Article

Decentralization Is Good or Not? Defending Consensus in Ethereum 2.0

Vojislav B. Mišić [1,*], Soosan Naderi Mighan [1], Jelena Mišić [1] and Xiaolin Chang [2]

1 Department of Computer Science, Toronto Metropolitan University, Toronto, ON M5B 2K3, Canada; soosan.naderi@torontomu.ca (S.N.M.); jmisic@torontomu.ca (J.M.)
2 Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100091, China; xlchang@bjtu.edu.cn
* Correspondence: vmisic@torontomu.ca

Abstract: Proof-of-Stake (PoS) protocols are widely accepted as a viable substitute for the Proof-of-Work-based consensus, which is why recent blockchain-based cryptocurrencies and applications, most notably Ethereum 2.0, are using some variant of PoS as the basis for the consensus protocol. However, the implementation of PoS protocols in Ethereum 2.0 are not without its share of problems and vulnerabilities, especially with respect to the malicious behavior of validator nodes. In this paper, we first review the basic tenets of PoS protocols. We then discuss some of the recently described attacks on the Ethereum 2.0 consensus, and we also show that some of the design rationales adopted in PoS implementation—the decentralization of the voting process in particular—have, in actuality, enabled attacks that can be launched at a very low cost to the attacker. We also propose simple remedies that can reduce or eliminate the impact of those attacks and can evaluate the performance of the Ethereum 2.0 consensus when these remedies are applied.

Keywords: Ethereum 2.0; PoS; attacks on consensus

check for updates

Citation: Mišić, V.B.; Naderi Mighan, S.; Mišić, J.; Chang, X. Decentralization Is Good or Not? Defending Consensus in Ethereum 2.0. *Blockchains* **2024**, 2, 1–19. https://doi.org/10.3390/blockchains2010001

Academic Editors: Keke Gai and Liehuang Zhu

Received: 8 November 2023
Revised: 11 January 2024
Accepted: 19 January 2024
Published: 23 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Ethereum [1,2] is one of the most popular cryptocurrencies with an over USD 190 billion market capitalization (https://coinmarketcap.com/currencies/ethereum/, last accessed 20 September 2023), which is second only to Bitcoin [3]. Ethereum initially used a Proof-of-Work-based consensus similar to Bitcoin, but it switched to Proof-of-Stake-based one in September 2022. The Ethereum 2.0 consensus uses two previously defined components: the finalization gadget known as Casper The Friendly Finality Gadget (Casper FFG) [4] and a Latest Message Driven variant of the Greedy Heaviest Observed Subtree (LMD-GHOST) principle [5] as the fork choice rule.

The switch, commonly referred to as The Merge, allowed the PoS-based Ethereum 2.0 to achieve a several orders of magnitude reduction in power consumption, as well as a noticeable reduction in mining and transaction fees. It also gave rise to some new problems and vulnerabilities. In particular, the emphasis on decentralization of the voting process has resulted in an increased sensitivity in the new PoS-based consensus to the malicious behavior of validator nodes, even when the number of such nodes is well below the established limits for successful operation. A number of patches have been introduced since, but they have not succeeded in resolving all systemic problems.

In this paper, we review the basic concepts of the consensus in distributed systems in Section 2. We then discuss the PoS consensus in Ethereum 2.0 and the decisions that guided its design in Section 3, where we highlight the differences between Ethereum 2.0 and the original Ethereum. In Section 4, we summarize the attack surface of Ethereum 2.0 by identifying the features of the consensus protocol that make attacks possible, or at least facilitate them, and then proceed to describe some of the recently described attacks that specifically target consensus. In Section 5, we propose simple countermeasures that would reduce the attack surface of the Ethereum 2.0 consensus, and we then evaluate their impact on the performance of Ethereum 2.0 systems in Section 6. Finally, Section 7 concludes the paper.

2. Preliminaries: Consensus and How to Achieve It

Consensus refers to a protocol that attempts to enable a group of nodes in a distributed system to agree on an action or a value [6]. It is essentially an extension of the problem of maintaining the synchronization of data on a number of nodes in a distributed database. It is a crucial component of blockchain systems, where the distributed blockchain ledger is replicated by a number of nodes that may be scattered all over the world.

The Practical Byzantine Fault Tolerance (PBFT) protocol of Castro and Liskov [7] was the first efficient protocol to offer Byzantine Fault Tolerance, i.e., resilience to a certain proportion of nodes behaving in a malicious manner. Many other similar ones, mostly modifications of PBFT, followed [8]. PBFT achieves the acceptance of a proposal (e.g., a block) in three stages or phases: PREPREPARE, PREPARE, and COMMIT. In the first phase, the leader broadcasts the proposal to all other nodes in a PREPREPARE message. In the second phase, nodes that have successfully validated the proposal broadcast their readiness to accept the block through PREPARE messages. Nodes that have received a sufficient number of PREPARE messages broadcast their acceptance through COMMIT messages. Once a node receives a sufficient number of COMMIT messages, it marks the proposal as finally and irrevocably accepted.

In both PREPARE and COMMIT stages, a majority of more than two-thirds of all nodes is needed. In other words, a system with $n \geq 3f + 1$ nodes is capable of operating properly in the presence of f faulty or malicious nodes. This is the case with most other Byzantine Fault Tolerant protocols.

Proof-of-Work, Bitcoin, and Forks

Bitcoin: The world's first cryptocurrency, Bitcoin, uses the so-called Nakamoto consensus [3], in which nodes in a permissionless peer-to-peer network create (or 'mine') new blocks by solving a non-trivial cryptographic puzzle. The node that solves the puzzle sends it through the network, and other nodes validate it and append it to their respective blockchains. The block is accepted when all the nodes in the network append the block to the top of their local blockchains.

The difficulty of the cryptographic puzzle makes mining an expensive and high-energy-consumption operation, which is why the Nakamoto consensus is often referred to as a Proof-of-Work, or a PoW-based, consensus. It also leads to a low throughput, with one block being mined only every ten minutes on average. However, the communication cost is low since the propagation of a block throughout the network requires n messages, and accepting the block does not incur any additional communication cost. Actual protocols use notification and request messages, but the total number of messages is still a linear function of the number of nodes.

Forks: Due to propagation delays in the network [9], one or more competing blocks may be mined and sent through the network before a prior block has finished its propagation through the network. In such cases, the blockchain 'forks'—i.e., two or more competing tips—appear, as shown in Figure 1. Some nodes in the network install one or the other block at the tip of their respective blockchains; this partitions the network [10] and renders the blockchain inconsistent, which is certainly undesirable.

Forked subchains can extend to several blocks. They are eventually resolved when one of the branches is adopted by the entire network. According to the Greedy Heaviest Observed Subtree (GHOST) principle [5], this should be the branch with the largest amount of invested work. In practice, it is usually the longest branch when counting from the genesis block, as shown in Figure 2.

The probability of forking in Bitcoin was never high due to the high cost of mining, and the length of the forked subchains rarely reached four or five, even in the early days of Bitcoin [9].

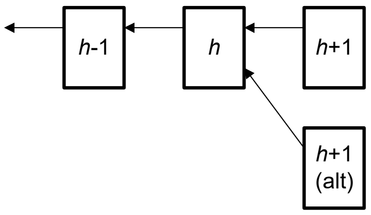


Figure 1. In Bitcoin, a fork is created when blocks are mined in alternate branches.

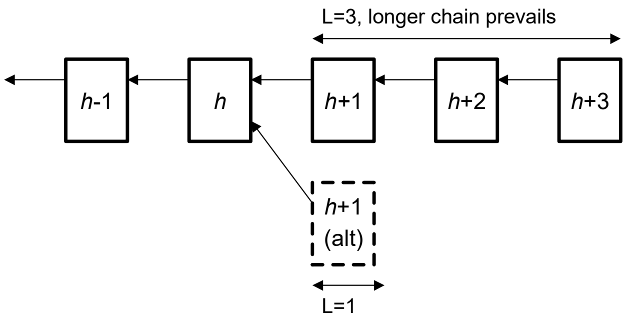


Figure 2. In Bitcoin, forks are resolved by choosing the branch with the most blocks as the canonical one.

