Impact of Saving Attacks on Blockchain Consensus

KAI OTSUKI¹, RYUYA NAKAMURA², AND KAZUYUKI SHUDO¹ (Senior Member, IEEE)

¹Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo 152-8552, Japan
²Research and Development, LayerX Inc., Tokyo 103-0004, Japan

Corresponding author: Kai Otsuki (ootuki.k.aa@m.titech.ac.jp)

This work was supported in part by the SECOM Science and Technology Foundation, and in part by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP21H04872.

ABSTRACT

Blockchain consensus, which enables nodes on a peer-to-peer network to agree on the same ledger history, is the core element of blockchain systems. In many blockchain systems, a node chosen as a block proposer, in accordance with the consensus protocol, generates a block, and each node chooses a chain to extend by a fork-choice rule. This study introduces saving attacks, a new kind of attack that prevents nodes from reaching a consensus. In saving attacks, the adversary “saves” its rights to propose blocks during a temporal consensus failure and utilizes them later to cause another consensus failure. As a result, the blockchain suffers from poor performance and high latency to block finalization. We study the effect of saving attacks on various fork-choice rules, including those that Ethereum 2.0 plans to employ. We simulate saving attacks on the longest-chain rule, Greedy Heaviest-Observed Sub-Tree (GHOST), latest-message-driven (LMD) GHOST, and fresh-message-driven (FMD) GHOST. We show that the saving attack has a very large negative impact on the consensus. For example, we observe that under a certain condition, an adversary with 30% of the total stake that has saved its blocks for 32 minutes succeeds in preventing a consensus against LMD GHOST for 83 minutes in the context of Ethereum 2.0. We also show that FMD GHOST decreases the attack duration to approximately 6.4 minutes under the same conditions. Our results are applicable to all slot-based proof-of-stake blockchains, not just Ethereum 2.0.

INDEX TERMS

Ethereum, Ethereum 2.0, fork-choice rule, proof-of-stake, network simulation.

I. INTRODUCTION

A consensus is the core part of a blockchain system and significantly contributes to blockchain security. In the consensus of many public blockchains, block proposers are selected according to the consensus protocol and employ a fork-choice rule to decide which chain should be extended. Bitcoin, the first blockchain project, introduced the Nakamoto consensus, in which the block proposer selection is based on a proof-of-work (PoW) algorithm and the fork-choice rule is the longest-chain rule [1]. After Bitcoin, fork-choice rules became the center of attention. For example, Greedy Heaviest-Observed Sub-Tree (GHOST) was proposed to improve blockchain security when more transactions were processed per second [2].

Ethereum, the second-largest market capitalization blockchain platform, is now undergoing a protocol upgrade through a project called Ethereum 2.0 [3]. The developer community of Ethereum 2.0 proposed two new fork-choice rules, i.e., latest-message-driven (LMD) GHOST [4] and fresh-message-driven (FMD) GHOST [5]. Ethereum 2.0 uses “Gasper” as its consensus [4]. We call this kind of consensus protocol slot-based proof-of-stake (PoS), where nodes are allocated to different slots randomly and propose new blocks at their allocated slots. In Gasper, FMD GHOST is employed as its fork-choice rule as a result of our preliminary work with them, although Gasper was originally scheduled to employ LMD GHOST. The use of fork-choice rules such as GHOST, LMD GHOST, and FMD GHOST in slot-based PoS protocols, including Gasper, has not been investigated sufficiently, even though some of them are actually in operation.

This study introduces a new kind of attack called a “saving attack” and investigates the robustness of fork-choice rules against it. In saving attacks, the adversary “saves” its rights to propose blocks during a temporal consensus failure and utilizes them to cause another consensus failure after the...
network reverts to the normal synchrony state. These attacks lead to low system capabilities and high latency to block finalization.

We simulate the saving attack to examine it. To our knowledge, we investigate for the first time the impact of the saving strategy on fork-choice rules. Although simulation is known to be effective in investigating blockchain performance and security and has been applied in a number of previous studies [6]–[8], most of these studies focused on PoW blockchains. Although Neu et al. [9] conducted a simulation of Gasper, they did not simulate the peer-to-peer (P2P) network, and the simulation was quite simplistic. Thus, this work is the first to perform a realistic PoS simulation reproducing a P2P network.

In our simulation, we adopt a general slot-based PoS consensus algorithm. Therefore, our results are adaptable to all blockchains with slot-based PoS, such as Ethereum 2.0 and all future blockchain platforms adopting the algorithm.

We perform extensive simulations varying parameters such as the attacker rate ($\gamma$), the length of the saving epoch ($k$), and the initial two-conflicting-chain score ratio ($\rho$). Then, as the results of our research, we present our significant discoveries and findings, which can be summarized as follows.

- The saving attack can be critical for LMD GHOST. In the worst-case scenario, the adversary succeeds in continuing the saving attack for more than 13 epochs, which would be equivalent to 83 minutes if Ethereum 2.0 employed LMD GHOST. The attack duration becomes longer as the number of block-proposal rights saved by the adversary increases.
- FMD GHOST is resilient to the saving attack. Regardless of the length of the adversary’s saving epochs, the performance of FMD GHOST is constant. It decreases the attack duration from 13 epochs to approximately 0.5 to 1.0 epochs under the above same
- GHOST is stronger against the saving attack than is LMD GHOST, excluding minor exceptions.
- The longest-chain rule performs the best against the saving attack in all situations. We find that the longest-chain rule requires at least one-third of all nodes to initiate the saving attack.
- When the initial score difference between conflicting chains becomes larger, the adversary needs more stakes to initiate the saving attack, and the attack duration is likely to diminish.

Our results are significant for two key reasons.

