0*, we consider that the adversary's storage power share α is smaller than 50% and that honest miners own the remaining $1-\alpha$ of global storage power.

5.2.3 Fork Resolution

As indicated in Section 3.3, the weights of $TChain_{pub}, TChain_{private}$ are defined by total storage power and the number of *WinCounts*. Considering the window of *TBW Attack 1.0* is only 2 epochs, where the change of network storage is negligible, we assume that the weights of $TChain_{pub}, TChain_{private}$ are completely determined by the number of *WinCounts* in our three attacks. When there is conflict, the chain of *tipsets* with more *WinCounts* would win. In terms of a draw, they have the same chance of winning[11].

[9] https://spec.filecoin.io/#section-systems.filecoin_blockchain

[10] https://github.com/filecoin-project/lotus/blob/master/chain/types/electionproof.go

[11] https://spec.filecoin.io/#section-algorithms.expected_consensus.selecting-between-tipsets-with-equal-weight

5.3 Strategy

The main idea is as follows. The adversary $\mathcal{A}$ temporarily withholds some blocks (i.e., *WinCounts*) to generate a fork, then leverages the asymmetric information to execute actions to optimize her success rate of replacing the honest miners' *WinCounts*. We detail the adversary's actions and the pseudo-code for *TBW Attack 1.0* as follows. Because the key metric of solving the fork is the number of *WinCounts* but not blocks, we use *WinCounts* to replace blocks when we talk about the adversary's actions. We emphasize that the withholding and releasing of *WinCounts* in our three attacks mean the withholding and releasing of the blocks that include the corresponding *WinCounts* (as introduced in Section 3.2).

5.3.1 Actions

- **Adopt**: the adversary accepts the public tipsetchain and abandons her private *WinCounts*.
- **Withhold**: the adversary privately keeps her newly discovered *WinCounts*.
- **Override**: the adversary uses her private tipsetchain to replace the public tipsetchain.
- **Match**: the adversary releases her private tipsetchain to match the public tipsetchain.

5.3.2 Algorithm

We use the abstraction of an adversary's private chain of *tipsets* ($TChain_{private}$) to evaluate the impact of temporary block-withholding behaviors. In each epoch, we define $\mathcal{A}$'s actions under different scenarios in Algorithm 1. Unlike in traditional PoW based consensus protocols, both $\mathcal{A}$ and $\mathcal{H}$ are able to obtain some *WinCounts* following the Poisson distribution in each epoch. Therefore, the competition in EC's epoch is actually to compare the number of *WinCounts* that $\mathcal{A}$ can obtain ($\mathbb{N}_a(e_t)$) and the number of *WinCounts* that $\mathcal{H}$ can obtain ($\mathbb{N}_h(e_t)$). As analyzed in Section 4.2, whenever $\mathcal{A}$ obtains some *WinCounts* and two chains are equal (Algo. 1 line 7), $\mathcal{A}$ withholds the *WinCounts*. If $\mathcal{A}$ does not obtain any *WinCount* in the epoch and two *tipsetchains* are equal (Algo 1 line 11), then $\mathcal{A}$ adopts $TChain_{pub}$. If $\mathcal{A}$'s accumulated hidden *WinCounts* (ω_a) are more than the accumulated public *WinCounts* (ω_h) that conflict with hidden *WinCounts* and two chains are not equal (Algo 1 line 19), then $\mathcal{A}$ uses her hidden *WinCounts* to replace the public conflicting *WinCounts*. If $\mathcal{A}$'s accumulated hidden *WinCounts* (ω_a) are less than the accumulated public *WinCounts* (ω_h) that conflict with hidden *WinCounts* and two chains are not equal (Algo. 1 line 22), then $\mathcal{A}$ abandons her hidden *WinCounts* and adopts $TChain_{pub}$. When two chains are not equal and their weights are equal (Algo. 1 line 25), then $\mathcal{A}$ releases the hidden *WinCounts*, which leads to a draw and $\mathcal{A}, \mathcal{H}$ have equal chances to win.

Algorithm 1 TBW Attack 1.0

$TChain_{pub} \leftarrow$ the heaviest chain of tipsets
$TChain_{private} \leftarrow$ the heaviest chain of tipsets
syn=$True$//two chains are equal
ω_a=0
ω_h=0

1: **At** epoch e_t
2: $\mathcal{A}$ obtains $\mathbb{N}_a(e_t)$ *WinCounts*
3: $\mathcal{H}$ obtains $\mathbb{N}_h(e_t)$ *WinCounts*
4: **if** ω_a==0 **and** ω_h==0 **and** syn==$True$
5: $tipsetchain_{pub} \leftarrow \mathbb{N}_h(e_t)$ *WinCounts*
6: $tipsetchain_{private} \leftarrow \mathbb{N}_h(e_t)$ *WinCounts*
7: **if** $\mathbb{N}_a(e_t)$!=0 //withhold
8: ω_a+=$\mathbb{N}_a(e_t)$
9: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
10: syn=$False$
11: **else** //adopt
12: $tipsetchain_{private}$=$tipsetchain_{pub}$
13: ω_a=0;ω_h=0;syn=$True$
14: **else**
15: ω_a+=$\mathbb{N}_a(e_t)$
16: ω_h+=$\mathbb{N}_h(e_t)$
17: $tipsetchain_{pub} \leftarrow \mathbb{N}_h(e_t)$ *WinCounts*
18: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
19: **if** ω_a>ω_h //override
20: $tipsetchain_{pub}$=$tipsetchain_{private}$
21: ω_a=0;ω_h=0;syn=$True$
22: **else if** ω_a<ω_h //adopt
23: $tipsetchain_{private}$=$tipsetchain_{pub}$
24: ω_a=0;ω_h=0;syn=$True$
25: **else** //match
26: **if** $\mathcal{A}$ wins (50% probability)
27: Execute line 20, 21
28: **else** $\mathcal{H}$ wins (50% probability)
29: Execute line 23, 24

5.4 Markov Decision Process

To evaluate the impact of the adversary's strategy, we detail the MDP model as follows.

5.4.1 States

Each state is represented by the number of leading *WinCounts* that the adversary has on her private tipsetchain versus the public tipsetchain. $a_1, a_2, ..., a_n$ denote the states that the adversary has $1, 2, ..., n$ leading *WinCounts*, and $h_1, h_2, ..., h_n$ denote the states that the honest miners have $1, 2, ..., n$ leading WinCounts. $0'$ denotes the state in which the adversary and

Table 1: State transition of *TBW Attack 1.0*. X, Y represent the number of *WinCounts* that $\mathcal{A}, \mathcal{H}$ can obtain in the second consecutive withholding epoch.