The PoW-based consensus offers only probabilistic finality, as the blockchain may be reorganized if a miner with sufficient hashpower mines an alternative chain and releases it to create a fork and invalidate the corresponding portion of the main chain. This is often referred to as the 51% attack [11] or, somewhat unfortunately, as 'selfish mining' (as if regular Bitcoin mining is altruistic). A 51% attack has never occurred in Bitcoin due to the high cost of mining, but equivalent attacks have been observed in other PoW cryptocurrencies such as Bitcoin Cash.

Original Ethereum: The second major cryptocurrency, Ethereum [1,2], was initially designed to also use a Proof-of-Work-based consensus. Unlike Bitcoin, the block interval was set to a much shorter value of about 13 s per block on average, which means that forking would be a much more frequent event. To discourage forking and reduce the possibility of malicious miners launching 51% attacks, Ethereum allows newly mined blocks to refer to uncle blocks, i.e., forked blocks that are not included in the canonical chain. In such cases, the miner of the uncle block receives a miner's fee, which is reduced with respect to the regular fee and proportional to the depth of the uncle block (up to the predefined maximum depth of seven with respect to the current blockchain tip). In this setup, the problems of excessive power consumption due to PoW and the probability of forking still remain, albeit at a reduced rate.

Other minor differences exist, but an Ethereum that uses the PoW consensus generally behaves in a very similar fashion to Bitcoin.

We also note that Ethereum was the first cryptocurrency to introduce smart contracts—self-contained pieces of code that can be stored on the blockchain and executed as needed. This, in turn, has necessitated a certain mechanism to gauge and limit resource consumption in the process of executing the smart contracts contained within the block, or the invocation of smart contracts that already exist on the blockchain. This mechanism is known as *gas*, and each transaction has to specify the maximum amount of gas it can consume. Smart contracts can be analyzed in many ways, often through the use of formal methods [12,13].

3. Proof-of-Stake and Ethereum 2.0

An alternative approach to consensus, based on the Proof-of-Stake (PoS) protocols, was proposed for use in cryptocurrencies as early as 2012 [14,15]. In a PoS consensus, a known group (i.e., a committee) of participants, often referred to as validators, have to deposit or freeze their own funds—typically, a certain amount of system cryptocurrency, which is referred to as the stake—in order to be able to vote for (or against) accepting a block.

The main purported advantage of PoS over PoW is the reduction in energy consumption (as there is no need to solve any puzzle) and a somewhat faster block processing due to the fact that the number of validators is typically much smaller than the number of nodes. This makes PoS systems suitable in a wide range of applications, including in the Internet of Things and other similar systems [16].

The membership of the committee is periodically revised to reduce the risk of malicious (and perhaps colluding) validators controlling the vote. The validators' stake may be revised during the time period they are active to reflect the truthfulness of their votes. Also, the current amounts of stake may be used as the weight for the validators' vote.

To limit the communication load, the number of validators has to be limited; but it still has to be high enough to reduce the likelihood of collusion between validators, as this might threaten the truthfulness of the voting.

In theory, any node in a peer-to-peer network could act as a validator if it possesses a sufficient stake. If the minimum stake is too low, we may end up with too many validators, which would make voting unwieldy and thus defeat the purpose of using a PoS consensus in the first place. If it is too high, the committee would be small and, consequently, efficient—but then it would also risk stalling if enough validators lose their required minimum stake.

Another solution is to limit the number of validators on the committee and allow all validators with a sufficient stake to actually vote for the membership of the validating committee first, where the elected validators will then vote for individual blocks. This is known as a Delegated Proof-of-Stake [17] system.

3.1. Proof-of-Stake in Ethereum 2.0

A solution that attempts to find the common ground between the two extremes has been adopted in Ethereum 2.0. As noted above, Ethereum switched to a PoS consensus in September 2022 after about two years of operation where the PoW consensus was augmented with a beacon block chain, as well as other elements that were implemented, to facilitate the transition to a PoS system.

Ethereum 2.0 divides time into epochs and slots. In each slot, which lasts 12 s as shown in Figure 3, a single block should be proposed to the network and voted for by the validator committee(s). An epoch consists of 32 slots, as shown in Figure 4, and all the blocks in an epoch should be finalized, i.e., irreversibly accepted, soon afterward.

In each slot, a committee of 128 validators is selected to attest for the block (or blocks) added in the current or the previous epoch. Any node can become a validator, provided it deposits the amount of 32 Ethers (ETH), which is held (and adjusted, as discussed below) until the node decides to withdraw or is ejected due to transgressions. Upon successfully invoking the appropriate contract, the prospective validator is entered into the waiting queue where it spends some time, around 17 h on average [18], before it is added to the validator pool.

The stake of a validator cannot exceed 32 ETH and any excess is periodically transferred back to the validator. The stake of a validator cannot drop below 16 ETH, otherwise it is removed from the validator pool. A validator can also refill its stake up to 32 ETH.

In each slot, one of the validators from the committee is randomly assigned the role of a block proposer, i.e., the node authorized to build and propose a new block in that slot. To avoid a Denial-of-Service attack, as well as other attacks, the identity of the proposer is known only two epochs in advance. The proposer should choose the parent of the new block on the basis of the LMD-GHOST [19] fork choice rule discussed below; however, other criteria may also be used.

In each epoch, validators for each slot are selected by a random permutation. Ideally, each validator will be able to vote exactly once in each epoch. As the total number of validators is much higher than is needed—as the current count is of the order of several hundred thousand—a two layer-structure is adopted in practice, in which a number of validator committees are created for each slot. Those committees are grouped into subnets of the Ethereum peer-to-peer network, and they vote in parallel within those subnets. Votes are signed by the validators, who aim to eliminate Sybil attacks.

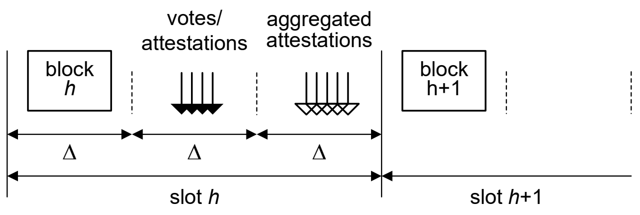


Figure 3. In Ethereum 2.0, a block is proposed in each slot and votes (attestations) are sent and aggregated.

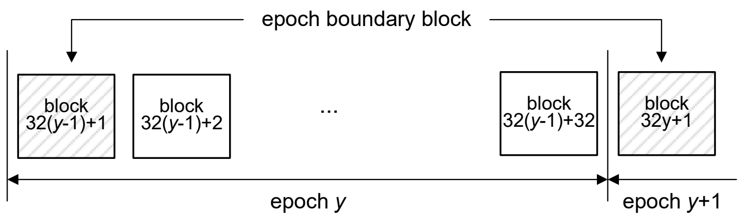


Figure 4. In Ethereum 2.0, an epoch lasts for 32 slots and the first block in an epoch is often referred to as the epoch boundary block.

The presence of separate subnets allows, or will eventually allow, the blockchain to be partitioned into a number of subsets or shards that could possibly overlap, in which case different validator committees would vote for their respective shards. However, proper sharding has not yet been implemented.

The validators' votes are then aggregated and propagated through the network by the so-called aggregating validators, of which there is at least one in each subnet. It is expected that most validators in a given subcommittee will vote the same way so that aggregated attestations will not be excessively large.