First, our analysis has social significance. Following Bitcoin, most of the current blockchain systems employ PoW. However, previous studies have suggested that PoW possesses significant shortcomings, e.g., energy consumption and centralization [10]. The most promising alternative is PoS, and the number of PoS blockchains has rapidly grown. This tendency is expected to continue. Our results, which correspond to PoS blockchain consensus, will help developers make engineering decisions. Moreover, our results are adaptable to Ethereum 2.0. Having an impact on this large blockchain is quite socially significant.

Second, we provide numerical results from a realistic simulation. We simulate a P2P network with real-world network latency and bandwidth by using SimBlock, which is a blockchain network simulator utilized in a number of studies [6], [11], [12]. Moreover, we assume modest adversary capabilities. Security studies tend to make strong assumptions about an adversary’s capabilities. For example, the study by New et al. [9], which simulated an attack on a slot-based PoS blockchain, had strong and unrealistic assumptions about the adversary’s capabilities, i.e., the adversary could control message dissemination to honest validators and knew when each honest validator received messages. In reality, these assumptions are impractical because, for example, they require the adversary to seclude honest validators from its controlled nodes. The saving attack is possible without the adversary being able to control message propagation at the P2P layer. We also assume that the adversary is indifferent about the message distribution to honest validators. In short, we consider the adversary to possess the same ability in regard to message delivery as that of honest validators. In addition, we present theoretical discussions to support some of the experimental results.

The remainder of the paper is organized as follows. In Section II, we overview the basic concepts, e.g., the fork-choice rules and saving attack. In Section III, we introduce our network and adversary models for our analysis. In Section IV, we present our theoretical and experimental results, obtained using the attack duration, which is a key metric representing the length of time during which an adversary can continue the saving attack. Then, we discuss the adaptability of our results to Ethereum 2.0 in Section V. In Section VI, we compare our contributions with related work. Finally, we conclude this paper in Section VII.

II. BACKGROUND

In this section, we first formalize the variables and values used in this work. Then, we describe the fundamentals.

A. FORMALIZATION

The following is a set of the variables and values we use in this study as appropriate.

- $H$: Honest validator set.
- $A$: Adversarial validator set.
- $V$: Validator set (i.e., $V = H \cup A$).
- $n$: # of validators (i.e., $n = |V|$).
- $f$: # of adversarial validators (i.e., $f = |A|$).
- $B$: A set of globally observed blocks, each of which has a validator and slot (i.e., $b.validators \in V$, $b.slot \in N$).
- $b_c, b_s, b_h \in B$: $b_c$, which is on the chain with the second-best score, is the block to which the adversary appends its blocks. $b_s$ is the block at the same height of $b_c$ on the current canonical chain; and $b_h$ is the current head block at a node.
2.0 uses a consensus protocol with a type of slot-based PoS, then select a parent block by the longest-chain rule. Ethereum determines the block proposers by the PoW; these proposers might deviate from the norm to obtain benefits. Honest validators obey a set of protocols, and adversarial validators do not. Herein, we consider two kinds of validators: honest validators and adversarial validators.

B. GASPER

The purpose of blockchain consensus is to enable participants to agree on a common ledger history. We refer to entities involved in the process as validators. Here, we consider two kinds of validators: honest validators and adversarial validators. Honest validators obey a set of protocols, and adversarial validators might deviate from the norm to obtain benefits.

The Nakamoto consensus, Bitcoin’s consensus protocol, determines the block proposers by the PoW; these proposers then select a parent block by the longest-chain rule. Ethereum 2.0 uses a consensus protocol with a type of slot-based PoS and FMD GHOST.

In Gasper, time is separated into epochs, each of which comprises 32 slots of 12 s each. In an epoch, validators are randomly divided into 32 groups called committees, which are then allocated to different slots by a random number generation algorithm called RANDAO. A member of a committee is a block proposer. A block proposer selects the parent block by a fork-choice rule and makes a block on top of that at the beginning of its allocated slot. Apart from a block, each of the committee members has the right to vote, called an attestation, and votes for a chain by issuing it. In Gasper, the score of a chain is calculated as the number of attestations on the chain. All the validators have exactly one attestation in an epoch, as all of them are allocated into one of the slots of an epoch without duplication. Figure 1 gives an overview of the slot-based consensus.

Casper the Friendly Finality Gadget (FFG) [13], which Gasper employs, is similar to practical Byzantine-fault-tolerance (PBFT) [14] protocols in terms of its liveness condition [15]. PBFT protocols satisfy both safety and liveness under the partial synchrony model. However, Casper FFG requires stronger assumptions than the standard partial synchrony model. Although Casper FFG assumes asynchrony until the GST, which is similar to other PBFT protocols, it requires stronger assumptions than synchrony after the GST [4], unlike other PBFT protocols. Therefore, Casper FFG does not satisfy both safety and liveness under partial synchrony.

C. FORK-CHOICE RULES

Herein, we describe the other core of blockchain consensus: the fork-choice rules. Two fundamental fork-choice rules exist: the longest-chain rule and GHOST. The longest-chain rule is the first and simplest rule introduced by Bitcoin. As Algorithm 1 shows, the basic strategy is to select the longest chain at each branch. The second rule is GHOST, which selects the heaviest chain as the canonical chain, as shown in Algorithm 2. GHOST was invented to maintain security better than the longest-chain rule when more transactions are processed per second [2]. Thus, in general, GHOST is more resilient to attacks than is the longest-chain rule. The new variants of GHOST are LMD GHOST and FMD GHOST, which the Ethereum community newly introduced.
**Algorithm 3 LMD GHOST**

