Two More Attacks on Proof-of-Stake GHOST/Ethereum

Joachim Neu
Stanford University
Stanford, CA, USA
jneu@stanford.edu

Ertem Nusret Tas
Stanford University
Stanford, CA, USA
nusret@stanford.edu

David Tse
Stanford University
Stanford, CA, USA
dntse@stanford.edu

ABSTRACT

Ethereum, the world's second largest cryptocurrency with a market capitalization exceeding 120 billion USD as of this writing, aims to switch from Proof-of-Work (PoW) to Proof-of-Stake (PoS) based consensus later in the year 2022 ('the Merge'). Yet, so far, the proposed PoS consensus protocol lacks in rigorous security analysis. We present two new attack strategies targeting the PoS Ethereum consensus protocol. The first attack suggests a fundamental conceptual incompatibility between PoS and the Greedy Heaviest-Observed Sub-Tree (GHOST) fork choice paradigm employed by PoS Ethereum. In a nutshell, PoS allows an adversary with a vanishing amount of stake to produce an unlimited number of equivocating blocks. While most equivocating blocks will be orphaned, such orphaned 'uncle blocks' still influence fork choice under the GHOST paradigm, bestowing upon the adversary devastating control over the canonical chain. While the Latest Message Driven (LMD) aspect of current PoS Ethereum prevents a straightforward application of this attack, our second attack shows how LMD specifically can be exploited to obtain a new variant of the balancing attack that overcomes 'proposer boosting', a recent protocol addition that was intended to mitigate balancing-type attacks. Thus, in its current form, PoS Ethereum without and with LMD is vulnerable to our first and second attack, respectively.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

KEYWORDS

Proof-of-stake, Ethereum, GHOST, avalanche, balancing, attack

ACM Reference Format:

Joachim Neu, Ertem Nusret Tas, and David Tse. 2022. Two More Attacks on Proof-of-Stake GHOST/Ethereum. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus (ConsensusDay '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3560829.3563560

1 INTRODUCTION

Ethereum [5] is the world's second largest cryptocurrency with a market capitalization exceeding 120 billion US dollars as of this writing. It furthermore provides consensus security for an ecosystem of fungible and non-fungible tokens and synthetic assets worth an additional estimated hundreds of billions of US dollars [4]. The Ethereum protocol is targeted to switch from Proof-of-Work (PoW) to Proof-of-Stake (PoS) consensus later in 2022 ('the Merge'). Yet, so far, the proposed PoS consensus protocol still lacks in rigorous security analysis. This manuscript contributes to this in the negative by presenting two new attack strategies.

The currently proposed PoS Ethereum consensus protocol [1–3] is constructed from an application of the finality gadget Casper FFG [7] on top of the fork choice rule LMD GHOST, a variant of the Greedy Heaviest-Observed Sub-Tree (GHOST) [23] rule which considers only each participant's most recent vote (Latest Message Driven, LMD). Subsequently, we refer as *validators* to participants with stake that allows them to vote as part of the protocol. A slightly simplified and analytically more tractable variant of the proposed PoS Ethereum protocol is given by the Gasper protocol [8]. The protocol is recapitulated on a high level in Section 2.

This manuscript continues a line of earlier works by various authors [9, 14, 15, 17–22] that highlighted fundamental flaws in various versions of the consensus mechanism underlying PoS Ethereum. Earlier works have predominantly targeted the fork choice rule LMD GHOST (which itself can be viewed as a consensus protocol, cf. Section 2; [17–20, 22]) as well as the interaction between LMD GHOST and the finality gadget Casper FFG [9, 14, 15, 21]. In this work, we report two new attack strategies on LMD GHOST.

The first attack, which we call *avalanche attack* and describe in Section 3, suggests a fundamental conceptual incompatibility between PoS on the one hand and the GHOST fork choice paradigm on the other hand. In a nutshell, so called 'orphaned' 'uncle blocks' off the canonical chain still influence the GHOST fork choice. In PoW, for which GHOST was originally conceived [23] and proven secure [13] under synchrony for a sufficiently small mining rate[2], this does not cause problems because the number of uncle blocks which the adversary controls and by which it can influence the fork choice is globally bounded by the PoW mechanism. PoS, on the other hand, allows the adversary to equivocate, *i.e.*, for each block production opportunity the adversary can produce an unlimited number of equivocating blocks with different contents and/or extending different parent blocks of the block tree. (Note that this is necessarily so, for if the block content was input to the PoS block production 'lottery', then this would give the adversary an

JN and ENT contributed equally and are listed alphabetically.

ConsensusDay '22, November 7, 2022, Los Angeles, CA, USA.

ACM ISBN 978-1-4503-9879-4/22/11…$15.00
https://doi.org/10.1145/3560829.3563560

[2]The balance attack suggested for PoW GHOST in [16] requires communication to be sufficiently delayed between two or more subgraphs of mining nodes (effectively, a period of asynchrony) so that the adversary has the opportunity to balance the trees built by the disconnected nodes. On the other hand, [13] has shown that PoW GHOST is secure under a synchronous network with a constant bound Δ on delay and a sufficiently small mining rate with respect to this delay. Unlike the attack in [16] that relies on prolonged delays between different sets of nodes, the attacks presented in this work as well as the earlier balancing attacks on LMD GHOST [17–20, 22] can happen under a synchronous network with a constant Δ.

opportunity to 'grind' on the payload, *e.g.*, try different permutations of transactions, and thus boost its chances to produce a block, consequently degenerating the protocol back to PoW.) As a result, GHOST's way of accounting for uncle blocks in the fork choice gives the adversary much more influence over fork choice in PoS than in PoW. Specifically, an adversary can use a single block production opportunity to issue multiple equivocating blocks with which it can influence the GHOST fork choice rule at various steps simultaneously. This highlights a fundamental conceptual incompatibility between GHOST (where orphaned uncle blocks influence fork choice) and PoS (where the adversary can produce an unbounded number of such uncle blocks), and casts doubt over whether GHOST should be used with PoS at all. Our avalanche attack shows specifically how the adversary can use equivocating blocks to subvert consensus security.

The LMD aspect of the current PoS Ethereum protocol interferes with a naive application of the avalanche attack to PoS Ethereum. However, our second attack, described in Section 4, shows how the LMD feature specifically can be exploited to obtain a new variant of the former balancing attacks (on LMD GHOST) outlined in [17, 18]. Namely, while previous balancing attacks relied on *perpetual* careful timing of adversarial messages to keep honest nodes split equally among two competing chains [18, 22], LMD is now used to enshrine a *permanent* split view among honest nodes (at most with *one-time* help of network timing [22]). Thus, consensus is broken with considerably lower requirements in terms of adversarial stake and network timing control/predictability than previous attacks. This new attack is furthermore of interest as it overcomes 'proposer boosting' [6], a recent PoS Ethereum protocol addition that was specifically introduced with the intention to thwart balancing-type attacks.