The objective of having that many validators is to *decentralize* the decision making by allowing all validators to participate. In this manner, if f out of n validators behave maliciously, their impact on the decision process will be proportional to f/n. (In actuality, it can be a higher since it is the validators' stake that, rather than a simple count, determines the outcome, but since the stake varies in the range from 16 to 32 ETH only, the difference is small.)

As a result, if all validators are involved, an honest majority should exist in most slots/epochs, thus leading to a smooth operation of the Ethereum 2.0 blockchain. Conversely, selecting only a subset of possible validators for a committee would increase the likelihood that malicious validators can exercise an undue influence on the decision reached by the committee.

A validator committee with a fixed size does not directly reduce the communication load as all validators should vote once per epoch. Some reduction is achieved through aggregation since the original attestations are restricted to the respective subnets and only the aggregated attestations are broadcast to the entire peer-to-peer network. However, aggregation also adds additional delay time to attestation propagation.

An additional benefit of spreading the vote over the entire epoch is that it gives sufficient time to check and aggregate the signatures within attestations. Ethereum 2.0

uses group signatures that follow the Boneh–Lynn–Shacham (BLS) protocol [20]; although efficient in terms of length and processing time, the sheer number of such signatures that need to be checked and aggregated in the attestation process presents a challenge in a permissionless system such as Ethereum.

There is yet another committee called the sync committee, which signs finalized blocks so that light clients (i.e., nodes that do not store the complete blockchain) can keep track of the blockchain without having to download each and every block and/or transactions. However, the sync committee has no role in the consensus process; as such, we will not discuss it here.

Table 1 highlights the major differences between Ethereum 2.0 and the original Ethereum.

Table 1. Ethereum 2.0 vs. original Ethereum.

	Original Ethereum	**Ethereum 2.0**
Introduced in	2013–2014	September 2022
Consensus protocol	Proof-of-Work	Proof-of-Stake
Time division	None	Slots (12 s each) and epochs (32 slots)
Inter-block arrival time	≈13 s	12 s (one per slot)
Block proposer	Random—first node to mine a block	Pseudo-random—known in advance
Block acceptance	By the implicit majority of all nodes	Through a majority of attestations cast by validator committee
Validator role	Implicit—all nodes participate	Requires 32 ETH deposits prior to being admitted
Chain growth	By the random choice of a parent block	Achieved using the LMD-GHOST rule (see text for details)
Forks	Possible due to propagation delays	Possible due to the deliberate actions of the proposer
Block finality	Probabilistic (i.e., can be overturned by sufficient majority)	Explicit—through the Casper FFG protocol (see text for details)
Beacon (checkpoint) blocks	Beacon chain introduced in December 2020 for future compatibility with Ethereum 2.0	Used to support block finality since The Merge (September 2022)

3.2. Voting: On Fork Choice Rule and Justification

Attestations can, in actuality, contain two votes: one that identifies the block that the validator considers to be the current tip or head of the blockchain, and one that identifies the link between checkpoint blocks. The former is used in the process of fork choice, i.e., the selection of one of the alternative subchains to be adopted as the canonical chain and subsequently extended with a new block; the latter is used in the process of blockchain justification and finalization. Both processes are executed by each validator, where packaging the votes together aims to reduce the communication load in the network (as does the aggregation of votes from a subnet). We note that attestations can be added to a block proposal or sent separately.

The vote for the head of a blockchain is weighted by the validator stake. As the allowable range of stake values is from 16 to 32 ETH, there is no risk that a single validator would have a big enough stake to take over the vote. However, adversarial validators can still collude if they decide to subvert the voting process.

The fork choice rule in Ethereum 2.0 follows the LMD-GHOST principle [19], in which forks are resolved by choosing the sub-branch with the heaviest weight, i.e., those with the highest number of attestations (counted from the last justified block in the chain). This scenario is shown in Figure 5.

Only the latest attestation from any given validator is taken into account, and that vote remains valid until a new one comes along (whenever that happens) [18].

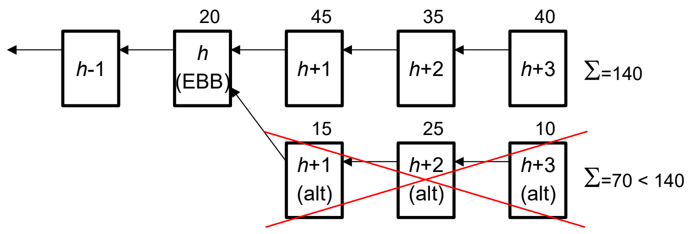


Figure 5. In Ethereum 2.0, forks are resolved using the LMD-GHOST rule, which is achieved by counting from the last justified block in the chain. In this manner, the lower branch with blocks labeled with (alt) is abandoned in favor of the upper branch which has the higher weight.

Finality is one of the most desirable properties of the blockchain: it means that the chain, at the current time, is considered immutable. In Bitcoin, finality is only achieved in a probabilistic sense, as an attacker with sufficient hashpower may create an alternative canonical chain and release it at an opportune moment with the objective of completely eliminating the current valid chain [5]. On the other hand, PBFT finalizes each proposal in sequence at the expense of inefficiency due to its three stages or phases.

To achieve finality in an efficient way, Ethereum 2.0 introduced the concept of checkpoint blocks: selected blocks in the blockchain that, when finalized, delineate the part of the chain that precedes them as final. This does not eliminate the possibility of a long-range reorg attack, but it limits the scope of such an attack to the non-finalized portion of the chain.

Finality is achieved in two steps: justification and finalization. It is achieved using the so-called Casper FFG [4], which uses the votes to first 'justify' and then 'finalize' selected blocks. A supermajority link from a previously justified checkpoint block makes the target checkpoint block justified, and a supermajority link from a previously justified checkpoint block to a direct child checkpoint block finalizes the former block and justifies the latter. These scenarios are schematically shown in Figures 6 and 7, respectively.

Supermajority here denotes a majority of more than two-thirds of the available stake, just like in PBFT [7]. Justification and finalization are thus broadly analogous to the PBFT stages of PREPARE and COMMIT, respectively. However, PBFT follows those steps sequentially for each proposal, whereas Casper FFG works on an expanded time scale so that only one block per epoch is explicitly justified or finalized; having said this, the preceding blocks in the epoch are also then finalized.

Once a block is finalized, it cannot be changed unless a full third of the validators vote for the change, which is highly unlikely as they would lose their stake in the process [18].

The genesis block is always justified and finalized, and there is one checkpoint block per epoch; moreover, it is usually, but not always, the first block in an epoch, and it is referred to as an epoch boundary block.

It is worth noting that Casper FFG requires stronger assumptions regarding node synchrony than the partial synchrony model [4,21] required by PBFT, although the difference is probably more important in theory than in practice.

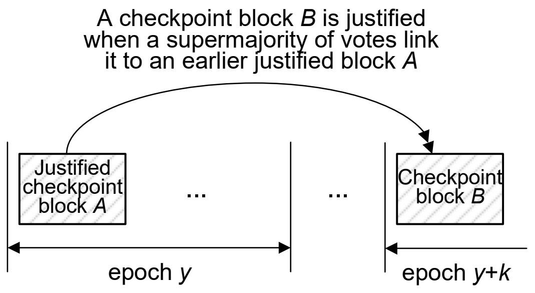

A checkpoint block *B* is justified
when a supermajority of votes link
it to an earlier justified block *A*
Justified
checkpoint
block *A*
...
...
Checkpoint
block *B*
epoch *y*
epoch *y*+*k*

Figure 6. The justification of a block requires a supermajority vote from an earlier block that has already been justified.