State	Action	Resulting State	Transition Probability	Reward (r_a, r_h)
s_0	adopt	s_0	$e^{-\mu*\alpha}$	$(0, \mu*(1-\alpha))$
s_0	withhold	s_{Δ_1,a_k}	$\frac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha}$	$(0, \mu*(1-\alpha))$
s_{Δ_1,a_k}	override	s_{Δ_2,a_i}	$\sum_{0\le X\le n, 0\le Y\le n, X+k-Y=i} (\frac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})$	$(\mu*\alpha+k, 0)$
s_{Δ_1,a_k}	adopt	s_{Δ_2,h_i}	$\sum_{0\le X\le n, 0\le Y\le n, Y-X-k=i} (\frac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})$	$(0, \mu*(1-\alpha))$
s_{Δ_1,a_k}	match	$s_{\Delta_2,0'}$	$\sum_{0\le X\le n, 0\le Y\le n, X+k=Y} (\frac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})$	$(\mu*\alpha+k, 0)$ or $(0, \mu*(1-\alpha))$
s_{Δ_2}	adopt	s_0	$e^{-\mu*\alpha}$	$(0, \mu*(1-\alpha))$
s_{Δ_2}	withhold	s_{Δ_1,a_k}	$\frac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha}$	$(0, \mu*(1-\alpha))$

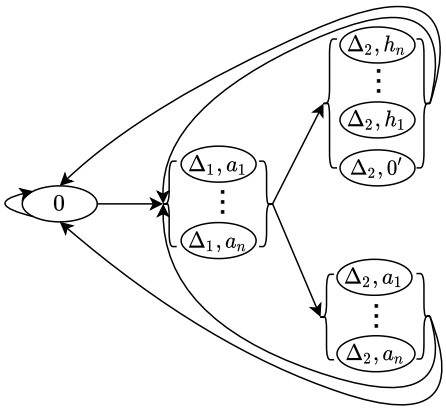


Figure 6: The Markov decision process of *TBW Attack 1.0*. We use Δ_1, Δ_2 to denote two slots in different epochs when there are forks. Δ_1 represents the ending time of the assemblage of the first pair of conflicting tipsets, and Δ_2 represents the ending time of the assemblage of the second consecutive pair of conflicting tipsets.

the honest miners have equal weight, but the tipsetchains are not same. We only consider two withholding windows, which are represented by Δ_1, Δ_2. The entire state space is denoted by set $S: \{s_0, s_{\Delta_1,a_1}, ..., s_{\Delta_1,a_n}, s_{\Delta_2,a_1}, ..., s_{\Delta_2,a_n}, s_{\Delta_2,h_1}, ..., s_{\Delta_2,h_n}, s_{\Delta_2,0'}\}$.

5.4.2 Transition probability

We summarize 7 types of transitions in Table 1, where s_{Δ_1,a_k} notes the $k^{th}, k \in [1,n]$ state after the first withholding window Δ_1, where the adversary withholds her private *WinCounts* to generate fork. The adversary then releases her private *WinCounts* after the second withholding window Δ_2. As a consequence, we have three types of states after Δ_2, which are represented by s_{Δ_2,a_i} (the $i^{th}, k \in [1,n]$ winning state after the second withholding window Δ_2), s_{Δ_2,h_i} (the $i^{th}, k \in [1,n]$ loss state after the second withholding window Δ_2), and $s_{\Delta_2,0'}$ (the draw state after the second withholding window Δ_2). We list transition probabilities in Table 1.

5.4.3 State probability

We calculate state probabilities based on Fig. 6. Let P_0 be the probability of state s_0, P_{Δ_1,a_k} be the probability of state s_{Δ_1,a_k}, and P_{Δ_2,a_k} be the probability of state s_{Δ_2,a_k}. Then we can obtain:

$$
\begin{cases}
P_0 * (1 - e^{-\mu*\alpha}) = (\sum_{i=1}^{i=n} (P_{\Delta_2,a_i} + P_{\Delta_2,h_i}) + P_{\Delta_2,0'}) * e^{-\mu*\alpha} \\
\forall k \in [1,n]: P_{\Delta_1,a_k} = (\sum_{i=1}^{i=n} (P_{\Delta_2,a_i} + P_{\Delta_2,h_i}) + P_{\Delta_2,0'} + P_0) \\
\qquad\qquad * \frac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha} \\
\forall i \in [1,n]: P_{\Delta_2,a_i} = \sum_{k=1}^{k=n} (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, X+k-Y=i} \\
\qquad\qquad (\frac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})) \\
\forall i \in [1,n]: P_{\Delta_2,h_i} = \sum_{k=1}^{k=n} (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, Y-X-k=i} \\
\qquad\qquad (\frac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})) \\
P_{\Delta_2,0'} = \sum_{k=1}^{k=n} (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, X+k=Y} (\frac{(\mu*\alpha)^X}{X!} * \\
\qquad\qquad e^{-\mu} * \frac{(\mu*(1-\alpha))^Y}{Y!})) \\
P_0 + \sum_{k=1}^{k=n} P_{\Delta_1,a_k} + \sum_{i=1}^{i=n} (P_{\Delta_2,a_i} + P_{\Delta_2,h_i}) + P_{\Delta_2,0'} = 1
\end{cases}
\tag{7}
$$

By solving Eq. 7 (see detailed calculations in Appendix A), we can obtain:

$$
\begin{cases}
P_0 = \dfrac{e^{-\mu*\alpha}}{2-e^{-\mu*\alpha}} \\
\forall k \in [1,n] : P_{\Delta_1,a_k} = \dfrac{1}{2-e^{-\mu*\alpha}} * \dfrac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha} \\
\forall i \in [1,n] : P_{\Delta_2,a_i} = \sum_{k=1}^{k=n} (\dfrac{1}{2-e^{-\mu*\alpha}} * \dfrac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha} * \\
\qquad \sum_{0\le X\le n, 0\le Y\le n, X+k-Y=i} (\dfrac{(\mu*\alpha)^X}{X!} \\
\qquad * e^{-\mu} * \dfrac{(\mu*(1-\alpha))^Y}{Y!})) \\
\forall i \in [1,n] : P_{\Delta_2,h_i} = \sum_{k=1}^{k=n} (\dfrac{1}{2-e^{-\mu*\alpha}} * \dfrac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha} * \\
\qquad \sum_{0\le X\le n, 0\le Y\le n, Y-X-k=i} (\dfrac{(\mu*\alpha)^X}{X!} \\
\qquad * e^{-\mu} * \dfrac{(\mu*(1-\alpha))^Y}{Y!})) \\
P_{\Delta_2,0'} = \sum_{k=1}^{k=n} (\dfrac{1}{2-e^{-\mu*\alpha}} * \dfrac{(\mu*\alpha)^k}{k!} * e^{-\mu*\alpha} * \\
\qquad \sum_{0\le X\le n, 0\le Y\le n, X+k=Y} (\dfrac{(\mu*\alpha)^X}{X!} \\
\qquad * e^{-\mu} * \dfrac{(\mu*(1-\alpha))^Y}{Y!}))
\end{cases}
\tag{8}
$$