Thus, in its current form, PoS Ethereum without LMD is vulnerable to our first new attack (avalanche attack), and PoS Ethereum with LMD is vulnerable to our second new attack (new LMD-specific variant of previous balancing-type attacks). What is more, the 'design-philosophical' incompatibilities between PoS and GHOST as highlighted by the avalanche attack fundamentally call into question the attempt to combine PoS and GHOST—an insight that might be of interest beyond the currently ongoing PoS Ethereum design and specification efforts.

2 THE PROOF-OF-STAKE ETHEREUM CONSENSUS PROTOCOL

Let Ledger_i^t denote a transaction ledger output by honest validator i at time t. For *security* of a ledger output by a consensus protocol one usually requires *safety* and *liveness*:

- **Safety:** For any given times t, t' and honest validators i, j, either Ledger_i^t is a prefix of $\mathsf{Ledger}_j^{t'}$, or vice versa. Less formally, the ledgers output by two honest validators at two points in time are consistent with each other.
- **Liveness:** If a transaction tx is received by all honest validators by some time t, then tx appears in $\mathsf{Ledger}_i^{t'}$ for any time $t' \geq t + T_{\mathsf{conf}}$ and for any honest validator i, where T_{conf} is the protocol's *confirmation time*. Less formally, transactions get confirmed in honest validators' ledgers with at most T_{conf} time delay.

Safety can be strengthened to *accountable safety* [7, 8, 10, 20]:

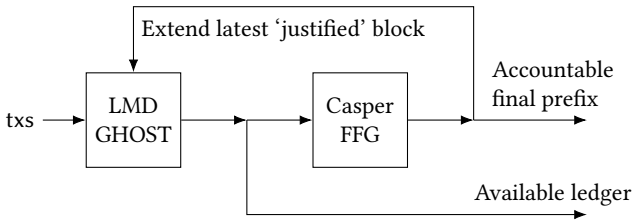

Figure 1: PoS Ethereum [8] consists of a concatenation of consensus protocols LMD GHOST and Casper FFG [7]. Its goal [19, 20] is to produce an available full ledger that remains live under dynamic participation of validators, and a prefix ledger that remains accountably safe under network partition. For this purpose, a finality/accountability gadget (such as Casper FFG) checkpoints decisions of an available protocol (which LMD GHOST was supposed to be). A feedback loop is introduced to ensure that the available protocol respects earlier checkpoint decisions. Multiple works have uncovered attacks on LMD GHOST [17–20, 22] and on the two-way interaction between LMD GHOST and Casper FFG [9, 14, 15, 21].

- **Accountable safety:** If there exist times t, t' and honest validators i, j, such that Ledger_i^t and $\mathsf{Ledger}_j^{t'}$ are not a prefix of one another (*i.e.*, there is a safety violation), then as many validators as the ledger's safety resilience can be identified to have provably violated the protocol.
  Less formally, there cannot be a safety violation if fewer validators than the safety resilience are adversarial; and if there is a safety violation, then more validators than the safety resilience must be adversarial, and one can obtain cryptographic proof that irrefutably identifies at least as many validators as the safety resilience as protocol violators.

On a high level, PoS Ethereum [8] consists of a concatenation of consensus protocols LMD GHOST and Casper FFG [7] (Figure 1). Its objective [19, 20] is to produce two nested ledgers with different security properties, namely an available full ledger that remains *live* under dynamic participation of validators (and *safe* as long as there is no network partition) assuming less than 50% adversarial stake, and a prefix ledger that remains *accountably safe* under network partition (and *live* once the network partition heals and all validators participate) assuming less than 33% adversarial stake. For this purpose, a finality/accountability gadget (such as Casper FFG) checkpoints decisions of an available protocol (which LMD GHOST was supposed to be). A feedback loop is introduced to ensure that the available protocol respects earlier checkpoint decisions. Multiple earlier works have uncovered attacks on LMD GHOST [17–20, 22] and on the two-way interaction between LMD GHOST and Casper FFG [9, 14, 15, 21]. This work provides two new attacks on LMD GHOST.

In what follows, we thus focus on PoS Ethereum's 'LMD GHOST component'. We first describe a simplified version of PoS Ethereum's 'LMD GHOST component' without the LMD rule, reduced to the relevant fork choice mechanics. Although we present a self-contained description, the reader might want to recall the protocols GHOST

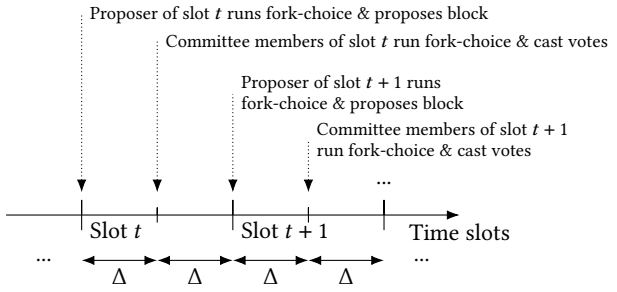

Figure 2: Committee-GHOST has time slots of 2Δ length. Each time slot has a unique proposer and a committee of validators. At the start of each slot, the proposer determines their canonical chain and proposes and broadcasts a block extending it. Half-way into each slot, each committee member determines their canonical chain and casts a vote on the tip.

[23] and Gasper [8], as well as the beacon chain's fork choice specification [3] and the idea of proposal weights[3] [6], to further consolidate their understanding of the material of this section.

In the consensus protocol (which we call Committee-GHOST; Figure 2), time proceeds in synchronized slots of duration 2Δ, since it is assumed that message delay between honest validators is bounded by Δ. For each slot, one *proposer* and a *committee* of W validators are drawn independently and uniformly at random (without replacement for the committee members) from the N validators. We say a slot is honest or adversarial if the corresponding block proposer is honest or adversarial, respectively. The following fork-choice rule is used in the view of validator i at slot t to determine a canonical block and its prefix of blocks as the *canonical chain*:

- Starting at the genesis block b_0, sum for each child block b the number of unique valid votes for that block and its descendents. Here, unique means counting only one vote per committee member in each slot $s < t$. Valid means that the vote from a committee member of slot $s < t$ was cast for a block produced in a slot $s' \leq s$. Ties are broken by the adversary.
  Add a proposal boost of W_p to the score of b if b or one of its descendants is a valid proposal from the current slot t. Here, valid means that the block is not from a future slot, was produced by the proposer of slot t, and that proposal time slots along block chains are strictly increasing.
- Pick the child block b^* with highest score (cf. GHOST rule [23]), breaking ties adversarially.
- Recurse ($b_0 \leftarrow b^*$) until reaching a leaf block. Output that block and its prefix as the canonical chain.