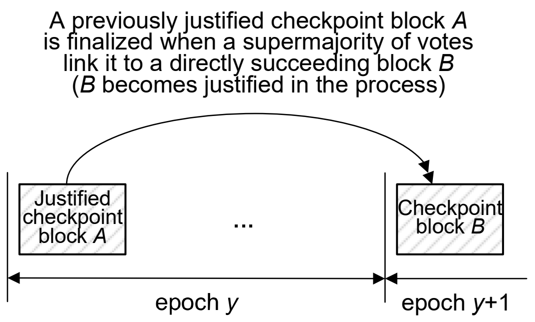

A previously justified checkpoint block *A* is finalized when a supermajority of votes link it to a directly succeeding block *B* (*B* becomes justified in the process)
Justified checkpoint block *A*
...
Checkpoint block *B*
epoch *y*
epoch *y*+1

Figure 7. The finalization of a previously justified block requires a supermajority vote to a direct child checkpoint block, which becomes justified in the process.

Validators from any given slot are free to send attestations for a block proposed in that same slot, but they can defer them for up to 32 slots, i.e., an entire epoch. For obvious reasons, justification/finalization votes must apply to the checkpoint blocks from previous epochs or (at best) to the checkpoint block at the beginning of the current epoch.

In this manner, the Ethereum 2.0 design intertwines two protocols [22] that address two desirable features, albeit on two different time scales. The Casper FFG protocol aims to achieve finality for an epoch, so that clients can be certain that all the blocks therein are irreversibly finalized. The LMD-GHOST protocol addresses the availability of each block, so that clients can access them at all times, even when not all of the chain is finalized. The two protocols are jointly referred to as Gasper [19].

3.3. Rewards, Penalties, and Slashing

Validators that send correct attestations are rewarded, while those that send incorrect ones or abstain from sending attestations are penalized. More severe transgressions are punished by slashing, i.e., removing a predefined amount from its stake and by removing it from the validator pool into the exit queue. After a predefined wait period, another part of the stake will be taken away, the actual amount of which depends on the number of simultaneous transgressions that occurred within a certain time period before and after the actual perpetrator's transgression. After another wait period, the perpetrator will ultimately be ejected from the exit queue.

Slashable offences are, in fact, equivocations—making two statements that contradict each other. These include the following scenarios:

- A proposer proposes two different blocks in the same slot ('at the same height' with respect to the genesis block at the beginning of the blockchain).
- A validator attests to different head blocks whilst voting for the same pair of checkpoints.
- A validator attests twice for the same target checkpoint but with different sources.
- Finally, a validator makes an attestation where the pair of checkpoints surround, or are surrounded, by another pair of checkpoints that the same validator has already attested for. (Note that the latest message rule does not apply in this case.)

The slashing procedure and the accompanying sanctions sound worse than they actually are. Ethereum 2.0 documentation claims that PoS protocols are a vehicle to prevent Sybil attacks [18]; however, an adversary with sufficient financial means can easily create a number of identities and deposit the required stake for each of them. Losing the 32 ETH stake deposited by one of those identities due to a slashable offence is still not a very high price to pay as long as there is a payout, financial or otherwise, to be gained in this manner.

Slashing does not happen automatically: someone (i.e., another validator) needs to take note of the transgression and report it to the network, together with the proof that it has actually taken place. Sanctions are applied only when the network accepts the report. Measurements spanning five-and-a-half months indicate that the number of slashings is minimal (refer to [23]), with the majority of violations being identified within ten slots or even less. However, blocks and attestations submitted by the perpetrator are not retracted.

4. Attacks on the Ethereum 2.0 Consensus

4.1. Consensus Attack Surface

The Ethereum 2.0 consensus protocol described above suffers from a number of vulnerabilities, most notably the following:
The long time needed to accept block proposals: The block proposer should propose the block in the first part of the designated slot, but block proposals that arrive throughout that slot or the next one are also accepted as valid. This has been done deliberately so as to accommodate unpredictable network delays that can occur in an actual block proposal. Delays can also occur in the process of obtaining the block via a block building service such as MEV-Boost by Flashbots (https://www.flashbots.net, last accessed on 1 September 2023).
The long time required to accept attestations: Similar to block proposals, attestations are valid if received up to 32 slots later than the slot in which the block in question was proposed. In Deneb, attestations will be valid in the next update of the protocol for the entire current and next epoch (https://github.com/ethereum/consensus-specs/pull/3360, last accessed on 30 September 2023). Long time windows aim to allow sufficient time to (1) collect the votes in the presence of network delays, and to (2) validate and aggregate the BLS signatures on those votes before sending out the aggregated attestations. (Note that both blocks and attestations are distributed throughout the network using a gossip-like protocol.) The ultimate goal is to decentralize the decision process by allowing all validators to vote and, consequently, minimize the impact of malicious validators.

Unfortunately, such a long time also allow enough time for malicious validators to use the knowledge of honest validators' messages to their benefit, which is achieved by a subtle manipulation of the fork choice rule or by trying to sway the checkpoint vote to prevent the chain from justifying or finalizing blocks in an epoch. (Note that, in PoW systems such as Bitcoin, a node has no way to influence the decision of other nodes except to mine a block itself, which is expensive and time consuming.)
Attacks are cheap to launch in a PoS system: Attacks can also be launched through untruthful voting actions, which come with a light or no penalty at all (nothing-at-stake), and only a few require the perpetrator(s) to actually commit a slashable offence. (In comparison, similar attacks in Bitcoin require a great deal of time and computational power, which is probably the reason why those type of attacks do not occur very often.)

We stress that all of the vulnerabilities described above stem from the need to decentralize the voting process and include all validators. At the same time, they allow smart adversaries to focus their efforts in order to sabotage the process of block proposing and attestation, thus ultimately affecting the consensus.

The mechanisms used to launch attacks directly follow from the vulnerabilities listed above. They can be summarized as follows.

1. Equivocation when proposing a block: The designated block proposer may propose different blocks to different subsets of validators, as shown in Figure 8. The goal is to create a fork and confuse the honest validators when they apply the fork choice rule for as long as possible and, consequently, slow down the justification/finalization process. This is the only mechanism that constitutes a slashable offence.
2. Delayed release of a block: An adversarial proposer may deliberately delay the block, only to propose it at an opportune moment, as shown in Figure 9. Namely, the block proposer is allowed to defer the block proposal in order to force validators to send attestations for the parent block, which then contribute to the weights of all of its child branches, legal or not. This can create a fork in a completely legal manner and used as the starting point of an attack. Depending on the actual position of the block, the goal of the attacker may be a reorganization of the chain or a delay in finalizing the chain.
3. Delayed and/or opportunistic release of attestations: One or more of the adversarial validators may deliberately delay their attestations, only to release them at an opportune moment, as shown in Figure 10. This requires the monitoring of honest validators' attestations in order to determine the best time in which to release a privately mined block (or blocks), as well as for privately collecting the malicious validators'

attestations. It is made possible by the long time period in which the attestations are accepted.

To their credit, the developers of Ethereum 2.0 have tried to modify the system in order to make it resilient to some of these attacks; hence, a number of patches were added since the launch of the new system. However, not all attacks can be prevented in this manner.

Let us now review a number of attacks on the consensus protocols. We can assume that cryptographic protection is not broken, and we can disregard the software flaws exploited in incidents such as the MEV boost relay incident (https://collective.flashbots.net/t/post-mortem-april-3rd-2023-mev-boost-relay-incident-and-related-timing-issue/1540, last accessed on 25 September 2023), which could also be categorized as attacks.

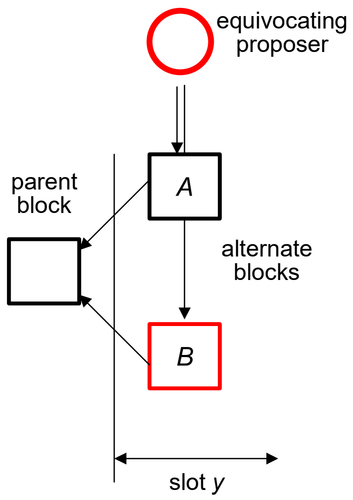


Figure 8. Equivocation when proposing a block.