5.4.4 Expected revenue and revenue share

Let R_a, R_h be the expected revenue of the adversary and the honest miners, respectively. According to Table 1, we can obtain:

$$
\begin{cases}
R_a = \sum_{i=1}^{i=n} \sum_{k=1}^{k=n} (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, X+k-Y=i} (\dfrac{(\mu*\alpha)^X}{X!} \\
\quad * e^{-\mu} * \dfrac{(\mu*(1-\alpha))^Y}{Y!}) * (\mu*\alpha+k)) + 0.5 * \sum_{k=1}^{k=n} \\
\quad * (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, X+k=Y} (\dfrac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \\
\quad \dfrac{(\mu*(1-\alpha))^Y}{Y!}) * (\mu*\alpha+k)) \\
R_h = \sum_{i=1}^{i=n} \sum_{k=1}^{k=n} (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, Y-X-k=i} (\dfrac{(\mu*\alpha)^X}{X!} \\
\quad * e^{-\mu} * \dfrac{(\mu*(1-\alpha))^Y}{Y!}) * \mu*(1-\alpha)) + 0.5 * \sum_{k=1}^{k=n} \\
\quad * (P_{\Delta_1,a_k} * \sum_{0\le X\le n, 0\le Y\le n, X+k=Y} (\dfrac{(\mu*\alpha)^X}{X!} * e^{-\mu} * \\
\quad \dfrac{(\mu*(1-\alpha))^Y}{Y!}) * \mu*(1-\alpha)) + (P_0 + \sum_{i=1}^{i=n} (P_{\Delta_2,a_i} + \\
\quad P_{\Delta_2,h_i}) + P_{\Delta_2,0'}) * \mu*(1-\alpha)
\end{cases}
\tag{9}
$$

Let RS_a be the expected revenue share of the adversary. Simply, we have $RS_a = \frac{R_a}{R_a+R_h}$, which can be solved by using the expressions of R_a, R_h in Eq. 9. We use RS_a as the key metric to evaluate the impact of temporary block withholding attacks in EC. We show the results of solving the MDP equations in Section 8 and compare them with results of the Monte Carlo simulations.

6 Temporary Block Withholding Attack for Splitting Honest Miners (*TBW Attack 2.0*)

In this section, we take honest miners' diversity into account. Precisely, this diversity relies on the fact that different honest miners may have different transmission cutoff times[12], and some rational miners assemble their blocks after the default cutoff time and update their *tipsets*[13], which have been detected by Filecoin's network monitors [14, 15] (i.e., some *tipsets* include the blocks that are generated after the default transmission cutoff time). We do not repeat the same procedures described in *TBW Attack 1.0*, but we present key updates of *TBW Attack 2.0*.

6.1 Intuition

The adversary leverages the cutoff time delay between altruistic miners and rational miners to generate forks. By doing so, the adversary not only splits the honest miners, but also further optimizes her revenue share thanks to the rational miners' collaboration (i.e., the rational miners would work on the adversary's branch as long as the adversary releases her private *WinCounts* after the altruistic miners' transmission cutoff time).

6.2 Model

The model *TBW Attack 1.0* is similar except for the types of participants. Let δ be the default transmission cutoff time. We define the participants in *TBW Attack 2.0* as follows.

- **Attacker $\mathcal{A}$.** She can withhold the *WinCounts* in epoch e_{t+1} when she is elected as the leader at epoch e_t.
- **Altruistic miners $\mathcal{AH}$.** They completely obey the default transmission cutoff time. At each epoch, they would not change their *tipsets* after δ even if new blocks arrive afterwards.
- **Rational miners $\mathcal{RH}$.** In each epoch, they update their tipsets when some blocks arrive after δ. They frequently check valid and available *WinCounts* in the network to optimize the weight of their *tipsets*. In this paper, the rational behavior of honest miners is defined as: *miners postpone the cutoff time to try to include more blocks in their tipsets; meanwhile, they must ensure that their blocks can be produced before the end of the epoch and can be released when the next epoch starts*.

6.3 Strategy

The adversary has a new action in *TBW Attack 2.0* compared to *TBW Attack 1.0*, as follows.

[12] https://spec.filecoin.io/#section-systems.filecoin_mining.storage_mining.mining_cycle.epochtiming

[13] https://github.com/filecoin-project/lotus/blob/master/miner/miner.go#L184

- **Cutoff Time (CT) release**: the adversary releases her private *WinCounts* at δ', which is greater than the default transmission cutoff time δ but smaller than the epoch time ($\delta < \delta' < 30s$). As a consequence, $\mathcal{AH}$ would not accept adversary's *WinCounts*, but $\mathcal{RH}$ would adopt them to update their *tipsets*. The adversary therefore can split $\mathcal{AH}$ and $\mathcal{RH}$, and attract $\mathcal{RH}$ to work on her branch. "release" is actually the common part of actions **override**, **adopt**, and **match**. In *TBW Attack 2.0*, this common part is executed in advance by action **CT release**. Therefore, action **override**, **adopt**, and **match** in *TBW Attack 2.0* only reflect the replacement of *tipsets* between $TChain_{pub}$ and $TChain_{private}$.

We update the algorithm based on action **CT release** for *TBW Attack 2.0*, which is attached in Appendix B.

6.4 Markov Decision Process

We introduce a new state s'_{Δ_1,a_k} (i.e., the state between s_{Δ_1,a_k} and s_{Δ_2}[14]) in *TBW Attack 2.0* because of the adversary's new action **CT release**. The main effect of this action is to shift $\mathcal{RH}$'s storage power λ completely from $TChain_{pub}$ to $TChain_{private}$. We summarize the updated transitions along with the transition probabilities in Appendix C, Table 2, where $\mathcal{RH}$'s storage power is shifted from state s_{Δ_1,a_k} to state s'_{Δ_1,a_k}, which increases the probability of arriving in state s_{Δ_2,a_i} and decreases the probability of arriving in state s_{Δ_2,h_i}. This would increase the success rate of the adversary's branch to be accepted.