1: procedure getCanonicalChain(b)
2: \( b \leftarrow \text{Genesisblock} \)
3: while \( b.\text{children} \neq \emptyset \) do
4: \( b \leftarrow \text{arg max}_{c\in b.\text{children}} \text{getScore}(c, V) \)
5: return \( b \)
6: end
7: procedure getScore(b, S)
8: \( \text{score} = 0 \)
9: if \( b.\text{proposer} \in S \) and isValid(b) then
10: \( \text{score} = 1 \)
11: return \( \text{score} + \sum_{c\in b.\text{children}} \text{getScore}(c) \)
12: end
13: procedure isLatest(b)
14: if \( b \) is the latest block of the \( b.\text{proposer} \) then
15: return True
16: else
17: return False

in the Ethereum 2.0 project. Standard GHOST and these two rules are different in that they have some additional requirements for blocks to contribute to a chain score. LMD GHOST permits only the latest message of each validator to be valid and included in the calculation of chain weights. In FMD GHOST, the latest message of each validator has to be one allocated in the current epoch or the previous epoch to be included in the calculation of chain weights. Therefore, validators have to issue messages within at most two epochs after they are given the rights to issue a block or an attestation. Their algorithms are displayed in 3, 4. In contrast to the longest-chain rule and GHOST, for these two new variants, the minority can never “beat” the majority regardless of how many messages are sent. In the longest-chain rule and GHOST, the minority can beat the majority by connecting more blocks to their chain than connected by the majority.

**D. SimBlock**

We examine the performances of fork-choice rules by simulation, which is an effective way to achieve a quantitative and realistic investigation. We utilize SimBlock as a simulator. SimBlock is a relatively new blockchain network simulator and reproduces a blockchain P2P network in detail [16]. Because the underlying P2P network plays an important role in the saving attack, this simulator is appropriate for our purpose. It has been utilized in a number of studies [6], [11], [12], and its values, such as the network latency and bandwidth, were updated in 2020 [12]. To the best of our knowledge, this simulator is the only one whose network settings have been updated to simulate the current network state.

**E. SAVING ATTACK**

Finally, we expound on the saving attack, which is one of the main parts of our work. An overview of this attack is shown in Figure 2. We define a saving attack as an attack with saving strategies. We call the saving attack targeted in this study “the saving attack”. The primary purpose of the attack is to deprive liveness. In the saving attack, the adversary needs to have multiple conflicting chains and withheld rights to propose blocks. The adversary enforces the withheld rights and sways honest validators by shifting the canonical chain from one chain to another. The number of attacks that the adversary attempts is different depending on the fork-choice rule used. When either the longest-chain rule or GHOST is used, the attack is one-off, as there is no need to save a part of its rights. In LMD/FMD GHOST, the adversary attempts multiple attacks by using only an epoch of its rights each time. For these two rules, using all the saved rights at one time is meaningless since LMD/FMD GHOST permits validators to have only one voting score and multiple blocks of the same validator are invalid. As a result of this attack, the blockchain suffers from poor performance and high latency to block finalization.

**III. METHODOLOGY**

The goal of our study is to assess the performances of fork-choice rules against the saving attack. We provide both theoretical discussions and simulation results. Although mathematical proofs are appropriate to assure a lower bound, they are susceptible to pessimistic results, or the results can be far from the actual results because they require model simplification. On the other hand, a simulation is suitable to provide a realistic assessment. Many studies have used simulations to evaluate blockchain security and performance.

In this section, we describe the fundamental knowledge regarding our simulation and theoretical discussions. First, we expound on our network model and the adversary model. Then, we define the attackable states of the saving attack and clarify the elements that we aim at in our analysis. Finally, we show the simulation settings and a key metric for the results section.

**A. NETWORK MODEL**

We conduct a slot-based PoS simulation of Ethereum 2.0. Since we selected SimBlock as a simulator, our basic network...
model follows it. We need to modify the consensus layer to simulate a slot-based PoS algorithm. The network model is as follows. Nodes exist on the network in one of six regions, as shown in Table 1. They connect with other nodes according to the distributions given in Table 1 and exchange messages in three steps. First, nodes send Inv messages that ask if receivers have obtained the blocks in question. When the receivers do not have the block, they reply with Req messages. Finally, nodes that receive Req messages send the block to the requesters. The block size is a value measured for situations that suffice to launch the saving attack. Here, we define the set of sufficient states, which we call the adversary can share them beforehand, as block proposers are randomly fixed at the start of each epoch.

1) ASSUMPTIONS ABOUT THE ADVERSARY
Herein, we describe the adversary model. The following are the assumptions made regarding the capabilities of the adversary.

1) The adversary controls the \( \gamma \) of all nodes.
2) Message delivery between the nodes of the adversary is negligible.

The second assumption is not unrealistic. In practice, for instance, the adversary is capable of that if all the adversarial nodes are controlled by a single entity. In addition, attackers can construct their private network similar to a relay network [19]. Regarding blocks issued by the adversary, the adversary can share them beforehand, as block proposers are randomly fixed at the start of each epoch.

2) ATTACKABLE STATE SET
To initiate the attack, an adversary needs to create or wait for situations that suffice to launch the saving attack. Here, we define the set of sufficient states, which we call \( \Omega \). We define \( \Omega \) as the set that satisfies all the following conditions.

1) \( \exists k \geq 1 \) s.t. multiple conflicting chains that have not satisfied the convergence conditions for the last \( k \) epochs exist at some height.
2) The adversary has unused block-proposal rights that it saved within the \( k \) epochs.