At the beginning of each slot, the slot's proposer determines a tip using the fork-choice rule and extends it with a new proposal. Half way into each slot (*i.e.*, Δ time after the proposal and after the beginning of the slot), the slot's committee members determine a tip using the fork-choice rule in their local view and vote for it. The unit of time is a time slot. A block from slot t and its prefix are confirmed and output as the ledger if and only if at time $t + T_\mathsf{conf} + \frac{1}{2}$ (*i.e.*, at the time of voting in slot $t + T_\mathsf{conf}$), the block is in the chain determined by the fork-choice rule in the view of the respective honest validator. Here, T_conf is the *confirmation time*.

For simplicity, we assume a fraction β such that (a) for any given slot the probability of the block proposer being adversarial is at most β, and (b) the fraction of adversarial validators in any committee is at most β throughout the protocol's execution.

The LMD rule modifies the above protocol as follows. Every validator keeps a table of 'latest votes' received from the other validators, in the following manner:[4] When a valid vote from a validator is received, then the table entry for that validator is updated, if and only if the new vote is from a slot *strictly later than* the current entry. Hence, when a validator observes a slot-t vote from the committee member i of some slot t, it records this vote and ignores all subsequent slot-t' votes for $t' \leq t$ by i. Thus, if a validator receives two equivocating votes from the same validator for the *same* time slot, the validator counts only the vote received earlier in time.

3 AVALANCHE ATTACK ON PROOF-OF-STAKE GHOST

We describe a generic attack on PoS GHOST variants. This points to conceptual issues with the combination of PoS and GHOST, awareness of which might be of interest beyond PoS Ethereum fork-choice design and specification efforts.

3.1 High Level Description

The *avalanche attack* on PoS GHOST combines *selfish mining* [12] with *equivocations*. The adversary uses withheld blocks to displace an honest chain once it catches up in sub-tree weight with the number of withheld adversarial blocks. The withheld blocks are released in a flat but wide sub-tree, exploiting the fact that under the GHOST rule such a sub-tree can displace a long chain. Only two withheld blocks enter the canonical chain permanently, while the other withheld blocks are subsequently reused through equivocations to build further sub-trees to displace even more honest blocks. The attack exploits a specific weakness of the GHOST rule in combination with equivocations, namely that an adversary can reuse copies of 'uncle blocks' in GHOST, and thus such equivocations contribute to the weight of multiple ancestors. This casts doubt over whether GHOST should be used with PoS at all.

We also provide a proof-of-concept implementation for vanilla PoS GHOST and Committee-GHOST.[5] By 'vanilla PoS GHOST', we mean a one-to-one translation of GHOST [23] from proof-of-work lotteries to proof-of-stake lotteries. In that case, every block comes with unit weight. By 'Committee-GHOST' we mean a vote-based variant of GHOST as specified in Section 2 and used in PoS Ethereum, where block weight is determined by votes and potentially a proposal boost [6]. Subsequently, we first illustrate the attack with an example, then provide a more detailed description, show plots produced by our proof-of-concept implementation, and finally conclude with a comprehensive analysis.

[3] https://github.com/ethereum/consensus-specs/pull/2730

[4] https://github.com/ethereum/consensus-specs/blob/72d45971310a24f6e5ecfb149d23c9fac4c7878a/specs/phase0/fork-choice.md#update_latest_messages

[5] Source code: https://github.com/tse-group/pos-ghost-attack

Algorithm 1 Pseudocode for avalanche attack on PoS GHOST variants. ReleaseAdversarialSubtree releases a flat-but-wide subtree of adversarial blocks as shown in Figure 3(b), and keeps equivocations of those withheld blocks that can be subsequently reused. The attack applies analogously to Committee-GHOST, where also adversarial votes are withheld.

$\mathcal{W} \leftarrow \emptyset$ ▷ Withheld adversarial blocks
for time slot $t = 1, 2, 3, \ldots$ **do**
  ▷ Rushing adversary
  $W_t^{\mathrm{h}} \leftarrow$ weight of honest subtree at beginning of t
  $W_{(t+1)^-}^{\mathrm{h}} \leftarrow$ weight of honest subtree at end of t
  $W_t^{\mathrm{a}} \leftarrow$ weight of adversarial subtree in $\mathcal{W}$ at beginning of t
  if $(W_t^{\mathrm{h}} \leq W_t^{\mathrm{a}}) \wedge (W_t^{\mathrm{a}} < W_{(t+1)^-}^{\mathrm{h}})$ **then**
    $\mathcal{W} \leftarrow$ ReleaseAdversarialSubtree$(\mathcal{W})$
  else if $W_t^{\mathrm{a}} < W_t^{\mathrm{h}}$ **then**
    ▷ Attack is over :(— Retry!
  else
    ▷ Do nothing in this time slot
  end if
  if time slot t is adversarial **then**
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{t\}$
  end if
  HonestNodesActions() ▷ Honest nodes' actions as per protocol
end for

3.2 A Simple Attack Example

We illustrate the attack using a simple example where the adversary starts with $k = 6$ withheld blocks and does not gain any new blocks during the attack. In this case, the attack eventually runs out of steam and stops. (In reality, the larger the number of withheld blocks, the more likely the attack continues practically forever, and even for low k that probability is not negligible.) Still, the example illustrates that $k = 6$ blocks are enough for the adversary to displace 12 honest blocks.

First, the adversary withholds its flat-but-wide sub-tree of $k = 6$ withheld blocks, while honest nodes produce a chain (Figure 3(a)). (Green and red indicate honest and adversarial blocks, respectively, and the numbers on blocks indicate which block production opportunity of honest/adversary they correspond to.) Once honest nodes have built a chain of length $k = 6$, the adversary releases the withheld blocks, and displaces the honest chain (Figure 3(b)). Honest nodes build a new chain on top of $2 \rightarrow 1 \rightarrow$ Genesis (following adversarial tie-break). Note that the adversary can reuse its blocks 3, 4, 5, 6. Once that new chain reaches length 4, the adversary releases another displacing sub-tree (Figure 3(c)). The adversary again reuses its blocks 5, 6, and the honest nodes build a new chain on top of $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow$ Genesis. Once the new chain reaches length 2, the adversary releases the last displacing sub-tree (Figure 3(d)). Honest nodes now build on $6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow$ Genesis. As all honest blocks so far have been displaced, if this attack persisted for a longer time, it would constitute both a liveness and safety violation. Overall, with this strategy, the adversary gets to displace $\Theta(k^2)$ honest blocks with k withheld adversarial blocks.