In terms of reward distribution (r_a, r_h), from s'_{Δ_1,a_k} to s_{Δ_2,a_i}, we have (r_a, r_h)=$(\mu*\alpha+k, \mu*\lambda)$; from s'_{Δ_1,a_k} to s_{Δ_2,h_i}, we have (r_a, r_h)=$(0, \mu*(1-\alpha-\lambda))$; and from s'_{Δ_1,a_k} to $s_{\Delta_2,0'}$, we have (r_a, r_h)=$(\mu*\alpha+k, \mu*\lambda)$ or (r_a, r_h)=$(0, \mu*(1-\alpha-\lambda))$. We indicate the results in Section 8.

7 Temporary Block Withholding Attack with Front Epoch Prediction (*TBW Attack 3.0*)

Our third attack is based on the fact that each participant is able to get the randomness beacon one epoch in advance[15]. This would further optimize $\mathcal{A}$'s revenue share.

7.1 Intuition

In *TBW Attack 1.0* and *TBW Attack 2.0*, the adversary withholds all *WinCounts* that she obtains at state s_0, which drives the first pair of conflicting tipsets. Although this withholding action accumulates an advantage for $TChain_{private}$, the adversary risks losing all of her private *WinCounts* in the next epoch, mainly due to the fact that the adversary might not be elected as the leader in the next epoch (i.e., the adversary obtains 0 *WinCounts*). This probability increases as the adversary's storage power decreases. Consequently, the crux for the adversary with a small hash power is, why should the adversary withhold *WinCounts* since she has a low chance to be selected as the leader in the next epoch? In *TBW Attack 3.0*, the adversary makes the decision depending on not only the number of *WinCounts* she can obtain at the current epoch, but also whether she would be elected as the leader in the next epoch, which we call **Front Epoch Prediction**. This updated strategy can further lower the threshold of launching such attacks.

7.2 Model

The model is as same as *TBW Attack 2.0*. In *TBW Attack 3.0*, we also consider that $\mathcal{AH}, \mathcal{RH}$ share the storage power of the honest miners.

7.3 Strategy

We introduce a new action in *TBW Attack 3.0* as follows.

- **Front Epoch Prediction (FEP) withhold.** The adversary withholds the *WinCounts* only when she is elected as the leader in two consecutive epochs via the front execution of the randomness beacon of the next epoch.

We update the algorithm based on action **FEP withhold** for *TBW Attack 3.0*, which is attached in Appendix D.

7.4 Markov Decision Process

The updates of the MDP model between *TBW Attack 2.0* and *TBW Attack 3.0* are the transition probabilities because of action **FEP withhold**. We summarize the updated transition probabilities in Appendix E, Table 3, where the transition probability from s_0 to s_{Δ_1,a_k} in *TBW Attack 3.0* decreases compared to *TBW Attack 1.0, 2.0* due to the fact that the adversary needs to be elected as the leader in two consecutive epochs. This further increases the transition probability between s_{Δ_1,a_k} and s_{Δ_2,a_i}, and it decreases the transition probability between s_{Δ_1,a_k} and s_{Δ_2,h_i}. As a result, the adversary with small storage power would be able to increase her revenue share and thus can launch *TBW Attack 3.0* to harm the system. We indicate the effects of this updated strategy in Section 8.

8 The Impacts of TBW Attacks in EC

This section evaluates the impacts of our three attacks in EC with a focus on system security and performance. We first introduce the methodology and then present the results. Finally, we provide implications and insights based on our findings.

[14] State s_{Δ_2} includes state s_{Δ_2,a_i},s_{Δ_2,h_i},and $s_{\Delta_2,0'}$.

[15] https://github.com/filecoin-project/lotus/blob/3e6c482229fb4230b871f2d2baab2357077482df/miner/miner.go#L424

8.1 Methodology

We calculate the expected revenue of $\mathcal{A}, \mathcal{AH}, \mathcal{RH}$ in our three attacks based on the MDP models, which allows us to estimate the expected revenue share and stale block rate. We build Monte Carlo simulators for our attacks respectively to mimic the behaviors of different types of participants. We indicate that the results of mathematical expectations via MDP models match the results of Monte Carlo simulations.

Monte Carlo Simulators. We build Monte Carlo simulators for our three attacks based on the algorithms in Section 5, 6, 7. We will release the source code as long as the shortcomings of EC are fixed by Protocol Labs[16], the maintainer of Filecoin. The results of our three attacks in this paper are based on 100 epochs with 100,000 rounds of Monte Carlo simulations.

8.2 Results

We use two metrics, revenue share and stale block rate, to evaluate the impacts of our attacks on EC. In particular, we use the former to capture the impact of our attacks on EC's security and the latter to evaluate EC's performance. Both metrics are applied in MDP models and Monte Carlo simulations; the results are as follows. Moreover, we discuss the minimum storage power the adversary needs to harm the system depending on the impacts.

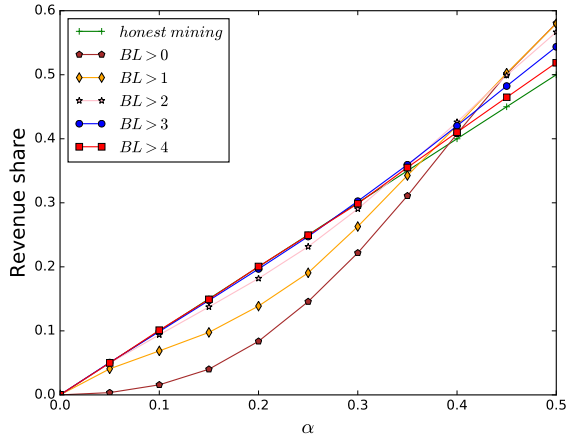


Figure 7: The expected revenue share of *TBW Attack 1.0*.

8.2.1 Revenue Share

As we have defined in Section 5.4.4, the adversary's revenue share reflects the ratio of adversary's revenue to global revenue. This metric has been widely used to analyze the mining re-centralization issue [11], mining fairness [4], and incentive compatibility [18] in PoW based blockchains. We use it to evaluate how much revenue the adversary can gain with α of the global storage power in EC.

Result 1. The result of our first attack anchors the threshold of breaking EC's mining fairness where the system is in perfect implementation and every participant is altruistic except the adversary. As shown in Fig. 7, in the standard case of *TBW Attack 1.0*, the threshold is $\alpha = 38.5\%$. When the adversary's storage power is sufficient, she can optimize her strategy here by accumulating many *WinCounts* to start *TBW Attack 1.0*. We use *BL* to represent the number of leading *WinCounts* that the adversary can obtain in the first withholding window and use it as the precondition to start *TBW Attack 1.0*. In the optimized case, the threshold is in the range of $[35\%, 38.5\%]$.