Herein, we demonstrate possible real-world scenarios that can cause the states included in \( \Omega \). One of the conceivable subsets is a set of network-partitioned states, i.e., states where validators are separated into several groups in the network for some reason, and these groups extend their own chains separately. For example, unintentional network failure or intentional network failure, such as engineering errors or routing attacks [20], [21], can cause these states. For instance, communication failure, which occurred in one of the Ethereum 2.0 testnets, i.e., the Medalla testnet, caused network partitions [22]. In the incident, nodes running the Prysm client,
which is the most popular client testnet of Ethereum 2.0, failed at the same time due to vulnerabilities stemming from its implementation. At the time of the incident, over 65% of the nodes were running the Prysm client, which resulted in a network catastrophe once all Prysm nodes went down. Hence, severe network partitions occurred. As seen from the above example, the states contained in $\Omega$ are in real-life danger. Moreover, the following scenario is possible. The adversary has a majority of the total stakes for a short period, and it consumes a part of its block-proposal rights to make and maintain a chain-conflicting state. For example, at the launch of new blockchain platforms, communities often distribute coins to gather users. Under these circumstances, an adversary can temporarily possess a majority of the total stakes fairly easily.

3) ATTACK INITIATION STATE

In our analysis, we address the states where two conflicting chains exist and the adversary has $k$ epoch block-proposal rights, i.e., $kf$ blocks. The left figure in Figure 2 shows the state in which the adversary begins its attack. For simplicity, we make two assumptions. First, we assume that the number of conflicting chains is 2. Second, we assume that the ratio of the initial score of $b_c$ to that of $b_h$ is $\rho$ to $1 - \rho$. The parameter settings are summarized in Table 2. We set $k$ to 1, 3, and 5 to focus on plausible situations. In an actual network, the adversary is more likely to have small savings such as 1, 3 and 5. Regarding $\gamma$, we consider the adversary’s rate to be smaller than one-third because PBFT blockchains including Ethereum 2.0 are based on the assumption that the number of attackers does not surpass a third of all the nodes. To simulate various situations, we also consider two patterns of $\rho$, as the actual situation does not always involve a fifty-fifty initial score.

Note that our model can be captured by the partial synchrony model in the network model context. In the situations we address, we can pinpoint the moment at which the adversary kicks off the saving attack as the GST. Until this time, messages may not propagate to all nodes, and two conflicting chains grow while the adversary saves its block-proposal rights. After the GST, the network reverts to the normal state, in which messages by a validator are received by any other validator within 1 slot. Therefore, our model meets the definition of partial synchrony.

4) ADVERSARY’S STRATEGY

Next, we describe the adversary’s strategy. When the adversary kicks off the attack by saving block-proposal rights equivalent to $k$ epochs, it behaves as shown in Algorithm 5. At the starting time of the honest validator’s slots, the adversary checks the attack condition defined by the function in Algorithm 5. The condition is different depending on the fork-choice rule chosen. If the condition is satisfied, the adversary commands all adversarial validators to propagate adversarial blocks to sway honest validators. At this time, the adversarial blocks start to spread from all the adversarial validators simultaneously. At its slots, the adversary only withholds rights.

**Algorithm 5 Adversary’s Behavior**

1: while A new slot comes do
2: if (A block proposer at the slot) $\in A$ then
3: add the right to $Av$ (withholds the right)
4: else
5: if checkAdversarialCondition() is True then
6: propagate adversary’s blocks from all $a \in A$
7: end
8: procedure checkAdversarialCondition
9: if The longest-chain rule is used then
10: return getScore($b_h$, $V$) $\geq |Av| - 1$
11: if GHOST is used then
12: return getScore($b_h$) - getScore($b_v$) $\geq |Av| - 2$
13: if Either LMD GHOST or FMD GHOST is used then
14: return getScore($b_h$, $V$) $\geq |Av|\cap Av.validators|$ $\geq \frac{|V|}{2} - 2$

**Algorithm 6 Convergence Detector**

1: while A request comes do
2: for $h$ in $H$ do
3: if checkConvergence is False at $h$ then
4: return False
5: return True
6:
7: procedure checkConvergence
8: if The longest-chain rule is used then
9: return currentBlockHeight $>$ |Av|
10: if GHOST is used then
11: return getScore($b_1$) - getScore($b_2$) $>$ |Av|
12: if Either LMD GHOST or FMD GHOST is used then
13: return getScore($b_1$, $H$) $>$ $|V|$ $\frac{1}{2}$

C. ATTACK DURATION

Here, we explain our key metric: the attack duration. We define the attack duration as the length of epochs in which the adversary successfully prevents consensus. In other words, the attack duration is the length of time from the initiation of the saving attack to when the convergence condition is satisfied for the first time. We use Convergence Detector to check whether the convergence condition is satisfied. Convergence Detector indicates whether the convergence condition is satisfied at a given height to requesters. It works according to Algorithm 6. When all the honest validators believe that the conflict has converged, it judges the convergence condition as met. The convergence conditions vary depending on the fork-choice rule chosen. Under the longest-chain rule, Convergence Detector checks whether the score of the current canonical chain is larger than the adversary’s saved rights to propose blocks. Regarding GHOST, it checks whether the difference in the score between two conflicting
chains exceeds the savings of the adversary. Under LMD and FMD GHOST, it confirms that the total honest validator’s score amounts to half the maximum score.

We select the attack duration as a key metric because it is the most essential to evaluate the saving attack. The fundamental target of the saving attack is liveness. The adversary acts to delay the finality for as long as possible. Therefore, the length of time that the adversary can maintain a chain-conflicting situation corresponds to the primary purpose.

D. FEASIBILITY OF PRESENTED ALGORITHMS

This subsection discusses the feasibility of algorithms presented in this paper. Algorithms 1, 2, 3 and 4 are existing fork-choice rules. Algorithms 5 and 6 are what we introduced in this paper. Algorithm 5 is adversary’s behavior and an adversary possibly utilizes Algorithm 5 to determine the timing to start the attack. Our simulator runs Algorithm 6 to check the chain convergence.