3.3 Attack Details

The attack uses an 'avalanche of equivocating sub-trees rolling over honest chains'. The following description is for vanilla PoS GHOST (cf. Algorithm 1), but can be straightforwardly translated for Committee-GHOST with proposer boosting [6].

Suppose an adversary gets k block production opportunities in a row, for a modest k. The adversary withholds these k blocks, as in *selfish mining* (Figure 3(a)). On average, more honest blocks are produced than adversary blocks, so the developing honest chain eventually 'catches up' with the k withheld adversarial blocks. In that moment, the adversary releases these k blocks, not as a competing adversarial chain (as in selfish mining for a Longest Chain protocol), but as an adversarial sub-tree of height 2, where all but the first withheld block are the children of the first withheld block. Due to the GHOST weight counting, this adversarial sub-tree is now of equal weight as the honest chain—so the honest chain is abandoned (Figure 3(b)). At the same time, ties between equal-weight sub-trees are broken such that honest nodes from now on build on what was the second withheld block. This allows the adversary to reuse in the form of *equivocations* the copies of withheld blocks 3, 4, ..., k on top of the chain $2 \rightarrow 1 \rightarrow$ Genesis formed by the first two withheld adversarial blocks, which is now the chain adopted by the honest nodes.

So far, the adversary started with k withheld blocks, has used those to displace k honest blocks, and is now left with equivocating copies of $k - 2$ adversarial withheld blocks that it can still reuse through further equivocations (Figure 3(c)). In addition, while the k honest blocks were produced, the adversary likely had a few block production opportunities of its own, which get added to the pool of adversarial withheld blocks. Note that the attack has renewed in favor of the adversary if the adversary had two new block production opportunities, making up for the two adversarial withheld blocks lost because their copies cannot be reused. The process is now repeated (Figure 3(d)): Whenever honest nodes build a chain of weight equal to the number of blocks withheld by the adversary, then the adversary releases a competing sub-tree of height 2. The chain made up from the first two released withheld blocks is adopted by the honest nodes, the other block production opportunities can still be reused in the future through equivocations on top of it and thus remain in the pool of adversary's withheld blocks.

3.4 Analysis

For any given adversarial stake $\beta > 0$, there exists a k such that if the adversary starts out with k withheld blocks k, then the adversary gains 2 block production opportunities during the production of the k honest blocks that will be displaced subsequently. Thus, the process renews or even drifts in favor of the adversary. No honest blocks enter the canonical chain permanently—a liveness failure, and all honest blocks are eventually removed from the canonical chain—a safety failure for any fixed confirmation time T_{conf}. This observation is formalized by the following theorem:

Theorem 3.1. *For both vanilla PoS GHOST and Committee-GHOST, there exists a constant $t < \infty$ such that after slot t, the avalanche attack is sustained forever.*

Proof. Suppose the adversary releases blocks from its repository of withheld blocks to attack honest blocks produced over

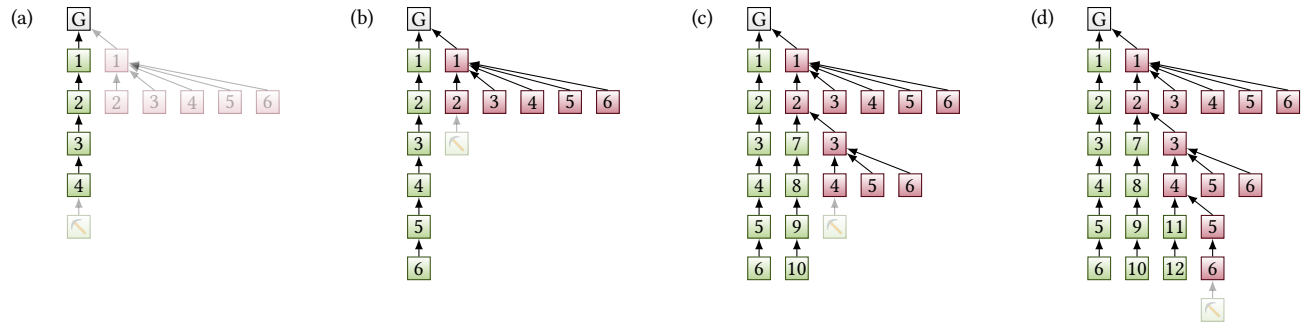

Figure 3: Green/red are honest/adversarial blocks, respectively, and the numbers on blocks indicate which block production opportunity of honest/adversary they correspond to. (a) First, the adversary withholds its flat-but-wide sub-tree of $k = 6$ withheld blocks, while honest nodes produce a chain. (b) Once honest nodes have built a chain of length $k = 6$, the adversary releases the withheld blocks, and displaces the honest chain. (c) Note that the adversary can reuse its blocks from the adversarial block production opportunities $3, 4, 5, 6$. Honest nodes build a new chain on top of $2 \longrightarrow 1 \longrightarrow$ Genesis. Once that new chain reaches length 4, the adversary releases another displacing sub-tree. (d) Finally, note that the adversary can reuse its blocks $5, 6$. Honest nodes build a new chain on top of $4 \longrightarrow 3 \longrightarrow 2 \longrightarrow 1 \longrightarrow$ Genesis. Once the new chain reaches length 2, the adversary releases the last displacing sub-tree.

periods of k slots. It loses two of the released blocks upon each release as these blocks enter the canonical chain while the remaining ones can be reused. Let X_i, $i = 1, 2, \ldots$, denote the number of withheld adversarial blocks or reusable equivocations at the beginning of each period. The attack continues over multiple periods of k slots if the adversary has at least k withheld or reusable blocks at the beginning of each period, *i.e.*, the attack is sustained over at least m periods if $X_i \geq k$ for $i = 1, 2, \ldots, m$. By definition, $X_0 = k$, and since each slot is adversarial with probability β, $\mathbb{E}[X_i - X_{i-1}] = \beta k - 2$ for all $i = 1, 2, \ldots, m$.

As the identities of the block proposers per slot are independent, the values of X_k constitute the states of a random walk. Moreover, we observe that

- If $k > 2/\beta$, expectation of $X_i - X_{i-1}$, $i = 1, 2, \ldots$, is positive.
- Expectation of $|X_i - X_{i-1}|$, $i = 1, 2, \ldots, m$, is finite.

Then, due to the Strong Law of Large Numbers, every state of this random walk is transient, and the random walk has a positive drift. Hence, starting at $X_0 = k$, the event of X_k never hitting or falling below k has positive probability $p \in (0, 1)$. This implies that with probability $p > 0$, the attack is sustained over infinite number of periods after it starts.