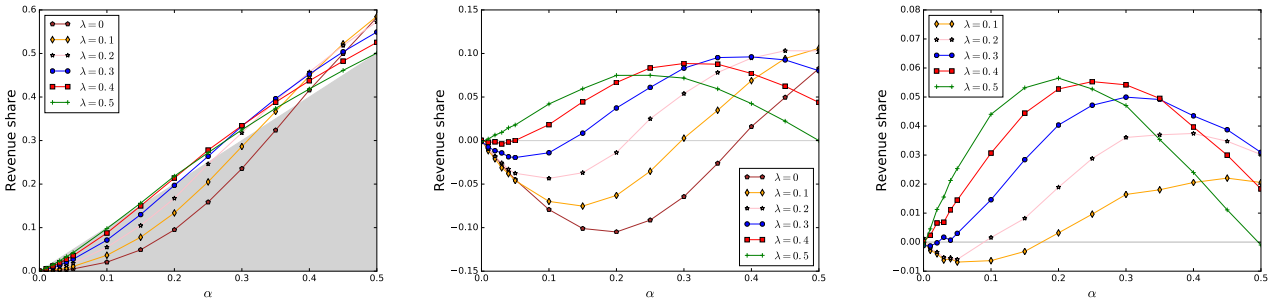
Result 2. The result of our second attack reveals the vulnerability of EC due to the fact that altruism is not trusted in the decentralized network [7] and some rational behaviors have been detected in Filecoin (as discussed in Section 6). As shown in Fig. 8(a), in the presence of $\mathcal{RH}$ who owns λ of global storage power, *TBW Attack 2.0* significantly decreases the threshold of breaking EC's mining fairness. For instance, the threshold would be approximately 10%, 15%, 22%, 26%, and 33% if λ is equal to 0.5, 0.4, 0.3, 0.2, and 0.1, respectively. Importantly, $\mathcal{AH}$ would suffer a loss even if the threshold of breaking the mining fairness is not reached. As shown in Fig. 8(b), the adversary with a small storage power share (e.g., 1%, 2%) is able to decrease $\mathcal{AH}$'s revenue share when $\lambda = 0.5$. The result is justified by the fact that the small miner leverages $\mathcal{RH}$'s storage power to beat $\mathcal{AH}$. As a consequence, $\mathcal{RH}$ would benefit from *TBW Attack 2.0* (as shown in Fig. 8(c)).

Result 3. The result of our third attack further amplifies the vulnerability of EC. By leveraging action **FEP release**, the adversary is able to optimize her revenue share, since she starts the attack only when she can be elected in two consecutive epochs. As shown in Fig. 9(a), the threshold of breaking mining fairness integrally decreases compared with the result of *TBW Attack 2.0*. By contrast, the impacts on $\mathcal{AH}$'s loss (as shown in Fig. 9(b)) and $\mathcal{RH}$'s increment (as shown in Fig. 9(c)) slightly decreases. The main reason is the fact that action **FEP release** converges all state probabilities in the second consecutive withholding epoch. It is remarkable that *TBW Attack 3.0* renders the adversary with a small storage power more threatening to the system. For instance, the adversary is able to decrease $\mathcal{AH}$'s revenue share when $\alpha = 5\%$ and $\lambda = 0.5$ without a loss of revenue share in *TBW Attack 3.0*, but with a loss revenue share in *TBW Attack 2.0*.

8.2.2 Stale Block Rate

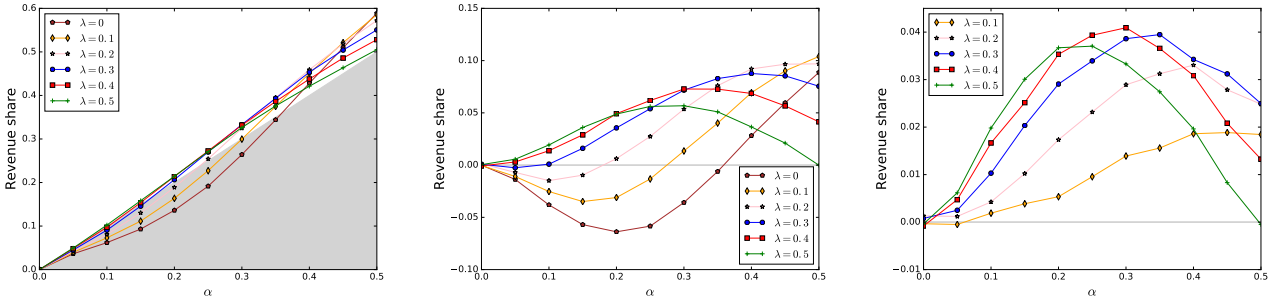
Stale block rate has been widely used to evaluate the performance of PoW based blockchains [18]. In this evaluation, we use this metric to evaluate EC's performance. We assume that one *WinCount* is equal to one block by considering two facts:

[16] https://protocol.ai/



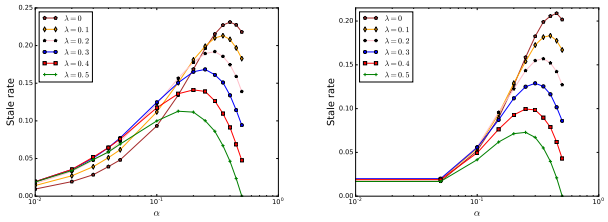
(a) $\mathcal{A}$'s revenue share (b) $\mathcal{AH}$'s loss (c) $\mathcal{RH}$'s increment

Figure 8: The results of *TBW Attack 2.0*. λ represents $\mathcal{RH}$'s storage power share.



(a) $\mathcal{A}$'s revenue share (b) $\mathcal{AH}$'s loss (c) $\mathcal{RH}$'s increment

Figure 9: The results of *TBW Attack 3.0*. λ represents $\mathcal{RH}$'s storage power share.



(a) *TBW Attack 1.0, 2.0* (b) *TBW Attack 3.0*

Figure 10: The results of stale rate. λ represents $\mathcal{RH}$'s storage power share.

1) the owner of storage space normally implements multiple miners to join EC; and 2) it is rare that one block includes more than one *WinCount* [14, 15]. The stale block rate is then estimated by using the number of orphaned *WinCounts* to divided by the total number of *WinCounts*. Results are as follows.