The bottleneck of Algorithms 5 and 6 in terms of computational complexity boils down to four fork-choice rules used in them. It is still open problem to clarify their computational complexity. What we can say today is that the two of them are implemented and running in real-world blockchain systems. The longest-chain rule works in Bitcoin [1] and FMD GHOST works in Ethereum 2.0 [3]. The remaining two algorithms, GHOST and LMD GHOST have not been employed in a real-world system. However, these two rules are similar to FMD GHOST and more lightweight than it because FMD GHOST requires additional conditions on top of ones required in GHOST and LMD GHOST as we checked in Section II.

IV. RESULT

In our study, we conduct two kinds of analysis: theoretical analysis and an experimental investigation. In this section, we first explain the mathematical results. Then, we expound on the experimental results, referring to the theoretical results when needed. The parameter settings are described in Section II.

A. THEORETICAL RESULTS

This subsection gives theoretical evaluations of each fork-choice rule.

1) THE LONGEST-CHAIN RULE

The resilience of the longest-chain rule against saving attacks has not been investigated. However, the longest-chain rule seems resilient to attackers with saving strategies because executing withheld block-proposal rights requires the adversary to make a new longest chain from the height where the two conflicting chains emerged. In this part, we theoretically evaluate the longest-chain rule and demonstrate that it has high resilience against the saving attack, as surmised.

First, we assess its attack duration, which is represented as \(\frac{k\rho}{n-f} \rho \left(\frac{2}{n-f+1}\right)\). This formula is introduced from \(k(n-f)\rho + n-f x = k\rho + n-f x\), where \(x\) which represents the number of slots, is the variable to solve. This equation represents the time at which the convergence condition is met. The left side represents the score of \(b_c\), and the right side represents the score that the adversary has saved until the current slot. Solving the equation and then converting slots to epochs by dividing \(x\) by 100, we obtain the theoretical attack duration. From the formula, in some situations, the convergence condition will have already been met by the time the adversary to initiate an attack. Thus, we attempt to introduce the minimum \(\gamma\) required to learn the initiation condition. The formula is represented as \(\gamma \geq \frac{b_c}{\rho}\), and the results are summarized in Figure 3 and Table 3. When the initial score of \(b_c\) surpasses the savings that the adversary has accumulated, the adversary cannot initiate the saving attack. Thus, the formula is derived from the inequality \(k(n-f)\rho \leq kf\). In two-conflicting-chain situations, the worst case for honest validators is where \(\rho\) equals 0.5. Even in this case, the attack initiation requires \(r \leq 0.33\), which is more than 30% of all the validators. This result shows that the longest-chain rule is quite robust against the saving attack.

2) GHOST

One attack relatively similar to the saving attack is the balance attack [8], which was introduced as an attack suitable for preventing consensus against GHOST. Both this attack and the saving attack take advantage of a characteristic of GHOST in which orphan blocks contribute to the weight of their chains. Thus, GHOST seems vulnerable to both the saving attack and the balance attack. Here, we theoretically evaluate GHOST.

To evaluate the attack duration, the formula is \(\frac{k(f-(n-f)(2p-1))}{n-2f}\), which is derived as in Section IV-A1. Moreover, we obtain the minimum required adversarial ratio as discussed in Section IV-A1, and its formula is represented as \(\gamma \geq 1 - \frac{1}{2p}\), which is independent of \(k\). Figure 4 depicts the relation between \(\rho\) and the minimum required ratio of attackers. When \(\rho\) is no greater than 0.5, the adversary can kick-start the attack regardless of \(\rho\). However, the minimum value becomes larger as \(\rho\) increases. For example, at least 0.1667 is needed to initiate the attack when \(\rho = 0.6\).

Here, the comparisons with the longest-chain rule are worth noting. First, GHOST is far more susceptible to the saving attack than the longest-chain rule. A comparison of Figure 3 with Figure 4 demonstrates that the adversary requires a far larger \(\rho\) to initiate the attack. For example, when \(\rho = 0.6\), the adversary against the longest-chain rule needs
an adversarial ratio larger than 0.375, which is more than twice that of GHOST. Therefore, as with the balance attack, GHOST is more susceptible to the saving attack.

3) LMD/FMD GHOST

Last, we briefly consider LMD and FMD GHOST. Unfortunately, the evaluations of their attack durations are challenging because of the complexity of LMD and FMD GHOST. For example, the attack is not one-off, and the number of attacks is based not only on the number of rights saved by the adversary but also on whose blocks are currently on the canonical chain. Therefore, we consider only the minimum required adversarial ratio here, and we analyze the attack duration against LMD GHOST and FMD GHOST by simulation. The minimum required adversarial ratio is \( \gamma \geq 1 - \frac{1}{k} \), which is independent of \( k \). Its introduction is the same as that for the previous discussions. The minimum required adversarial ratio of GHOST-family rules, i.e., GHOST, LMD GHOST, and FMD GHOST, are common.

B. EXPERIMENTAL RESULTS

In this subsection, we check the experimental results, using the abovementioned theoretical results as appropriate. In our experiments, we use two patterns of \( \rho \): 0.5 and 0.6. Figures 5, 6, and 7 show the results when \( \rho = 0.5 \) and \( k = 1, 3, 5 \), respectively. Figures 8, 9, and 10 show the results when \( \rho = 0.6 \) and \( k = 1, 3, 5 \), respectively. The horizontal and vertical axis represent \( \gamma \) and the attack duration, respectively.