In the case of the Committee-GHOST protocol, adversarial validators vote only for adversarial blocks and withhold their votes for those blocks along with the blocks themselves as part of the avalanche attack. Otherwise, they follow the attack described above, starting with $(1-\beta)k/\beta$ withheld blocks. Let X_i, $i = 1, 2, \ldots$, denote the number of votes on withheld adversarial blocks or still reusable equivocations at the beginning of each period. The attack continues over multiple periods of k slots if the adversary has at least $(1-\beta)kW$ votes on withheld or reusable blocks at the beginning of each period, *i.e.*, if $X_i \geq (1-\beta)kW$ for $i = 1, 2, \ldots, m$. Adversary loses two blocks and $2\beta W$ votes upon each release as these blocks enter the canonical chain while the remaining ones can be reused. It gains in expectation βk block production opportunities and β votes with each adversarial block over periods of k slots. Then, by definition, $X_0 = (1-\beta)kW$, and $\mathbb{E}[X_i - X_{i-1}] = (k\beta - 2)\beta W$ for all $i = 1, 2, \ldots, m$.

Finally, as the identities of the block proposers per slot are independent, the values of X_k constitute the states of a random walk. Moreover, we observe that

- If $k > 2/\beta$, expectation of $X_i - X_{i-1}$, $i = 1, 2, \ldots$, is positive.
- Expectation of $|X_i - X_{i-1}|$, $i = 1, 2, \ldots, m$, is finite.

Then, due to the Strong Law of Large Numbers, every state of this random walk is again transient, and, the random walk has a positive drift. Hence, starting at $X_0 = (1-\beta)kW$, the event of X_k never hitting or falling below k has positive probability $p \in (0, 1)$. This implies that with probability $p > 0$, the attack is sustained over infinite number of periods after it starts.

Starting at any given slot, adversary has k consecutive block production opportunities with probability β^k. As the attack succeeds after each such $k > 2/\beta$ slots with probability $p > 0$, by the Borel-Cantelli lemma, almost surely, there exists a slot after which both attacks are sustained forever on the respective protocols. □

3.5 Proof-of-Concept Implementation

For illustration, we plot a snapshot of the block tree resulting after 100 time slots in our proof-of-concept implementation[6]—see Figure 4 for PoS GHOST and Figure 5 for Committee-GHOST. The attack is still ongoing thereafter, and as long as the attack is sustained, no honest blocks remain in the canonical chain permanently. The length of the displaced honest chains in Figures 4 and 5 increases over time. This is because during each attack cycle the adversary gains more than two block production opportunities. Thus, the number of withheld adversarial blocks increases over time, enabling the adversary to displace increasingly longer honest chains.

[6] Source code: https://github.com/tse-group/pos-ghost-attack

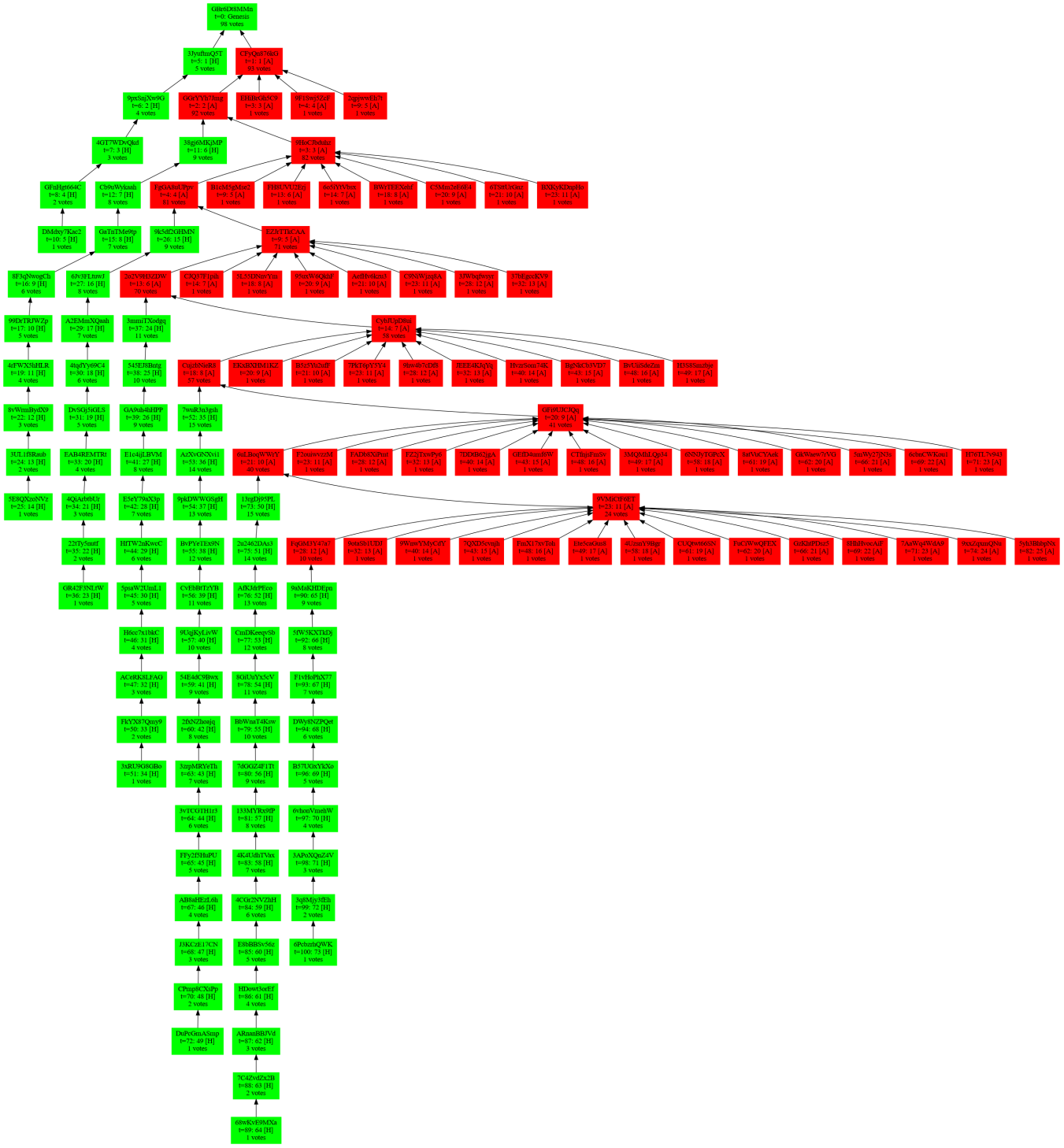

Figure 4: Snapshot of block tree resulting after 100 time slots in proof-of-concept implementation of avalanche attack on PoS GHOST (adversarial blocks: red, honest blocks: green; adversarial stake: 30%, initially withheld adversarial blocks: 4)