Result 4. Our three attacks increase the stale block rate, which decreases Filecoin's transaction throughput. As shown in Fig. 10, the adversary with a small storage power would ($\alpha < 5\%$) benefit from a high value of λ, which reflects the fact that $\mathcal{RH}$ assists $\mathcal{A}$ to beat $\mathcal{AH}$ in the mining competition. When $\alpha > 5\%$, there exists an optimal λ at which the adversary maximizes her impact on EC's transaction throughput. Remarkably, the adversary with 2% of the network storage is able to cause a 5% stale block rate as shown in Fig. 10(a), which reveals that *TBW Attack 2.0* is severely threatening to Filecoin's performance in terms of transaction throughput.

8.3 Insights and Countermeasures

Our results indicate that EC is vulnerable to temporary block withholding attacks. While it is proved that EC's designs can effectively prevent double spending attacks, its shortcomings can be leveraged by the adversary to harm a system's mining fairness and decrease its performance. Over the long term, our three attacks are able to evict some profit-driven miners from the system since their revenue shares are decreased, which would affect system stability. In particular, the adversary with a small network storage is able to launch *TBW Attack 2.0,3.0* without suffering loss of revenue share, which implicates EC's vulnerability.

We provide countermeasures to mitigate temporary block

withholding attacks as follows.

8.3.1 Synchronizing Filecoin network and drand nework

TBW Attack 3.0 would be impossible if **Front Epoch Prediction** is prevented. In fact, this can be simply fixed by synchronizing Filecoin network and drand network; doing this would ensure no one can get the randomness beacon in advance. However, we do not know whether this creates other issues. For instance, the miner with better network connectivity would be able to get the randomness beacon earlier than others, which amplifies the diversity that might be leveraged by *TBW Attack 2.0*. We will further explore the feasibility of this scheme.

8.3.2 Adjusting the default transmission cutoff time

$\mathcal{AH}$ obeys the default protocol, which does not accept blocks after the default transmission cutoff time. The adversary leverages this setting to launch *TBW Attack 2.0*. If the default transmission cutoff time is adjusted properly, $\mathcal{AH}$ would be able to assemble *tipsets* like $\mathcal{RH}$ and update their *tipsets* as much as possible in the epoch. This would make it difficult for the adversary to split the honest miners. However, considering the fact that different miners have different capacities to execute PoSt mechanism to prove the validity and consistency of storage, the diversity of honest miners would still exist in the network. *TBW Attack 2.0* might have an updated strategy to adopt it.

8.3.3 Monitoring the network

Since 2013, network monitors [6, 9, 21, 25, 28] have been widely deployed in cryptocurrencies (e.g., Bitcoin, Ethereum, Monero) to evaluate and track network performance, which could lead to some potential issues. For instance, by monitoring the network, one can estimate fork rate (or stale block rate), which is often used to analyze blockchain's security and performance [8, 18]. To mitigate *TBW attack 2.0*, we suggest deploying monitors in the Filecoin network in order to track system performance and detect anomalous behaviors. For instance, from data we collected from September 2021 to September 2022, the fork rate was approximately 2% to 3%. Based on this, an increased fork rate suggests an anomaly. Besides, a miner is suspected to be malicious if her blocks frequently arrive in other nodes after the cutoff time (this can observed via the monitoring nodes). Though it remains challenging to detect attackers (e.g., delayed blocks might be because of loose network connections), the monitor would help the community to lock the suspects. Furthermore, if the default cutoff time is increased properly (e.g., from the 15^{th} to the 20^{th} second of each epoch), the adversary who wants to launch *TBW Attack 2.0* must withhold her blocks longer than 15 seconds, which is obviously longer than the normal block propagation delay in Internet-based peer-to-peer networks. If this anomalous delay is frequently observed by the monitor, an attack can be detected with high probability. Recall that all miners have been registered in the power table, and Filecoin has deployed an audit protocol to punish some malicious behaviors[17]. A network monitor would help to alleviate *TBW Attack 2.0*, since they would be punished as long as they are detected.

9 Conclusion

In this paper, we first conduct a novel study of Filecoin's consensus layer. In particular, we deconstruct EC by considering some well-established properties, which not only helps improve our understanding of EC's mechanisms and sub-protocols, but also illustrates the dis/advantages of EC's design. We then propose three temporary block withholding attacks to challenge EC. Our first attack anchors the adversary's threshold in cases where there is no implementation issue and every participant is altruistic and honest. Our second attack splits honest miners and decreases the adversary's threshold by leveraging miners' rationality. Our third attack further optimizes the adversary's threshold based on the second attack. We built MDP models and Monte Carlo simulations to evaluate the impacts of the three attacks. Our results indicate that such temporary block withholding attacks are threatening in Filecoin's current settings. For instance, the adversary with 1% of the network storage is able to launch the attack without suffering a loss of revenue share, which is rare in PoW based blockchains. Moreover, we indicated that such attacks can affect Filecoin's performance in terms of transaction throughput, which implicates another vulnerability of EC. Finally, we provide our insights and countermeasures.

Acknowledgment

We would like to thank Sarah Azouvi and Xuechao Wang for their valuable comments and suggestions. This work is partially supported by Protocol Labs under grant RFP-X.

References

[1] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*, 2013.

[2] Juan Benet and Nicola Greco. Filecoin: A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

[3] Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. In *International Symposium on Cyber Security*

[17] https://spec.filecoin.io/#section-algorithms.expected_consensus.consensus-faults

Cryptography and Machine Learning, pages 114–127. Springer, 2021.

[4] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy*, pages 104–121. IEEE, 2015.

[5] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.

[6] Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verissimo. Exploring the monero peer-to-peer network. In *International Conference on Financial Cryptography and Data Security*, pages 578–594. Springer, 2020.

[7] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

[8] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.

[9] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[10] David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 1992.

[11] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[12] Lei Fan and Hong-Sheng Zhou. A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake). *Cryptology ePrint Archive*, 2017.

[13] Chen Feng and Jianyu Niu. Selfish mining in ethereum. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1306–1316. IEEE, 2019.

[14] filfox.info. Filfox. https://filfox.info/, accessed January 31, 2022.

[15] filscan.io. filscan. https://filscan.io/, accessed January 31, 2022.

[16] Ben Fisch. Tight proofs of space and replication. In *Advances in Cryptology–EUROCRYPT 2019*. Springer, 2019.

[17] Ben Fisch, Joseph Bonneau, Nicola Greco, and Juan Benet. Scaling proof-of-replication for filecoin mining. *Benet//Technical report, Stanford University*, 2018.

[18] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[19] Xin He and Yaacov Yesha. Parallel recognition and decomposition of two terminal series parallel graphs. *Information and Computation*, 1987.

[20] Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer, 2014.

[21] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. Measuring ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*, pages 91–104, 2018.