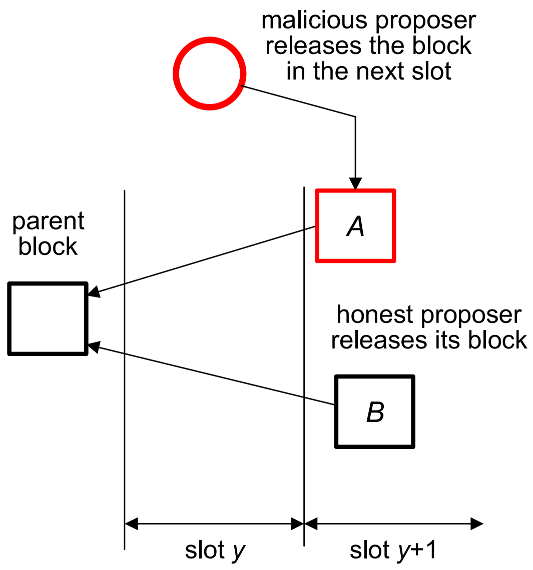

malicious proposer releases the block in the next slot
parent block
A
honest proposer releases its block
B
slot y
slot y+1

Figure 9. Delayed release of a block.

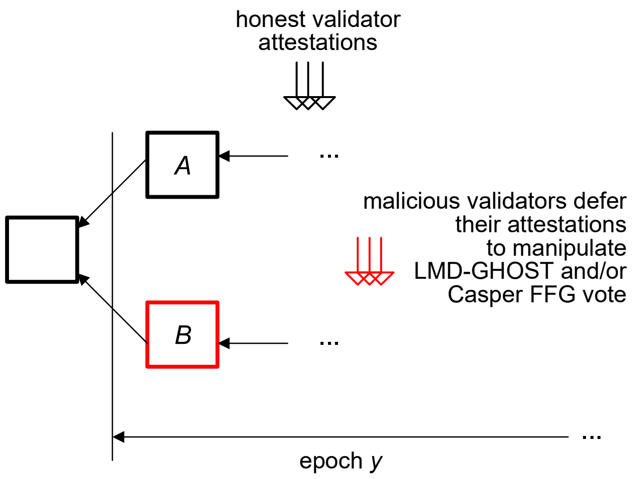

honest validator attestations
A
...
malicious validators defer their attestations to manipulate LMD-GHOST and/or Casper FFG vote
B
...
...
epoch *y*

Figure 10. Delayed and/or opportunistic release of attestations.

4.2. Replacement/Reorg Attacks

Replacement or reorg (short for reorganization) attacks use forking (which may occur due to network delays) to reorganize the blockchain by replacing a part, or all, of the blockchain created by honest block proposers and validators. Their impact is particularly damaging in cases of network failure that, either accidental or deliberate, results in the partitioning of the validator network. Most of the attacks of this type aim for short-term reorgs that replace a single accepted block to facilitate double spending [24].

In a *saving attack* [21], the attacker withholds its block only to propose it at an opportune moment in order to draw the votes of the honest validators and create a conflict that delays block finalization. The *malicious reorg attack* [25] is similar to the saving attack, but it also bears superficial similarity to the 51% attack in PoW blockchains [5]. In this attack, the adversary deliberately creates a private fork (i.e., a new block) and follows with private attestations for it, but it does not release either of them. As a result, honest validators will attest for the parent of the private block. When an honest miner proposes a valid block in the next slot, the attacker releases its block and the accompanying attestations. As the vote in that slot will be divided between the new honest block and the attacker's block, which is supported by the attacker's attestations from the previous slot, the attacker's block can easily be selected by the LMD-GHOST fork-choice rule to effectively displace the otherwise valid block as the canonical head of the chain.

The malicious reorg attack was elaborated on and refined in [26], where the constraints on releasing the adversarial votes were relaxed. It was also claimed that an adversary that controls the block proposer and only one other validator can succeed in achieving a one-block reorg through this attack.

The avalanche attack targets the GHOST protocol without the LMD part, but it is claimed that it can also damage the LMD-GHOST [27]. This attack combines private mining, which is similar to the long-range attack, with a deliberate equivocation that leads to forks. Its goal is to replace a number of valid blocks in the blockchain with privately mined blocks using an 'avalanche of equivocating sub-trees' to displace the honest chain. In this case, the attacker mines a number of blocks, which are released in an opportune moment as a forked sub-tree. This tree is flat but wide, which takes advantage of the LMD-GHOST fork choice rule that allows such a tree to displace the current canonical chain. Furthermore, the attacker can repeatedly inject previously mined blocks (as virtual 'uncle' blocks, even though uncle blocks available in PoW Ethereum are not possible in Ethereum 2.0) in order to confuse honest validators. It is claimed that this procedure is technically not an equivocation, although effectively it is one, and thus it is formally allowed by the LMD-GHOST protocol [27]; however, we have not found a confirmation of this claim elsewhere in the literature.

We note that network attacks, such as those described in [28] in the context of PoW Ethereum, can be also launched in the Ethereum 2.0 environment [29], where the ultimate effect of slowing down block propagation and degrading the performance of the system without any consequences for the attacker is sought after.

4.3. Attacks on the Finalization of Checkpoint Blocks

A number of attacks target the finalization of checkpoint blocks.

The *balancing attack* [22] makes use of the LMD-GHOST part of the Gasper protocol to stall, or at least delay, the finalization of the checkpoint blocks. In this attack, the adversary needs to be the proposer in the first slot of the epoch, and there needs to be at least six validators in each slot. (Note that this number is much smaller than one-third of the committee needed to damage a PBFT vote.) The adversarial proposer creates a fork by proposing two blocks that are propagated to equal-sized subsets of the validator committee. The adversarial validators monitor the development of the attestations used in the fork choice rule and opportunistically withhold or release their votes so as to keep the balance between the two blocks/branches. In this manner, the competing blocks are prevented from receiving a sufficient number of votes to ensure finalization. The 'split' chain that ensues can persist for some time without either of the branches being able to reach finalization. Note that the attacker need not have control over message propagation delays in the network—it suffices to have random delays between nodes, as is the case in most peer-to-peer networks anyway.

To counter the balancing attack, a patch known as the 'proposer boost' was introduced, which adds additional weight to the attestations of the block proposer in the slot where the block is proposed. However, even this patch can be taken advantage of by an attacker that manages to propose two competing chains, and it can subsequently maintain the approximate balance between them by a judicious release of appropriate attestations (using equivocating if necessary [27]).

The balancing attack inspired the *refined liveness attack* [26], which can stall the process of finalization in the presence of a probabilistic network delay. As before, the attacker waits until the validators it controls are selected as block proposers in two subsequent slots. Then, the proposers create a fork and use carefully timed attestations to maintain the tie between the competing branches. It has been claimed that only a small number of attacker-controlled validators are needed to accomplish this scenario, as the necessary timing parameters can be easily calculated on the basis of the data available to the attacker [26].

A combination of the refined malicious reorg attack and the refined liveness attack has been shown as able to achieve a long-range reorg attack in the presence of a probabilistic network delay with only a small portion of the total validator stake [26].

The *finality delay attack* [25] is similar to the balancing attack as it attempts to delay the finalization of a legitimate checkpoint block, but it is somewhat harder to launch. More specifically, the attacker must control the proposer of an epoch boundary block and the proposer of the subsequent block in the chain. The probability that a single validator will be given this opportunity is virtually zero due to the manner in which the random selection of the block proposer is made, but two or more colluding validators may well be in a position to launch the attack.

In this case, the attackers may launch the finality delay attack by creating a private fork in which the attacker withholds the epoch boundary block (i.e., the first block in an epoch) and, possibly, several subsequent blocks. The attacker then waits until the number of attestations from honest validators becomes sufficient so as to cause the last block in the previous epoch to be recognized as the checkpoint block. Once that happens, the attacker releases the withheld blocks, which will overwhelm the Casper FFG votes for the last block of the previous epoch. As the first withheld block cannot gain a sufficient number of attestations either, there can be no supermajority link between the last finalized checkpoint block and either of the competing checkpoint blocks. As a result, the finalization is delayed for the time period that depends on the number of adversarial validators.