TABLE 3. Minimal required \( \gamma \).

| \( F \backslash \rho \) | 0.5 | 0.6 |
|---------------------|-----|-----|
| Longest-chain rule  | 0.333 | 0.375 |
| GHOST               | 0   | 0.167 |
| LMD/FMD GHOST       | 0   | 0.167 |

1) VULNERABILITY OF LMD GHOST AGAINST THE SAVING ATTACK

The saving attack was extremely dangerous for LMD GHOST. For example, when \( \gamma = 0.3 \) and \( k = 3 \) or 5, the attack duration was more than 7 and 13 epochs, respectively, and the adversary succeeded in continuing its attack for twice as long as the length of its saving epoch \( k \) in Figure 6 and 7. Although the adversary was not able to maintain the attack duration against other fork-choice rules when \( \rho \) became 0.6, it successfully maintained the attack duration when \( \gamma = 0.3 \) to the same degree as for the results when \( \rho = 0.5 \). Figures 10 and 9 show that the attack durations were approximately 13 and 6 epochs, respectively. These numbers are quite similar to those in Figures 7 and 6. We also observe that the duration increased as \( \gamma \) rose. Clearly, all the lines of all figures monotonically increase. In particular, LMD GHOST experienced sharp increases; for instance, the value increased by approximately 26.5 times from the lowest to that of the highest \( \gamma \) in Figure 7.

Herein, we consider the reasons why LMD GHOST was more susceptible to the saving attack than the other fork-choice rules. The essential point is that the adversary can split its unused block-proposal rights and switch canonical chains multiple times. In other rules, i.e., the longest-chain rule and GHOST, the scores of chains monotonically increase as new blocks are added to the chains. However, in LMD GHOST, where each validator has exactly one weight in chain score calculation because only the latest messages are valid, the score of block \( b_c \) increases by one and the score of block \( b_s \) decreases by one when a validator that has proposed a block on the chain of \( b_s \) proposes a block on the chain of \( b_c \). This difference enables the adversary against LMD GHOST to switch canonical chains multiple times and sway honest validators between the two chains for a longer period. The consumption rate of unused block-proposal rights is another reason. When the adversary has saved the same amount of rights, the rate of LMD GHOST is slower than that of GHOST. In GHOST, every time an honest validator adds new blocks to the chain of block \( b_c \), the score increases, and the attack time approaches. In LMD GHOST, every time an honest validator that contributed to the chain of block \( b_s \) adds new blocks to the chain of block \( b_c \), the attack time grows closer. Thus, the rate is faster in GHOST, and the adversary spends block-proposal rights earlier.

2) ROBUSTNESS OF FMD GHOST

We also find that the employment of LMD GHOST significantly decreases the risk of the saving attack. As shown in Figures 7 and 10, FMD GHOST decreased the attack...
duration to 1.1 and 0.4 epochs, respectively, from approximately 13 epochs, which was the attack duration of LMD GHOST. The characteristics of FMD GHOST improved this number. In FMD GHOST, the validity period of block proposal rights is at most two epochs, which forbids the adversary from lengthening the attack duration in proportion to \( k \). Actually, the values of FMD GHOST are always consistent at each \( \gamma \). Notably, the graphs of FMD GHOST are always below those of GHOST and LMD GHOST when \( k = 3, 5 \). Thus, FMD GHOST is effective at alleviating the risk of the saving attack.

3) INCOMPATIBILITY OF THE SAVING ATTACK WITH THE LONGEST-CHAIN RULE

Another observation we can make is the incompatibility of the saving attack with the longest-chain rule. The longest-chain rule is the most robust to this attack in all cases. As Figure 3 shows, the adversary could not initiate the saving attack under the simulated \( \rho \). Therefore, the saving attack is not suitable for preventing consensus against the longest-chain rule. Note that the longest-chain rule, FMD GHOST, GHOST, and LMD GHOST were more resilient to the saving attack in this order when \( k \geq 1 \), as shown in Figures 6, 7, 9 and 10.

4) SAVING ATTACK AGAINST GHOST

Clearly, GHOST is more robust to the saving attack than LMD GHOST except when both \( k \) and \( \rho \) are small. For example, when \( k \) was either 3 or 5, the attack duration against GHOST was shorter than that against LMD GHOST for the most part, as shown in Figures 6, 7, 9 and 10. When \( k \) was 1, the attack duration against GHOST was longer than that of LMD GHOST until \( \rho \) surpassed 0.2 in Figure 5 and 8. However, in these cases, the attack duration was no more than approximately 0.33 epochs.

5) COMPARISONS BETWEEN TWO DIFFERENT \( \rho \) SETTINGS

Next, we pay attention to the differences in the results under two different \( \rho \) settings. Our results show that the adversary had difficulty preventing the consensus when \( \rho \) diverged from 0.5, which is the most convenient situation for the adversary. The two fork-choice rules experienced decent drops in the duration. A comparison of values at the same \( \gamma \) but different \( \rho \) reveals that the attack duration of GHOST when \( \gamma = 0.3 \) and \( \rho = 0.5 \) was 5 times larger than that when \( \gamma = 0.3 \) and \( \rho = 0.5 \). When \( \gamma = 0.2 \), the ratio of these values was approximately 2. For FMD GHOST, comparisons of the experimental results shown in the graphs reveal that the values when \( \rho = 0.5 \) were twice those when \( \gamma = 0.2 \) and \( \gamma = 0.3 \), with the gap between the two different values of \( \rho \) being approximately 6-fold.

6) COMPARISONS OF THE THEORETICAL RESULTS WITH THE EXPERIMENTAL RESULTS

Finally, we describe these results from the viewpoint of theoretical evaluations. The attack duration against the longest-chain rule was always 0 because \( \gamma \) was smaller than the minimum required value shown in Figure 3 and Table 3. Similarly, the values of GHOST and LMD/FMD GHOST were 0 when \( \rho \) was 0.6 and \( \gamma \) was smaller than 16.6, which is the minimum value depicted in Figure 4. Clearly, the lines of GHOST depicting these experimental results in all figures agree with those of the theoretical evaluations.