[22] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[23] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[24] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.

[25] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. *et al*, 2015.

[26] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[27] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[28] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. 2016.

[29] Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. Low-cost attacks on ethereum 2.0 by sub-1/3 stakeholders. *arXiv preprint arXiv:2102.02247*, 2021.

[30] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptol. ePrint Arch.*, 2015:1098, 2015.

[31] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[32] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2017.

[33] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[34] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[35] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on proof-of-stake ethereum. *arXiv preprint arXiv:2110.10086*, 2021.

[36] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on proof-of-stake ethereum. In *Financial Cryptography and Data Security*. Springer, 2022.

[37] Wellington Fernandes Silvano and Roderval Marcelino. Iota tangle: A cryptocurrency to communicate internet-of-things data. *Future generation computer systems*, 2020.

[38] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[39] Shawn Wilkinson, Jim Lowry, and Tome Boshevski. Metadisk a blockchain-based decentralized file storage application. *Storj Labs Inc., Technical Report, hal*, pages 1–11, 2014.

[40] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[41] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.

A Probability Calculation Based on MDP Model

Let $P_{\Delta_1}, P_{\Delta_2}$ be the sum of the probabilities of all states at Δ_1, Δ_2, simply, we have:

$$\begin{cases} P_0 + P_{\Delta_1} + P_{\Delta_2} = 1 \\ P_{\Delta_2} = P_{\Delta_1} \\ P_{\Delta_1} = (P_0 + P_{\Delta_2}) * (1 - e^{-\mu * \alpha}) \end{cases} \tag{10}$$

By solving 10, we can obtain:

$$\begin{cases} P_0 = \dfrac{e^{-\mu*\alpha}}{2 - e^{-\mu*\alpha}} \\ P_{\Delta_1} = \dfrac{1 - e^{-\mu*\alpha}}{2 - e^{-\mu*\alpha}} \\ P_{\Delta_2} = \dfrac{1 - e^{-\mu*\alpha}}{2 - e^{-\mu*\alpha}} \end{cases} \tag{11}$$

Putting Eq. 11 into Eq. 7, we can get all state probabilities, which are indicated in Eq. 8.

B Algorithm of *TBW Attack 2.0*

Algorithm 2 TBW Attack 2.0

$tipsetchain_{pub} \leftarrow$ the heaviest tipsetchain
$tipsetchain_{private} \leftarrow$ the heaviest tipsetchain
syn=$True$//two tipsetchains are equal
ω_a=0
ω_{ah}=0
ω_{rh}=0

1: **At** epoch e_t
2: $\mathcal{A}$ obtains $\mathbb{N}_a(e_t)$ *WinCounts*
3: $\mathcal{AH}$ obtains $\mathbb{N}_{ah}(e_t)$ *WinCounts*
4: $\mathcal{RH}$ obtains $\mathbb{N}_{rh}(e_t)$ *WinCounts*
5: **if** ω_a==0 **and** ω_{ah}==0 **and** ω_{rh}==0 **and** syn==$True$
6: $tipsetchain_{pub} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
7: $tipsetchain_{pub} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
8: $tipsetchain_{private} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
9: $tipsetchain_{private} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
10: **if** $\mathbb{N}_a(e_t)$!=0 //withhold
11: ω_a+=$\mathbb{N}_a(e_t)$
12: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
13: syn=$False$
14: **else** //adopt
15: $tipsetchain_{private}$=$tipsetchain_{pub}$
16: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
17: **else**
18: ω_a+=$\mathbb{N}_a(e_t)$
19: ω_{ah}+=$\mathbb{N}_{ah}(e_t)$
20: ω_{rh}+=$\mathbb{N}_{rh}(e_t)$
21: $tipsetchain_{pub} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
22: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
23: $tipsetchain_{private} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
24: **if** $\omega_a + \omega_{rh} > \omega_{ah}$ //override
25: $tipsetchain_{pub}$=$tipsetchain_{private}$
26: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
27: **else if** $\omega_a + \omega_{rh} < \omega_{ah}$ //adopt
28: $tipsetchain_{private}$=$tipsetchain_{pub}$
29: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
30: **else** //match
31: **if** $\mathcal{A}$ wins (50% probability)
32: Execute line 25, 26
33: **else** $\mathcal{H}$ wins (50% probability)
34: Execute line 28, 29

C State Transitions of *TBW attack 2.0*

D Algorithm of *TBW Attack 3.0*

E State Transitions of *TBW attack 3.0*

Table 2: Updated state transitions of *TBW Attack 2.0*. λ represents the storage power share of $\mathcal{RH}$. X represents the number of *WinCounts* that $\mathcal{A}$ and $\mathcal{RH}$ can obtain in the second consecutive withholding epoch, while Y represents the number of *WinCounts* that $\mathcal{AH}$ can obtain in the second consecutive withholding epoch.

State	Action	Resulting State	Transition Probability	Reward (r_a, r_h)
s_{Δ_1,a_k}	CT release	s'_{Δ_1,a_k}	1	$(0,0)$
s'_{Δ_1,a_k}	override	s_{Δ_2,a_i}	$\sum_{0\le X\le n,0\le Y\le n,X+k-Y=i}(\frac{(\mu*(\alpha+\lambda))^X}{X!}*e^{-\mu}*\frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(\mu*\alpha+k,\mu*\lambda)$
s'_{Δ_1,a_k}	adopt	s_{Δ_2,h_i}	$\sum_{0\le X\le n,0\le Y\le n,Y-X-k=i}(\frac{(\mu*(\alpha+\lambda))^X}{X!}*e^{-\mu}*\frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(0,\mu*(1-\alpha-\lambda))$
s'_{Δ_1,a_k}	match	$s_{\Delta_2,0'}$	$\sum_{0\le X\le n,0\le Y\le n,X+k=Y}(\frac{(\mu*(\alpha+\lambda))^X}{X!}*e^{-\mu}*\frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(\mu*\alpha+k,\mu*\lambda)$ or $(0,\mu*(1-\alpha-\lambda))$
s_{Δ_2}	CT release	s'_{Δ_1,a_k}	1	$(0,0)$

Algorithm 3 TBW Attack 3.0

$tipsetchain_{pub} \leftarrow$ the heaviest tipsetchain
$tipsetchain_{private} \leftarrow$ the heaviest tipsetchain
syn=$True$//two tipsetchains are equal
ω_a=0; ω_{ah}=0; ω_{rh}=0