Simulations show that an attacker with a 30% stake is able to execute this attack, even though the nearly official Ethereum documentation claims that this portion of the total stake is not sufficient for affecting the finality of the chain [18].

Note that this attack can be launched without any penalty to the attacker, as withholding a block is not an offence.

A similar mechanism is used in the *staircase attack*, which aims to starve honest nodes of their stake, even though they may be following the protocol specifications to the letter [30]. The attack exploits the possibility that adversarial blocks and attestations can be delayed and released at an opportune moment to force the LMD-GHOST rule two switch to the adversary-proposed block and invalidate the attestations from the home validators. In this manner, the adversary can achieve both the chain reorganization and replacement of an already finalized checkpoint block. In addition, the adversary that controls about 29% of the stake may launch the attack repeatedly so that all honest validators lose their stake whereas the malicious validators experience increased stake. Once the adversary controls a sufficient portion of the total stake, system security will be compromised [18].

The *bouncing attack* (https://ethresear.ch/t/analysis-of-bouncing-attack-on-ffg/6113, last accessed on 29 September 2023) tries to divide the honest validators' votes between two candidate subchains so that neither of them can get finalized. In this case, one of the subchains contains a justified checkpoint block, whereas a checkpoint block in the later epoch is justifiable, i.e., if it could be justified that all adversarial nodes cast their votes for it. Once the honest validators begin switching over to the branch containing the justifiable checkpoint, the adversarial nodes can again revert their votes. In this manner, the justifiable chain bounces from one branch to another, thus preventing finalization of the checkpoint blocks in either branch.

A patch to prevent this attack (https://ethresear.ch/t/prevention-of-bouncing-attack-on-ffg/6114, last accessed on 28 September 2023) was proposed to impose a limit on the validity of justification votes so that all votes that are delayed for more than a predefined number of slots are considered missing and, consequently, ignored. Nonetheless, a *probabilistic bouncing attack* [31] is still possible if a malicious block proposer manages to propose, just before the end of the validity period, a block on an alternative branch with a sufficient number of adversarial justification votes that have not been released before.

Fortunately, justification and finalization cannot be deferred indefinitely. Specifically, if no checkpoint block has been finalized in four consecutive epochs, Ethereum 2.0 activates a special operational mode known as the *inactivity leak* [18]. In this mode, attesters do not receive any rewards, while validators that fail to participate in the attestation process are penalized by increasingly larger amounts. The goal is to motivate the validators to try and restore finality, as well as return to normal operation as soon as possible, as any delay in justification and finalization affects the availability of the Ethereum 2.0 blockchain.

Yet even this feature cannot prevent scenarios such as the *double finality attack* (https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/attack-and-defense/, last accessed on 28 September 2023) in which the checkpoint blocks in two competing branches are finalized simultaneously. As justification and finalization require supermajority links, this would mean that more than a third of all validators have used equivocation: a slashable offence that is punishable by exclusion from the validator pool. Such an event is unlikely but not altogether impossible, and it is questionable how many the majority would need to exclude of the large number of perpetrators to obtain this. More importantly, the ambiguity created by this attack cannot be resolved without an off-chain decision, i.e., a hard fork, which is costly and damaging to businesses that use Ethereum 2.0, but also to the reputation of the system.

We also note an interesting scenario that affects finalization, even though it has not been formalized as an attack; such a scenario is highlighted in [18] (p. 46). Specifically, if attestations are always delayed by exactly one epoch, the Casper FFG algorithm may lock into a leap-frog behavior where justifications always occur one epoch late, so that no finalization takes place. This would be taken care of by the inactivity leak mode, but it can delay finalization and affect the availability of the Ethereum 2.0 blockchain.

Table 2 summarizes the attacks, their goals, and their impact, as well as the information (taken from the papers that first described the attacks) about the number of colluding validators needed for the attack.

Table 2. An overview of the attacks on the Ethereum 2.0 PoS consensus.

		Equivocation of	**Long Time Window for**		
Target Disruption	**Attack**		**Proposals**	**Attestations**	**Minimum Number of Colluding Validators**
Chain reorg	Malicious reorg [25]		×	×	Depends on the frequency of attack: e.g., an attacker with 30% stake can execute a 3-reorg approx. once per hour
	Combined reorg and balancing [26]		×	×	$2k-1$ for a k-reorg, if the attacker controls network delays $\mathcal{O}(k\sqrt{W_{honest}})$ for a k-reorg with W_{honest} honest validators under probabilistic network delay
	Avalanche attack [27]	Blocks	×	×	Not clear from the description
Finalization of a checkpoint block	Balancing attack [22]	Blocks		×	6 (proposer and five validators)
	Refined balancing attack [27]	Blocks and attestations		×	An unknown constant number sufficient to overcome the proposer boost
	Refined liveness attack [26]		×	×	$\mathcal{O}(1/\sqrt{W_{honest}})$ fraction of total stake
	Finality delay attack [25]		×	×	Depends on the duration of delay: e.g., an attacker with 30% stake can ensure one non-justified epoch with probability of 0.09
	Bouncing attack (https://ethresear.ch/t/analysis-of-bouncing-attack-on-ffg/6113, last accessed on 29 September 2023)			×	Less than 1/3 of the total stake
	Probabilistic bouncing attack [31]		×	×	Less than 1/3 of the total stake; the duration depends on the number of adversarial validators and the number of slots before which the validators are allowed to switch
Starving honest validators of their stake	Staircase attack [30]		×	×	About 29% of validators is sufficient for launching the attack repeatedly, and this will eventually starve all honest validators of their stake
Finalization of competing checkpoints	Double finality attack (https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/attack-and-defense/, last accessed on 28 September 2023)	Blocks			More than 1/3 of the total stake

5. Reducing the Attack Surface

It is generally agreed [18] that a software client that controls less than one-third of a total stake does not present a risk to the operation of Ethereum 2.0. This number might also be interpreted as the portion of the total stake controlled by an adversary through a number of colluding validators. This value is consistent with the standard values for PBFT and PBFT-based consensus algorithms. However, values in the rightmost column of Table 2 indicate that Ethereum 2.0 is vulnerable to a number of attacks launched by adversaries with less than one-third of the total stake, and this is sometimes possible by only a handful of adversarial validators. Therefore, an in-depth study of attacks and countermeasures is not a simple academic exercise but a necessity.

Evidently, the attack surface can be reduced, in some cases significantly, by taking the following steps.

1. The time period in which the block proposals are accepted needs to be shortened to (ideally) a single slot. As block proposals refer to the slot number, incorrectly proposed or delayed blocks can be immediately discarded. This would eliminate the possibility that a block is withheld and then proposed in a subsequent slot, which is the basic mechanism in a number of attacks.
2. The time period in which attestations are accepted needs to be shortened. This is a sensitive issue as it goes against the rationale adopted in the current version of Ethereum, namely that the longer time window in which attestations are accepted minimizes the impact of malicious validators (which would increase when the window is shortened). At the same time, a reduction in the required majority (as in a two-thirds supermajority of the attestations received, possibly with a predefined minimum attestation count) would ensure the smooth operation of the Casper FFG protocol. Initial experiences gained since The Merge [23] indicate that the actual number of malicious validators is small; thus, perhaps a reduction in the time for receiving attestations is a risk worth taking.
3. The procedure to slash a misbehaving validator should be streamlined by prioritizing the notification of the slashable behavior submitted by an honest validator, such that it could be taken into account as soon as possible (perhaps even in the next proposed block).
   - (3a) Honest validators that detect slashable behavior should be allowed to ignore any messages sent by the dishonest ones, including those already received, without hesitation.
   - (3b) The list of slashable offences could be extended to cover some of the behaviors described in the context of attacks listed above.
4. Finally, an increase in the stake needed to become a validator could be conducted in order. We do not advocate using the values of the order of 1500 ETH as was initially proposed [18]. However, an increase in the required stake deposit to 64 or even 100 ETH might discourage some potential attackers whilst keeping the validator role affordable for a large number of potential validators.