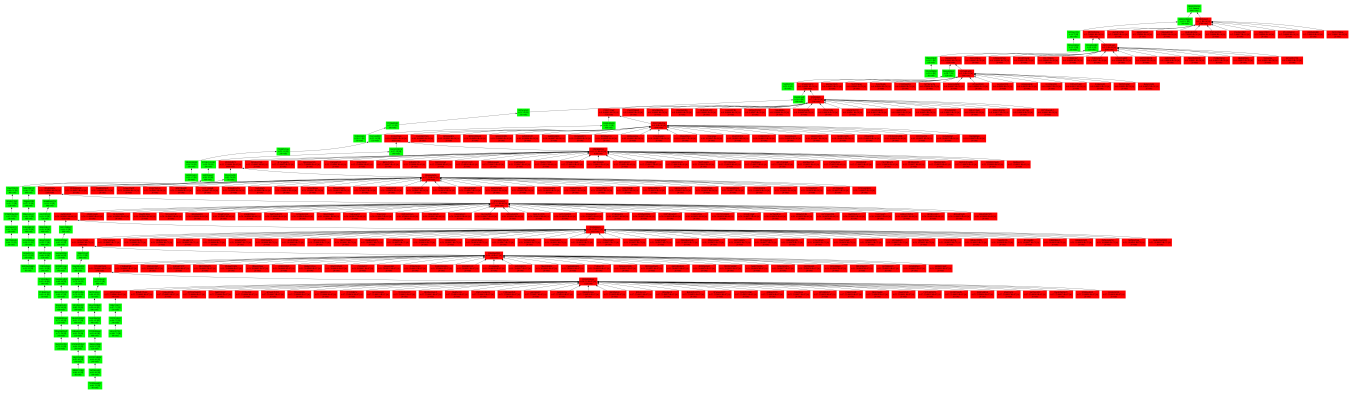

Figure 5: Snapshot of block tree after 100 time slots in proof-of-concept implementation of avalanche attack on Committee-GHOST (adversarial blocks: red, honest blocks: green; adversarial stake: 20%, initially withheld adversarial blocks: 12)

3.6 Applicability to PoS Ethereum

Section 2 gives a description of the Committee-GHOST protocol based on the consensus protocol of PoS Ethereum, albeit without the LMD aspect. The avalanche attack works on this protocol as well as shown by Theorem 3.1. In particular, an adversary controlling any positive fraction of validators can still replace $\Theta(k^2)$ honest blocks with k withheld blocks in this protocol.

PoS Ethereum's LMD (Latest Message Driven) aspect interferes with the avalanche attack, but comes with its own challenges, as shown in Section 4. Nevertheless, the avalanche attack suggests a fundamental conceptual incompatibility between PoS, and GHOST, casting doubt over whether GHOST should be used with PoS at all.

4 LMD-SPECIFIC BALANCING ATTACK ON POS ETHEREUM

Proposal weights (also called 'proposer boosting') were suggested [6] and implemented[7] to mitigate earlier balancing attacks [17, 18]. However, we show that the LMD aspect of PoS Ethereum's fork choice enables a new balancing attack even with proposal weights. This is particularly dire because PoS GHOST without LMD is susceptible to the avalanche attack, as described in Section 3. A version of PoS Ethereum fork choice with the LMD rule is described in Section 2.

4.1 Preliminaries

The original balancing attack [17–19] consists of two steps: First, the adversarial block proposers initiate two competing chains, hereafter called Left and Right. They can do this by proposing two conflicting blocks for the same height when they are elected as a block proposer. Then, few adversarial votes per slot, released at carefully selected intervals, suffice to split the honest validators of each slot into two sets voting for the two conflicting chains. This enables the adversary to maintain two conflicting and constantly growing chains, thus violate the safety of LMD GHOST.

It is feasible for an adversary to adjust the timing of the release of its votes to the network such that roughly half of the honest validators receive one message first, and the other half receives the other message first. This certainly holds in the setting of adversarial network propagation delay [17] but also in the weaker setting of random network propagation delay [18].

[7] https://github.com/ethereum/consensus-specs/pull/2730

4.2 High Level Description

The LMD rule described in Section 2 gives the adversary a remarkable power in a balancing attack: Once the adversary has set up two competing chains, it can equivocate on them. The release of these equivocating votes can be timed such that the vote for Left is received by half of honest validators first, and the vote for Right is received by the other half. Honest validators are split in their views concerning the 'latest messages' from adversarial validators. Even though all validators will soon have received both votes, the split view persists for a considerable time due to the LMD rule, and the fact that the adversarial validators release no votes for later slots.

As a result, half of the honest validators see Left as leading, and proceed to vote for it in later slots; half see Right as leading, and vote for it. But since the honest validators are split roughly in half, their votes balance, and they continue to see their respective chain as leading. Here, the adversary might have to release a few votes every now and then to counteract any drift stemming from an imbalance on the chains different honest validators see as leading. However, the disagreement among the two sets of honest validators voting on the two conflicting chains is so persistent that it can only be overcome using proposer boosting if the proposal weight exceeds the adversary's equivocating votes, which is some fraction of the committee size, by more than a constant factor. Otherwise, if the adversary controls sufficiently many slots for once over the running time of the protocol, it can collect the equivocating votes into two blocks, and using the total weight of these blocks, surpass the proposer boost once and for all. In that case, the block that collected the adversarial votes from these earlier slots effectively overpowers the committees by far, thus eliminating the purpose of committees.