1: **At** epoch e_0
2: $\mathcal{A}$ obtains $\mathbb{N}_a(e_0)$ *WinCounts*
3: $\mathcal{A}$ obtains $\mathbb{N}_a(e_1)$ *WinCounts*
4: $\mathcal{AH}$ obtains $\mathbb{N}_{ah}(e_0)$ *WinCounts*
5: $\mathcal{RH}$ obtains $\mathbb{N}_{rh}(e_0)$ *WinCounts*
6: $tipsetchain_{pub} \leftarrow \mathbb{N}_{ah}(e_0)$ *WinCounts*
7: $tipsetchain_{pub} \leftarrow \mathbb{N}_{rh}(e_0)$ *WinCounts*
8: $tipsetchain_{private} \leftarrow \mathbb{N}_{ah}(e_0)$ *WinCounts*
9: $tipsetchain_{private} \leftarrow \mathbb{N}_{rh}(e_0)$ *WinCounts*
10: **if** $\mathbb{N}_a(e_0)$!=0 **and** $\mathbb{N}_a(e_1)$!=0
11: ω_a+=$\mathbb{N}_a(e_0)$
12: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_0)$ *WinCounts*
13: syn=$False$
14: **else if** $\mathbb{N}_a(e_0)$!=0 **and** $\mathbb{N}_a(e_1)$==0
15: $tipsetchain_{pub} \leftarrow \mathbb{N}_a(e_0)$ *WinCounts*
16: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_0)$ *WinCounts*
17: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
18: **else**
19: $tipsetchain_{private}$=$tipsetchain_{pub}$
20: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
21: **At** epoch $e_t (t > 0)$
22: $\mathcal{A}$ obtains $\mathbb{N}_a(e_{t+1})$ *WinCounts*
23: $\mathcal{AH}$ obtains $\mathbb{N}_{ah}(e_t)$ *WinCounts*
24: $\mathcal{RH}$ obtains $\mathbb{N}_{rh}(e_t)$ *WinCounts*
25: **if** ω_a==0 **and** ω_{ah}==0 **and** ω_{rh}==0 **and** syn==$True$
26: $tipsetchain_{pub} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
27: $tipsetchain_{pub} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
28: $tipsetchain_{private} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
29: $tipsetchain_{private} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
30: **if** $\mathbb{N}_a(e_t)$!=0 **and** $\mathbb{N}_a(e_{t+1})$!=0
31: ω_a+=$\mathbb{N}_a(e_t)$
32: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
33: syn=$False$
34: **else if** $\mathbb{N}_a(e_t)$!=0 **and** $\mathbb{N}_a(e_{t+1})$==0
35: $tipsetchain_{pub} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
36: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
37: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
38: **else**
39: $tipsetchain_{private}$=$tipsetchain_{pub}$
40: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
41: **else**
42: ω_a+=$\mathbb{N}_a(e_t)$
43: ω_{ah}+=$\mathbb{N}_{ah}(e_t)$
44: ω_{rh}+=$\mathbb{N}_{rh}(e_t)$
45: $tipsetchain_{pub} \leftarrow \mathbb{N}_{ah}(e_t)$ *WinCounts*
46: $tipsetchain_{private} \leftarrow \mathbb{N}_a(e_t)$ *WinCounts*
47: $tipsetchain_{private} \leftarrow \mathbb{N}_{rh}(e_t)$ *WinCounts*
48: **if** $\omega_a + \omega_{rh} > \omega_{ah}$ //override
49: $tipsetchain_{pub}$=$tipsetchain_{private}$
50: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
51: **else if** $\omega_a + \omega_{rh} < \omega_{ah}$ //adopt
52: $tipsetchain_{private}$=$tipsetchain_{pub}$
53: ω_a=0;ω_{ah}=0;ω_{rh}=0;syn=$True$
54: **else** //match
55: **if** $\mathcal{A}$ wins (50% probability)
56: Execute line 49, 50
57: **else** $\mathcal{H}$ wins (50% probability)
58: Execute line 52, 53

Table 3: The updated transition probabilities of *TBW Attack 3.0*. X_1, X_2 represent the number of *WinCounts* that $\mathcal{A}$ can obtain in the first and second consecutive withholding epoch ($X_1 + X_2 = k$), and Z, Y note the number of *WinCounts* that $\mathcal{RH}, \mathcal{AH}$ can obtain in the second consecutive withholding epoch.

State	Action	Resulting State	Transition Probability	Reward (r_a, r_h)
s_0	adopt	s_0	$2 * e^{-\mu*\alpha} - e^{-2*\mu*\alpha}$	$(0, \mu*(1-\alpha))$
s_0	FEP withhold	s_{Δ_1,a_k}	$\sum_{0<X_1<n, 0<X_2<n, X_1+X_2=k} (\frac{(\mu*\alpha)^{X_1}}{X_1!} * e^{-2*\mu*\alpha} * \frac{(\mu*\alpha)^{X_2}}{X_2!})$	$(0, \mu*(1-\alpha))$
s_{Δ_1,a_k}	CT release	s'_{Δ_1,a_k}	1	$(0,0)$
s'_{Δ_1,a_k}	override	s_{Δ_2,a_i}	$\sum_{0\le Z\le n, 0\le Y\le n, k+Z-Y=i} (\frac{(\mu*\lambda)^Z}{Z!} * e^{-\mu*(1-\alpha)} * \frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(k, \mu*\lambda)$
s'_{Δ_1,a_k}	adopt	s_{Δ_2,h_i}	$\sum_{0\le Z\le n, 0\le Y\le n, Y-Z-k=i} (\frac{(\mu*\lambda)^Z}{Z!} * e^{-\mu*(1-\alpha)} * \frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(0, \mu*(1-\alpha-\lambda))$
s'_{Δ_1,a_k}	match	$s_{\Delta_2,0'}$	$\sum_{0\le Z\le n, 0\le Y\le n, k+Z=Y} (\frac{(\mu*\lambda)^Z}{Z!} * e^{-\mu*(1-\alpha)} * \frac{(\mu*(1-\alpha-\lambda))^Y}{Y!})$	$(k, \mu*\lambda)$ or $(0, \mu*(1-\alpha-\lambda))$
s_{Δ_2}	adopt	s_0	$2 * e^{-\mu*\alpha} - e^{-2*\mu*\alpha}$	$(0, \mu*(1-\alpha))$
s_{Δ_2}	FEP withhold	s_{Δ_1,a_k}	$\sum_{0<X_1<n, 0<X_2<n, X_1+X_2=k} (\frac{(\mu*\alpha)^{X_1}}{X_1!} * e^{-2*\mu*\alpha} * \frac{(\mu*\alpha)^{X_2}}{X_2!})$	$(0, \mu*(1-\alpha))$