V. ADAPTABILITY TO ETHEREUM 2.0

We posted our preliminary results regarding the saving attack in an Ethereum online forum [23] even though the results were based on an approximate simulation, which did not reproduce a P2P network. After discussion, Ethereum 2.0 proposed FMD GHOST and employed it. Our results thus far have shown the robustness of FMD GHOST against the saving attack. Therefore, the saving attack is prevented in Ethereum 2.0 because FMD GHOST is adopted.

Here, we discuss whether the results that we obtained are adaptable to Ethereum 2.0. First, we describe the differences between the models we used in our research and the actual situation of Ethereum 2.0 and then discuss the adaptability of our results to Ethereum 2.0. When appropriate, please refer to Section III, where we describe our models.

First, we consider the differences in the network layer. Ethereum 2.0 will use GossipSub [24] as its communication protocol. This protocol carries out communication by an “eager push”, which enables nodes to send blocks directly without Inv and Req messages, in addition to the three-step communication we reproduced in our simulation. Moreover, validators select their neighborhood nodes based on a score. Regarding the consensus layer, attestations exist in Ethereum 2.0, as mentioned in Section II. In our simulation, we did not reproduce the attestations; hence, the number of nodes and the number of slots in an epoch were different from those of Ethereum 2.0, which made our study more versatile and adaptable to general slot-based PoS blockchains.

Nonetheless, our results are adaptable to the actual Ethereum 2.0 because of the following. First, the communication protocol differences merely speed up the network and make it much more likely that the P2P network is synchronous and messages completely propagate within one slot. The important aspect is whether messages are disseminated to the whole network within one slot. Thus, these differences do not affect our results much because we simulated a normal network condition that maintained synchrony after the GST. In other words, the message propagation speed does not affect the consensus as long as all nodes receive messages within a slot. For the same reason, the difference in the number of nodes is not influential.

Then, we consider the consensus layer. In a real network, messages, i.e., blocks and attestations, propagate to the whole network within a slot under normal network conditions. In this situation, nodes can receive a block and attestations in a slot, and we can equate a combination of the block and attestations to a block in the context of a slot-based block consensus. In other words, the following are satisfied.
All validators have exactly one proposal right per epoch.
- Each slot in an epoch possesses the same number of rights.

Thus, the consensus with attestations is equivalent to the slot-based consensus we simulated, as long as we simulate the normal network conditions.

VI. RELATED WORK

In this section, we compare our study to related work. At least three categories related to our work exist: studies on blockchain security by simulation, studies on fork-choice rules and research on the Ethereum 2.0 consensus.

First, we describe the simulation work. Stoykov et al. [25] proposed VIBES, a configurable blockchain simulator. They simulated a double-spend attack and investigated the effects on the orphan block rate. The study by Gervais et al. [7] is another example of an investigation on blockchain security by simulation. They simulated a PoW blockchain and investigated the impact of eclipse attacks on selfish mining. While these studies addressed a PoW consensus, we simulated a slot-based PoS consensus and investigated the blockchain security.

Bitcoin, the first blockchain, introduced the longest-chain rule, which was the first fork-choice rule. A number of studies have proposed attacks under the consensus with the longest-chain rule, e.g., the double-spend attack [1] and selfish mining [26], [27]. To alleviate the effects of adversarial strategies, ” was proposed [2]. Natoli and Gramoli [8] introduced the “balance attack”, which exploits the traits of GHOST. In addition to these two rules, the Ethereum community introduced two new fork-choice rules: LMD GHOST and FMD GHOST [4]. Because these two new rules have not been investigated sufficiently, an investigation was necessary.
This study proposed a new kind of attack and examined the influences on the blockchain consensus of these fork-choice rules.

In terms of research on the Ethereum 2.0 consensus, the study by Neu et al. [9] is relevant. Although this study simulated Gasper with LMD GHOST, it did not reproduce a P2P network. Not only this study but also many security studies tend to have strong assumptions about the adversary’s capabilities, i.e., the adversary can control message dissemination to honest validators. We introduced the saving attack, which is possible without the adversary being able to control message propagation at the P2P layer. We investigated the effects of the saving attack on the slot-based PoS blockchain consensus by simulation, reproducing a P2P network with real-world network latency and bandwidth.

**VII. CONCLUSION AND FUTURE WORK**

In this study, we introduced saving attacks and investigated their effects on blockchain consensus. These attacks lead to poor performance and high latency to block finalization. We found that the saving attack has a very large negative impact on the consensus. In our simulation, without the adversary having special capabilities at the network layer, the adversary succeeded in preventing consensus against LMD GHOST for a long time, e.g., 13 epochs. However, FMD GHOST decreased the attack duration to approximately 1.1 epochs under the same conditions.

Here, some possibilities of future work are worth noting. We investigated the risk of the saving attack under a slot-based PoS consensus protocol. However, attestations exist in Ethereum 2.0, and they might make an adversary’s strategies more flexible and disperse. Thus, saving strategies with attestations should be covered. Although we investigated some cases of the attackable state set $\Omega$ defined in Section III, other possible states are included in $\Omega$. Therefore, a follow-up work should investigate the other states in $\Omega$. One example is situations where more than two conflicting chains exist and/or the network is slower than the normal state are conceivable. Moreover, theoretical evaluations of the attack duration of LMD/FMD GHOST are expected to support our experimental results.

**REFERENCES**

[1] S. Nakamoto. **Bitcoin: A Peer-to-Peer Electronic Cash System.** [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Y. Sompolinsky and A. Zohar, “Secure high-rate transaction processing in bitcoin,” in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2015, pp. 507–527.