4.3 A Simple Example

Let $W = 100$ denote the number of validators per slot. Suppose the proposal weight is $W_p = 0.7W = 70$, and the fraction of adversarial validators is $\beta = 0.2$. Furthermore, for simplicity, assume that the

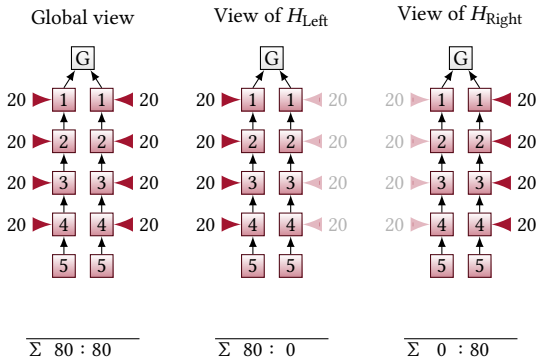

Figure 6: By the LMD rule, validators in H_{Left} and H_{Right} believe that Left **and** Right **have** 80 **votes, respectively. They also believe that the respective other chain has** 0 **votes (as the later arriving votes are not considered due to LMD).**

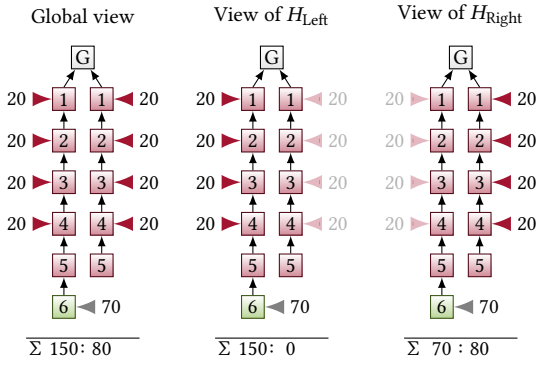

Figure 7: Suppose the validator of slot 6 **is honest and from set H_{Left}. Then, it proposes a block extending** Left**, which gains a proposal boost equivalent to** 70 **votes. Due to the proposal boost, validators in H_{Left} see** Left **as leading with** 150 **votes and vote for it. Validators in H_{Right} see** Left **has** 70 **votes while** Right **has** 80 **votes, so they vote for** Right**.**

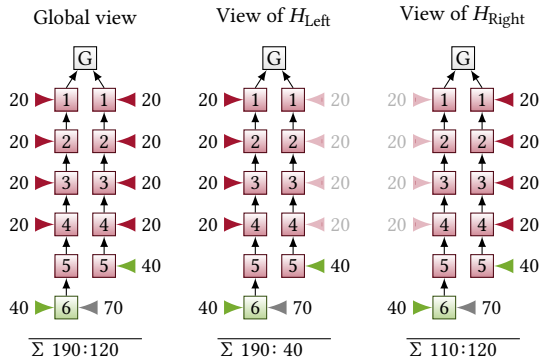

Figure 8: As a result of the split view, the vote of slot 6 **is tied—**Left **increases by roughly half of honest votes (here** 40**) and** Right **increases by roughly half of honest votes (here** 40**).**

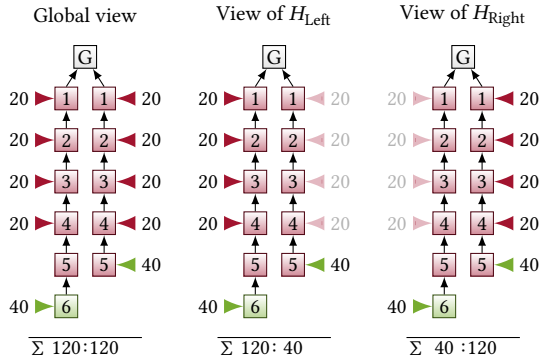

Figure 9: At the end of slot 6**, the proposer boost disappears. In the view of each honest validator, both chains gained roughly the same amount of votes, namely half of the honest validators' votes. Assuming a perfect split of $|H_{\text{Left}}| = |H_{\text{Right}}| = 40$,** Left:Right **is now** 120 : 40 **in the view of H_{Left} and** 40 : 120 **in the view of H_{Right} (up from** 80 : 0 **and** 0 : 80**, respectively). The pattern of Figures 6–9 repeats in subsequent slots, with the honest validators in H_{Left} and H_{Right} solely voting for the chains** Left **and** Right**, respectively, thus maintaining a balance of weights (in global view–in the LMD view of each validator, they keep voting for the chain they see leading, and 'cannot understand' why other honest validators keep voting for the other chain) and perpetuating the adversarially induced split view.**

attack starts when there are five consecutive slots with adversarial proposers.

During the first four slots, the adversary creates two parallel chains Left and Right of 4 blocks each, which are initially kept private from the honest validators. Each block is voted on by the 20 adversarial validators from its slot. Thus, there are equivocating votes for the conflicting blocks proposed at the same slot. For the fifth slot, the adversary includes all equivocating votes for the Left chain into a block and attaches it on the Left chain; and all the equivocating votes for the Right chain into an equivocating block and attaches it on the Right chain. With this, votes are 'batched' in the following sense. The adversary releases the two equivocating blocks from the fifth slot in such a way that roughly half of the honest validators see the Left block first (call H_{Left} that set of honest validators) and with it all the equivocating votes for the Left chain; and half of the honest validators see the Right block first (call H_{Right} that set of honest validators) and with it all the equivocating votes for the Right chain (Figure 6). (Note that this trick is not needed in networks with adversarial delay, where the release of equivocating votes can be targeted such that each honest validator either sees all Left votes first or all Right votes first.) By the LMD rule, validators in H_{Left} and H_{Right} believe that Left and Right have 80 votes, respectively. They also believe that the respective other chain has 0 votes as the later arriving votes are not considered due to LMD (Figure 6).

Now suppose the validator of slot 6 is honest and from set H_{Left}. Then, it proposes a block extending Left. Left gains a proposal boost equivalent to 70 votes (Figures 6 and 7). Thus, validators in H_{Left} see Left as leading with 150 votes and vote for it. Validators in H_{Right}

believe that Left has 70 votes while Right has 80 votes, so they vote for Right. As a result, their vote is tied—Left increases by roughly half of honest votes and Right increases by roughly half of honest votes (Figure 8). At the end of the slot, the proposer boost disappears. In the view of each honest validator, both chains gained roughly the same amount of votes, namely half of the honest validators' votes (Figure 9). Assuming a perfect split of $|H_{\text{Left}}| = |H_{\text{Right}}| = 40$, Left:Right is now 120:40 in the view of H_{Left} and 40:120 in the view of H_{Right} (up from 80:0 and 0:80, respectively).

This pattern repeats in subsequent slots, with the honest validators in H_{Left} and H_{Right} solely voting for the chains Left and Right, respectively, thus maintaining a balance of weights the in global view and perpetuating the adversarially induced split view—in the LMD view of each validator, they keep voting for the chain they see leading, and 'cannot understand' why other honest validators keep voting for the other chain.

4.4 Analysis

Recall that W and W_p respectively denote the number of voters per slot and the proposer boost. When adversary controls over β fraction of the committee in each slot, the LMD-specific balancing attack can be started after $W_{\text{p}}/\beta + 1$ adversarial slots, where the extra slot is used by the adversary to batch the equivocating votes into the left and right adversarial blocks. Release of this block can be timed to ensure that at least $(1-\beta)W/2 - \sqrt{W}$ of the honest attesters receive the left block first by analyzing the propagation of messages over the peer-to-peer network. In this case, the attack is sustained indefinitely as long as $(1-\beta)W - \sqrt{W} < W_{\text{p}}$.

