TL;DR

We discuss why saving strategy

is problematic in LMD GHOST compared to other fork-choice rules and introduce FMD GHOST as the mitigation.

Prerequisites:

Decoy-flip-flop attack on LMD GHOST [

Casper

](/c/casper)

TL;DR We present an attack on LMD GHOST called “decoy-flip-flop” attack, by which an adversary can delay the finalization for a few hours ~ days by leveraging a network failure. This attack does not break the basic security of ETH2.0 but implies some manipulability of LMD GHOST. (Prerequisites: Vitalik’s blog post on LMD GHOST) Setup: LMD toy consensus To make the analysis simple, we discuss a binary consensus where honest validators try to make a consensus between two colors (RED and BLUE). …

Saving strategy

One of the properties of a fork-choice rule we want is that any player cannot increase their “power” on the consensus against their anti-Sybil asset (e.g. stake in PoS, amount of computation in PoW).

(Related work: Vitalik’s post on other fork-choice desiderata)

In our previous analysis on LMD GHOST, we discussed the saving strategy

, where the attacker “save up” blocks/votes for a while and spend these savings to keep the balance between two chains, delaying the finality.

Here, we look at how such a strategy works in various fork-choice rules.

Longest-chain rule

The traditional longest-chain rule is resistant to saving strategy because the protocol can prevent blocks/votes to extend a future block. In PoW, a block can extend only the block which has been published before mining. In PoS with a slot-based approach, we can invalidate blocks/votes which vote for a block of a future slot.

Although the attacker can potentially make a single reorg by savings after an extreme network failure (e.g. where there is a fork whose head is older than the saved blocks/votes), he cannot continue to grow the two chain.

GHOST

There is a fundamental difference between longest-chain rule and GHOST about saving strategy; in GHOST, a vote for any height can affect the current fork-choice and hence any vote will never “lose its power”.

This figure shows that the attacker can start to keep the balance of two chains using the saved blocks, starting from the same state as the previous example.

[

image

894×546 33.4 KB

](https://ethresear.ch/uploads/default/original/2X/9/9cecc7ed2a6025af0b63f421d0db0842ecfa7d2d.png)

For an attacker to start balancing attack by saved votes, there must be two conflicting chains forking from the slot older than the slot from which he started savings. Also, the attacker must spend some of the savings to make these chains tied for balancing attack.

Therefore, the attacker cannot start balancing with a small network where sufficient blocks are accumulated in either of the two chains like this.

[

image

496×545 16.7 KB

](https://ethresear.ch/uploads/default/original/2X/0/09c666a0dca6263475bd8fc1d531c35b6acee1a9.png)

To keep balancing, the attacker must continue spending his savings as many as the number of new honest blocks.

If the attacker controls x

(x < 1/2

) of the total validator set by ratio, he must spend 1 - x

epoch worth of his savings and gains x

epoch worth of new savings every epoch. Therefore, 1 - x - x = 1 - 2x

epoch worth of savings decreases every epoch.

Sidenote: PoW VS PoS

Saving strategy in slot-based approach (basically PoS) is slightly more flexible than in mining (basically PoW).

In a slot-based approach, an attacker can skip his slot and create a block later so he can choose the chain to extend later.

In mining, an attacker must continue mining in saving so it is more withholding rather than skipping.

That's being said, in either case, an attacker will end up voting for two chains almost equally so in mining the optimal saving strategy is to mine on two chains equivalently. Hence the difference between PoS and PoW about saving strategy would not that big.

LMD GHOST

Saving strategy also works for LMD GHOST. Here we focus on the differences with GHOST.

The main difference is that the score is not accumulated by nature and the attacker can do better at balancing by decoy-flip-flop strategy.

As described in the previous post, the amount of savings the attacker must spend per epoch decreases exponentially with the number of adversarial validators. >37% attacker spends less than 0.01 epoch worth of votes on average per epoch

On the other hand, as described in the above, the amount of savings the attacker must spend per epoch decreases linearly in GHOST. For comparison, 37% attacker spends 0.26 epoch worth of savings.

An advantage of LMD GHOST over GHOST is that a fork can converge quicker (if it is not attacked); in LMD GHOST, honest votes more than the majority of the total is sufficient for convergence.

Also, unlike the normal GHOST, the attacker must keep being silent until he starts attacking (i.e. should not publish his votes newer than the saved votes) because only the latest message is counted.

(Analysis on LMD GHOST in CBC Casper is interesting but out of the scope of this post.)

FMD GHOST

Fresh-message-driven (FMD) GHOST is a variant of LMD GHOST which only counts votes from the current epoch and the previous epoch.

With this modification, an attacker cannot save up the votes older than the previous epoch. Saving is possible at most 1 epoch worth of votes.

Q. Why "previous and current" epoch?

The rationale of the definition of freshness

is that we want to consider opinions from all the validators. Otherwise, an attacker can temporally control a large portion of the votes considered in the fork-choice.

In a protocol like ETH2.0 where one vote is allocated to each validator every epoch, to consider the votes from previous and current epoch is the most simple approach for that goal.

Q. Does this introduce a new synchrony assumption on Layer1 security?

No.

First of all, a fork-choice rule is a function whose input is a view

(i.e. blocks/votes which a validator have received) and output is the head block(s). What differs FMD GHOST from the others is the input of FMD GHOST includes the current time (slot/epoch) along with a view. In other words, the interpretation of the view depends on when it is.

I would call a family of these fork-choice rules synchronous fork-choice rule

.

This is different from the synchrony assumption of consensus protocol i.e. the assumption on the message delay under which the protocol satisfied safety and liveness. Any fork-choice rule requires some assumption on message delay for convergence.

Implications for ETH2.0

Emergency fix on the validity condition of attestation

For now, ETH2.0 allows attestations to vote for a future block. (GitHub issue)

This makes saving strategy possible even in a fully synchronous network and hence the decoy-flip-flop attack becomes much easier.

[

image

946×501 22.8 KB

](https://ethresear.ch/uploads/default/original/2X/4/4fd6a616135d81dcd0fd8a0a27968f39dd5ce7ff.png)

(In this figure, all the votes are blocks and there is no attestation for simplicity.)

Therefore, I think this should be fixed in as soon as possible.

Why LMD GHOST, in the first place?

FMD GHOST is a potential alternative to restrict saving strategy but it is still a family of LMD GHOST and decoy-flip-flop with 1 epoch worth of savings is possible. (Again, >37% attacker can make liveness failure for >100 epochs with 1 epoch of savings).

I think this raises a question like why LMD GHOST is chosen for ETH2.0 and what we are optimizing for (e.g. security, performance, implementation-simplicity) with that choice. I do not know any formal analysis on this and it would be interesting future work.