[3] **Ethereum 2.0 Specifications.** Accessed: Jan. 23, 2021. [Online]. Available: https://github.com/ethereum/eth2.0-specs

[4] V. Buterin, D. Hernandez, T. Kampefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X Zhang, “Combining GHOST and casper,” 2020, arXiv:2003.03052. [Online]. Available: http://arxiv.org/abs/2003.03052

[5] **Release Decoy and Bounce—Ethereum/eth2.0-Specs.** Accessed: Jul. 6, 2021. [Online]. Available: https://github.com/ethereum/eth2.0-specs/releases/tag/v0.9.1

[6] K. Otsuki, R. Banno, and K. Shudo, “Quantitatively analyzing relay networks in Bitcoin,” in *Proc. IEEE Int. Conf. Blockchain*, Nov. 2020, pp. 214–220.

[7] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in *Proc. ACM CCS*, 2016, pp. 3–16.

[8] C. Natoli and V. Gramoli, “The balance attack or why forkable blockchains are ill-suited for consortium,” in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2017, pp. 579–590.

[9] J. Neu, E. Nusret Tas, and D. Tse, “Ebb-and-flow protocols: A resolution of the availability-finality dilemma,” 2020, arXiv:2009.04987. [Online]. Available: http://arxiv.org/abs/2009.04987

[10] F. Saleh, “Blockchain without waste: Proof-of-stake,” 2020.

[11] Y. Aoki and K. Shudo, “Proximity neighbor selection in blockchain networks,” in *Proc. IEEE Int. Conf. Blockchain*, Jul. 2019, pp. 52–58.

[12] R. Nagayama, R. Banno, and K. Shudo, “Identifying impacts of protocol and internet development on the bitcoin network,” in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6.

[13] V. Buterin and V. Griffith, “Casper the friendly finality gadget,” 2017, arXiv:1710.09437. [Online]. Available: http://arxiv.org/abs/1710.09437

[14] M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.

[15] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and L. Abraham, “HotStuff: BFT consensus with linearity and responsiveness,” in *Proc. ACM Symp. Princ. Distrib. Comput.*, Jul. 2019, pp. 347–356.

[16] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, “SimBlock: A blockchain network simulator,” in *Proc. CryBlock*, Apr. 2019, pp. 325–329.

[17] **Testnet Goerli (GTH) Blockchain Explorer.** Accessed: Jan. 23, 2021. [Online]. Available: https://goerli.etherscan.io

[18] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. *Discovering Bitcoin’s Public Topology and Influential Nodes.* [Online]. Available: https://www.cs.umd.edu/projects/coinscope/coinscope.pdf

[19] **Falcon—A Fast Bitcoin Backbone.** Accessed: Jan. 23, 2021. [Online]. Available: https://www.falcon-net.org/

[20] S. Henningsen, D. Teunis, M. Florian, and B. Scheuermann, “Eclipsing ethereum peers with false friends,” in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Jun. 2019, pp. 300–309.

[21] Y. Marcus, E. Heilman, and S. Goldberg, “Low-resource eclipse attacks on Ethereum’s peer-to-peer network,” in *Proc. IACR Cryptol.*, 2018, p. 236.

[22] **Prysmatic Lab—Medium.** Accessed: Jan. 23, 2021. [Online]. Available: https://medium.com/prysmatic-labs/eth2-medalla-testnet-incident-f7b3cc934a

[23] **Ethereum Research.** Accessed: Jan. 23, 2021. [Online]. Available: https://ethresear.ch/saving-strategy-and-fmd-ghost/6226

[24] D. Vyzovitis, Y. Napora, D. McCormick, D. Dias, and Y. Pbras, “GossipSub: Attack-resilient message propagation in the filecoin and ETH2.0 networks,” 2020, arXiv:2007.02754. [Online]. Available: http://arxiv.org/abs/2007.02754

[25] L. Stoykov, K. Zhang, and H.-A. Jacobsen, “VIBES: Fast blockchain simulations for large-scale peer-to-peer networks,” in *Proc. 18th ACM/IFIP/USENIX Middleware Conference: Posters Demos*, Dec. 2017, pp. 19–20.

[26] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” in *Int. Conf. Financial Cryptogr. Data Secur.* Springer, 2014, pp. 436–454.

[27] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, “Formal barriers to longest-chain proof-of-stake protocols,” in *Proc. ACM Conf. Econ. Comput.*, 2019, pp. 459–473.
**RYUYA NAKAMURA** has worked on the security of real-world blockchain consensus protocols, discovered several vulnerabilities in the Ethereum 2.0 protocol, and contributed to the development of the specification. He was on the first Japan-based team to win a grant from the Ethereum Foundation. He was selected for the IPA Exploratory IT Human Resource Development Project, in 2020. He is currently an Executive Officer at LayerX Inc. His research interests include the development of technologies to solve the security and privacy issues of data in finance, government, and internet voting.

**KAZUYUKI SHUDO** (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer science from Waseda University, in 1996, 1998, and 2001, respectively. He worked as a Research Associate with Waseda University, from 1998 to 2001. Later, he worked as a Research Scientist with the National Institute of Advanced Industrial Science and Technology. In 2006, he joined Utagoe Inc., as the Director, i.e., Chief Technology Officer. Since December 2008, he has been worked as an Associate Professor with Tokyo Institute of Technology. His research interests include distributed computing, programming language systems, and information security.

He received the best paper award at SACSIS, in 2006, the Information Processing Society Japan (IPSJ) best paper award, in 2006, Super Creator certification from the Japanese Ministry of Economy, Trade and Industry (METI) and Information Technology Promotion Agency (IPA), in 2007, the IPSJ Yamashita SIG Research Award, in 2008, Funai Prize for Science, in 2010, the Young Scientists’ Prize, Commendation for Science and Technology by the Minister of Education, Culture, Sports, and Technology, in 2012, and the IPSJ Nagao Special Researcher Award, in 2013. He is a member of the IEEE Computer Society, IEEE Communications Society, and ACM.

***