We note that these measures can be effected through very small changes in the code base, or even by just changing the values of some global constants.

A number of recent proposals have addressed the vulnerabilities listed above.

Goldfish, a revised fork choice rule proposed in [32], includes message buffering and allows attestations to expire. Together, the steps provide resilience to a number of reorg attacks that are possible with the original LMD-GHOST fork choice rule. Goldfish is also resilient to a number of equivocation-based attacks as the votes from validators who have sent votes for two or more different blocks are not taken into account at all. Goldfish modifies the attestation protocol by not requiring the participation of all validators in each epoch (i.e., it uses the subsampling of validators), which allows validators to leave the network (or go to sleep) and reappear as they wish. At the same time, Goldfish is sensitive to the asynchrony caused by network delays, as even simple violations of the delay assumptions can compromise the finality of previously confirmed blocks.

The Recent Latest Message Driven (RLMD) variant of GHOST [33] relaxes the vote expiry constraint to retain safety despite the asynchrony (provided it does not last too long). It also tolerates dynamic participation in accordance with the sleepy model of consensus [34]. RLMD-GHOST is the basis for the improved consensus protocol described in [35], which manages to achieve block finalization with only a single slot delay with respect to the slot in which the block has been proposed.

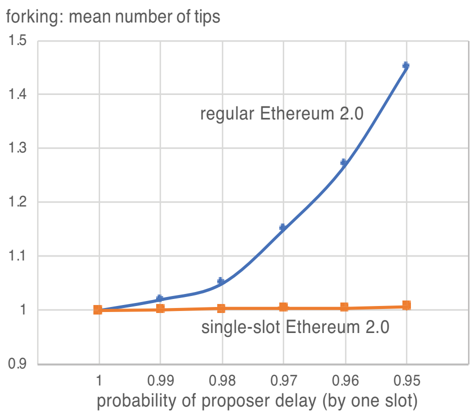
In this manner, Ethereum could be brought closer to reaching a single-slot finality, which can be thought of as the Holy Grail of Ethereum and other PoS protocols (https://notes.ethereum.org/@vbuterin/single_slot_finality, last accessed on 20 September 2023). However, as was pointed out in [35], achieving single-slot finality could turn out to be extremely costly in the Ethereum 2.0 environment due to the excessive communication and computation load: the former on account of the need to propagate the information throughout the network, and the latter on account of the need to aggregate the BLS signatures of the validator attestations.

6. Performance of the Original and Modified Consensus Protocols

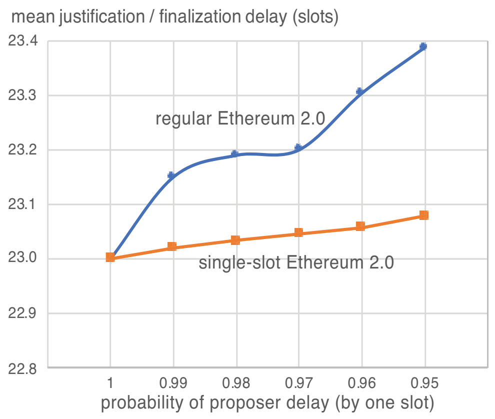
To evaluate the impact of the countermeasures proposed above, we have conducted a number of simulations with varying probabilities of delays in block proposing or attestations, or both, using a simulator based on Anylogic 8.8.5 from Anylogic, Inc., Oakbrook Terrace, IL. We considered a system with 4096 validators, and the results were obtained after running the simulator for a total of 5000 epochs. Network delays between any two nodes were randomly determined; however, they were limited to ensure that a block proposal or an attestation can safely reach the entire network within the time period of a single slot.

In our first experiment, we varied the probability that a block is proposed with a delay of one slot, as this is the maximum allowed under current Ethereum 2.0 rules, the results of which are shown in Figure 11. The left diagram shows the mean number of tips, which is good proxy measure for the probability of a deliberate fork, whereby the upper blue curve corresponds to a regular Ethereum 2.0 protocol. As can be seen, this number is surprisingly high despite all the steps taken to reduce or eliminate forking. Inserting a dummy block in each slot in which a regular block has not been proposed, as proposed in [36], reduces the mean number of tips to nearly one, as shown by the red curve below. (Delayed blocks are ignored in this case.) Evidently, any of the attacks from Table 2 that make use of delayed block proposals would be much harder to launch in the latter case.

Similar observations may be made from the diagram on the right, which shows the mean time (expressed in slots) needed to justify/finalize a checkpoint block. We used the time to justify or finalize a block as a measure of adherence to the Ethereum 2.0 protocol as it is much easier to observe than the violations of the LMD-GHOST rule, which may change from one slot to the next one. As block justification and finalization require a supermajority of two-thirds of the validator votes according to the Casper FFG rules—which are slightly thrown off-balance when a block is delayed, particularly when that block is a checkpoint block—this delay is substantially reduced, although not completely eliminated, when dummy blocks are inserted, as shown by the red curve below.



(**a**) The mean number of tips in the chain.



(**b**) The mean time to justify/finalize a checkpoint block.

Figure 11. Performance of consensus vs. the probability that a block is proposed with a one-slot delay.

In our second experiment, we showed the mean justification/finalization delay for the checkpoint blocks under varying numbers of validators that delay their attestations, whereby the maximum attestation delay was represented as a parameter (the value of which was randomly chosen between zero and the maximum). The results are shown in Figure 12.

As can be seen, the increased number of validators that delayed their attestations and the increased maximum attestation delay (and, consequently, the increased mean delay) meant that a longer period of time was needed to reach the required supermajority in the Caspet FFG protocol. This, consequently, lead to an increase in the mean time to justify/finalize a checkpoint block. In many cases, the justification/finalization time exceeded 32 slots, which meant that it extended into a subsequent epoch. The practical implication was an increased risk in confusing the Casper FFG protocol into reaching the wrong results, as extending the delay beyond the epoch in question increases the likelihood that attestations for the next pair of checkpoint blocks will start arriving, which—in turn—increases the risk of attacks on Casper FFG in the manner described above.

However, when the maximum delay at which the attestations are accepted is limited to 16, 20, and 24 slots, we obtained the curves that are shown with dashed lines in Figure 12. While this did not make a significant impact on the LMD-GHOST fork choice rule, which is dynamically calculated in each slot, it provided an upper bound for the time needed to justify/finalize a checkpoint block and, thus, reduced the risk of attacks on the justification described above—as was the original intention. It is worth noting that limiting the acceptable delay limit also requires the adjustment of the required supermajority, otherwise we increase the risk that Casper FFG will not be able to justify/finalize a checkpoint block.

Evidently, the initial results are encouraging as they show that countermeasures have an effect. Nevertheless, much more work is certainly needed before definitive conclusions on the applicability and extent of the aforementioned remedies can be made.

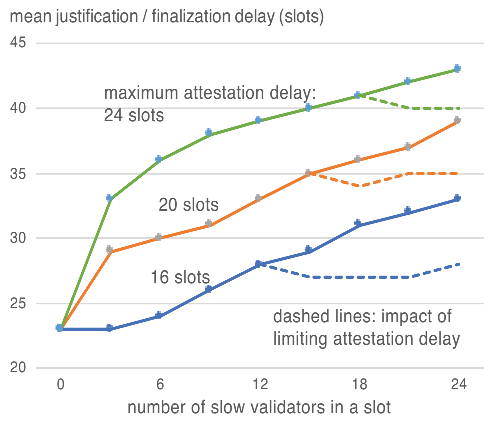


Figure 12. The mean time to justify/finalize a checkpoint block under variable attestation delay, possibly with limited delay numbers (dashed lines).