The attack requires only a constant number of equivocating votes, enough to overcome the proposer boost. As a result, its cost is bounded, and payed one-time, despite the fact that it causes a permanent split among honest validators.

5 CONCLUSION AND FUTURE WORK

We have shown that the LMD functionality of the Gasper protocol can be exploited to conduct another balancing attack that is not mitigated by the patches (*e.g.*, proposer boosting [6]) proposed to prevent earlier balancing attacks (*e.g.*, [9, 14, 15, 17–22]). This LMD-specific balancing attack also works on HLMD GHOST proposed in [8], which is a slight variation of LMD GHOST used in combination with Casper FFG. This is because the attack does not need the votes on blocks that conflict with the latest *Casper FFG justified block* to cause a permanent split between two groups of honest validators. This is especially dire since such a permanent split would imply the existence of two branches with roughly equal votes, none of which gathers sufficiently many votes to have a new block justified by Casper FFG, thus stalling the progress of Casper FFG. We have also demonstrated that removing the LMD feature from Gasper's LMD GHOST component cannot be a solution to the above problems as the protocol without LMD suffers from the avalanche attack.

In response to the avalanche and LMD-based balancing attack, another patch, called *equivocation discounting*, was proposed as mitigation. Equivocation discounting requires honest validators to ignore votes within the LMD GHOST fork-choice rule from *equivocating* validators that have voted for multiple blocks during the same slot. Although equivocation discounting prevents the reuse of votes for uncle blocks by adversarial validators, not least because of its complexity, the security of Gasper with proposer boosting and equivocation discounting remains an open question.

A new protocol, Goldfish [11] was recently proposed as a provably secure drop-in replacement satisfying the design goals of LMD GHOST.

ACKNOWLEDGMENTS

We thank Danny Ryan, Aditya Asgaonkar, and Francesco D'Amato for feedback and fruitful discussions. JN, ENT and DT are supported by a gift from the Ethereum Foundation. JN is supported by the Protocol Labs PhD Fellowship and the Reed-Hodgson Stanford Graduate Fellowship. ENT is supported by the Stanford Center for Blockchain Research. The pick emoji by Twemoji/Twitter is licensed under CC-BY 4.0.

REFERENCES

[1] 2020. *Ethereum 2.0 Phase 0 – Beacon Chain Fork Choice.* https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/fork-choice.md
[2] 2021. *Ethereum 2.0 Phase 0 – Honest Validator.* https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/validator.md
[3] 2021. *Ethereum 2.0 Phase 0 – The Beacon Chain.* https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/beacon-chain.md
[4] Brunny.eth. [n. d.]. *The Market Cap of the Ethereum Ecosystem.* https://mirror.xyz/brunny.eth/-0Jn0dD5868h_WshCirxJPyjBD6hQvqPL18blZrEbsU
[5] Vitalik Buterin. 2014. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*
[6] Vitalik Buterin. 2020. *Proposal for mitigation against balancing attacks to LMD GHOST.* https://notes.ethereum.org/@vbuterin/lmd_ghost_mitigation
[7] Vitalik Buterin and Virgil Griffith. 2019. Casper the Friendly Finality Gadget. *arXiv:1710.09437 [cs.CR]* (2019). https://arxiv.org/abs/1710.09437
[8] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. 2020. Combining GHOST and Casper. *arXiv:2003.03052 [cs.CR]* (2020). https://arxiv.org/abs/2003.03052
[9] Vitalik Buterin and Alistair Stewart. 2018. *Beacon chain Casper mini-spec (comments #17, #19).* https://ethresear.ch/t/beacon-chain-casper-mini-spec/2760/17
[10] Pierre Civit, Seth Gilbert, and Vincent Gramoli. 2021. Polygraph: Accountable Byzantine Agreement. In *ICDCS*. IEEE, 403–413.
[11] Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. 2022. No More Attacks on Proof-of-Stake Ethereum? *arXiv:2209.03255 [cs.CR]* (2022). https://arxiv.org/abs/2209.03255
[12] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102.
[13] Aggelos Kiayias and Giorgos Panagiotakos. 2017. On Trees, Chains and Fast Transactions in the Blockchain. In *LATINCRYPT (Lecture Notes in Computer Science, Vol. 11368)*. Springer, 327–351.
[14] Ryuya Nakamura. 2019. *Analysis of bouncing attack on FFG.* https://ethresear.ch/t/analysis-of-bouncing-attack-on-ffg/6113
[15] Ryuya Nakamura. 2019. *Prevention of bouncing attack on FFG.* https://ethresear.ch/t/prevention-of-bouncing-attack-on-ffg/6114
[16] Christopher Natoli and Vincent Gramoli. 2017. The Balance Attack or Why Forkable Blockchains are Ill-Suited for Consortium. In *DSN*. IEEE Computer Society, 579–590.
[17] Joachim Neu, Ertem Nusret Tas, and David Tse. 2020. *A balancing attack on Gasper, the current candidate for Eth2's beacon chain.* https://ethresear.ch/t/a-balancing-attack-on-gasper-the-current-candidate-for-eth2s-beacon-chain/8079
[18] Joachim Neu, Ertem Nusret Tas, and David Tse. 2021. *Attacking Gasper without adversarial network delay.* https://ethresear.ch/t/attacking-gasper-without-adversarial-network-delay/10187
[19] Joachim Neu, Ertem Nusret Tas, and David Tse. 2021. Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma. In *IEEE Symposium on Security and Privacy*. IEEE, 446–465.
[20] Joachim Neu, Ertem Nusret Tas, and David Tse. 2022. The Availability-Accountability Dilemma and its Resolution via Accountability Gadgets, In International Conference on Financial Cryptography and Data Security. *arXiv:2105.06075 [cs.CR].* https://arxiv.org/abs/2105.06075 Forthcoming.
[21] Michael Neuder, Daniel J. Moroz, Rithvik Rao, and David C. Parkes. 2021. Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders. *Workshop for Game Theory in Blockchain (GTiB) at the 2020 Conference on Web and Internet Economics (WINE)* (2021). https://arxiv.org/abs/2102.02247

[22] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. 2022. Three Attacks on Proof-of-Stake Ethereum, In International Conference on Financial Cryptography and Data Security. *arXiv:2110.10086 [cs.CR]*. https://arxiv.org/abs/2110.10086 Forthcoming.

[23] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure High-Rate Transaction Processing in Bitcoin. In *Financial Cryptography (Lecture Notes in Computer Science, Vol. 8975)*. Springer, 507–527.