7. Conclusions

Ethereum 2.0 is a successful blockchain system that implements a Proof-of-Stake-based consensus. However, the need to decentralize decision making and the desire to limit the impact of malicious nodes in the consensus process have resulted in vulnerabilities that allow serious attacks to be launched at little cost to the attacker and with only a small number of adversarial nodes. The risks are small but, given the popularity of the Ethereum 2.0 platform, there is every reason for them to be taken seriously. Fortunately, it seems that most of the risks can be addressed, as well as their impact substantially reduced, by simple modifications to the underlying protocols for fork choice rule and block justification/finalization.

Author Contributions: Conceptualization, V.B.M.; methodology, V.B.M., S.N.M., J.M. and X.C.; formal analysis, V.B.M. and S.N.M.; writing: V.B.M., S.N.M., J.M. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSERC Discovery Grants—Individual (V.B.M., J.M.) and by National Natural Science Foundation of China under Grant 62272028 (X.C.).

Data Availability Statement: There is no data available for this research.

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

References

1. Buterin, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2017. Available online: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (accessed on 11 January 2024).
2. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
3. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://www.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf (accessed on 11 January 2024).
4. Buterin, V.; Griffith, V. Casper the Friendly Finality Gadget. *arXiv* **2019**, arXiv:1710.09437.
5. Sompolinsky, Y.; Zohar, A. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. Cryptology ePrint Archive, Report 2013/881. 2013. Available online: https://eprint.iacr.org/2013/881 (accessed on 11 January 2024).
6. Pease, M.; Shostak, R.; Lamport, L. Reaching agreement in the presence of faults. *J. ACM (JACM)* **1980**, *27*, 228–234. [CrossRef]
7. Castro, M.; Liskov, B. Practical Byzantine Fault Tolerance. In Proceedings of the OSDI: Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999.
8. Aublin, P.L.; Guerraoui, R.; Knežević, N.; Quéma, V.; Vukolić, M. The next 700 BFT protocols. *ACM Trans. Comput. Syst. (TOCS)* **2015**, *32*, 12. [CrossRef]
9. Decker, C.; Wattenhofer, R. Information propagation in the Bitcoin network. In Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing (P2P'13), Trento, Italy, 9–11 September 2013; Volume 26.
10. Mišić, J.; Mišić, V.B.; Chang, X. On ledger inconsistency time in Bitcoin's blockchain delivery network. In Proceedings of the IEEE Globecom 2019, Waikoloa, HI, USA, 9–13 December 2019.
11. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. *arXiv* **2013**, arXiv:1311.0243.
12. Abdellatif, T.; Brousmiche, K.L. Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018.
13. Krichen, M.; Lahami, M.; Al-Haija, Q.A. Formal Methods for the Verification of Smart Contracts: A Review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022.
14. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]. *ACM SIGMETRICS Perform. Eval. Rev.* **2014**, *42*, 34–37. [CrossRef]
15. King, S.; Nadal, S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. 19 August 2012. Available online: https://peercoin.net/assets/paper/peercoin-paper.pdf (accessed on 11 January 2024).
16. Weerapanpisit, P.; Trilles, S.; Huerta, J.; Painho, M. A Decentralized Location-Based Reputation Management System in the IoT Using Blockchain. *IEEE Internet Things J.* **2022**, *9*, 15100–15115. [CrossRef]
17. Larimer, D. Delegated Proof of Stake (DPoS). 2018. Available online: https://tokens-economy.gitbook.io/consensus/chain-based-proof-of-stake/delegated-proof-of-stake-dpos (accessed on 11 January 2024).
18. Edgington, B. Upgrading Ethereum: A Technical Handbook on Ethereum's Move to Proof of Stake and Beyond; Online PDF Edition 0.3: Capella [WIP]. Ethereum Foundation. 2023. Available online: https://eth2book.info/capella/ (accessed on 11 January 2024).
19. Buterin, V.; Hernandez, D.; Kamphefner, T.; Pham, K.; Qiao, Z.; Ryan, D.; Sin, J.; Wang, Y.; Zhang, Y.X. Combining GHOST and Casper. *arXiv* **2020**, arXiv:2003.03052.
20. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, QLD, Australia, 9–13 December 2001; pp. 514–532.
21. Otsuki, K.; Nakamura, R.; Shudo, K. Impact of saving attacks on blockchain consensus. *IEEE Access* **2021**, *9*, 133011–133022. [CrossRef]
22. Neu, J.; Tas, E.N.; Tse, D. Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma. *arXiv* **2021**, arXiv:2009.04987.
23. Grandjean, D.; Heimbach, L.; Wattenhofer, R. Ethereum Proof-of-Stake Consensus Layer: Participation and Decentralization. *arXiv* **2023**, arXiv:2306.10777.
24. Karame, G.O.; Androulaki, E.; Roeschlin, M.; Gervais, A.; Čapkun, S. Misbehavior in Bitcoin: A study of double-spending and accountability. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2015**, *18*, 1–32. [CrossRef]
25. Neuder, M.; Moroz, D.J.; Rao, R.; Parkes, D.C. Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders. *arXiv* **2021**, arXiv:2102.02247.
26. Schwarz-Schilling, C.; Neu, J.; Monnot, B.; Asgaonkar, A.; Tas, E.N.; Tse, D. Three Attacks on Proof-of-Stake Ethereum. *arXiv* **2021**, arXiv:2110.10086.
27. Neu, J.; Tas, E.N.; Tse, D. Two More Attacks on Proof-of-Stake GHOST/Ethereum. In Proceedings of the ConsensusDay '22, Los Angeles, CA, USA, 7–11 November 2022; pp. 43–52.
28. Naderi Mighan, S.; Mišić, J.; Mišić, V.B.; Chang, X. An In-Depth Look at Forking-Based Attacks in Ethereum with POW Consensus. *IEEE Trans. Netw. Serv. Manag.* **2023**. [CrossRef]

29. Heo, H.; Woo, S.; Yoon, T.; Kang, M.S.; Shin, S. Partitioning Ethereum without Eclipsing It. In Proceedings of the Network and Distributed System Security (NDSS) Symposium, San Diego, CA, USA, 27 February–3 March 2023.
30. Zhang, M.; Li, R.; Duan, S. Max Attestation Matters: Making Honest Parties Lose Their Incentives in Ethereum PoS. Cryptology ePrint Archive, Paper 2023/1622. 2023. Available online: https://eprint.iacr.org/2023/1622 (accessed on 11 January 2024).
31. Pavloff, U.; Amoussou-Guenou, Y.; Tucci-Piergiovann, S. Ethereum Proof-of-Stake and the Probabilistic Bouncing Attack. *arXiv* **2023**, arXiv:2210.16070.
32. D'Amato, F.; Neu, J.; Tas, E.N.; Tse, D. Goldfish: No More Attacks on Proof-of-Stake Ethereum. *arXiv* **2023**, arXiv:2209.03255.
33. D'Amato, F.; Zanolini, L. Recent Latest Message Driven GHOST: Balancing Dynamic Availability with Asynchrony Resilience. *arXiv* **2023**, arXiv:2302.11326.
34. Pass, R.; Shi, E. The sleepy model of consensus. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 380–409.
35. D'Amato, F.; Zanolini, L. A Simple Single Slot Finality Protocol For Ethereum. *arXiv* **2023**, arXiv:2302.12745.
36. Chan, B.Y.; Pass, R. Simplex Consensus: A Simple and Fast Consensus Protocol. Cryptology ePrint Archive, Paper 2023/463. 2023. Available online: https://eprint.iacr.org/2023/463 (accessed on 11 January 2024).

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