Removing Trusted Relays in MEV-Boost Using Threshold Encryption

TheMerge™ is approaching quickly and with it mev-geth (the core of almost all of today's MEV extraction infrastructure) will become dysfunctional. The successor most likely to be adopted by the validator community is the Flashbots' championed mev-boost. Unfortunately, its current design relies on trusted and therefore presumably fairly centralized relays who have far-reaching censorship powers – a very controversial property and topic of many recent stormy discussions in our community.

In this post, we propose a modification to the design of mev-boost that replaces relays with large fault-tolerant threshold committees. We discuss tradeoffs, potential attacks, and corresponding counter measures. We hope that our proposal is a helpful contribution to mev-boost as well as future in-protocol PBS designs.

TL;DR

- Builders encrypt block bodies using an threshold encryption scheme and send them to proposer
- Proposer picks one and signs its header
- Threshold committee provides decryption key for body, so that proposer can decrypt and broadcast the full block
- If the builder builds an invalid block (undetectable for the proposer due to the encrypted body), they are slashed retroactively using an optimistic rollup-style fault proof mechanism.

Block Production Protocols

In order to motivate our proposal, we start with a description of (our understanding of) plain mev-boost with relays as proposed by Flashbots. We then iterate on this protocol by adding a threshold committee that reduces trust assumptions in the relay significantly. Finally, we realize that the relay can be removed completely.

Block Production With MEV-Boost

Users -----> Builders -----> Relays -----> Proposers -----> Network

MEV-Boost introduces two classes of nodes involved in block production: Builders and relays. The block production process is as follows:

1. Users send transactions to builders.
2. Builders build blocks for the proposer (without signing them) and send them with a bid to a relay.
3. The relay validates all blocks they receive and on request passes the header of the valid block with the highest bid to the proposer.
4. The proposer signs the header and returns it to the relay.
5. The relay returns the corresponding block body to the proposer who broadcasts it.

The relay protects builders from proposers and vice versa. If they had to communicate directly:

- Proposers could steal MEV from builders, i.e., copy their blocks, but redirect the MEV they found to themselves (more complicated strategies are possible to increase profits and deniability)
- If builders would send only block headers to proposers in order to prevent above attack, builders could grief proposers by either sending invalid blocks or not publishing block bodies which would result in the network rejecting the proposals.
- Simple implementations might rely on direct communication channels between builders and proposers, opening proposers up for network level DoS attacks.

However, even though they are permissionless, relays are trusted and therefore presumably fairly centralized. This is problematic as they are in a powerful position. Concretely, they may decide to

1. stop relaying
2. censor some builders

3. relay headers, but withhold block bodies
4. steal MEV
5. censor blocks according to arbitrary criteria, e.g., inclusion of transactions from or to blacklisted accounts

MEV-Boost With Blinded Relays

|----------> Keypers <---------| | | | | Users -----> Builders -----> Relays -----> Proposers -----> Network

In order to mitigate attacks 3 to 5, we can “blind” the relay using identity-based threshold encryption. In short, in identity-based encryption, messages are encrypted with a master public key for a particular “identity”, represented by an arbitrary string. For each identity there is a corresponding decryption key. Combined with threshold encryption, a committee of “keypers” is collectively responsible to generate first the master public key and later the decryption key for any identity that requests it.

The block production process is as follows:

1. The keyper committee generates the master key using a distributed key generation protocol and broadcasts it.
2. Users send plaintext transactions to builders.
3. Builders build blocks and encrypt the body using the master key with the block hash acting as identity. They send the encrypted body, the unsigned plaintext header, and a bid amount to the relay.
4. On request, the relay passes the unsigned header and encrypted body with the highest bid amount to the proposer.
5. The proposer signs the header and sends the signed header to the keyper committee.
6. The keypers broadcast their shares of the decryption key corresponding to the block hash.
7. The proposer aggregates enough shares to compute the decryption key. They then decrypt, validate, and broadcast the block.

As the relay does not have access to the plaintext body they cannot steal MEV or censor blocks according to some criteria. As proposers sign block headers only when they already have the (encrypted) body, the relay cannot withhold the body. Relays are still able to censor builders or refuse service altogether.

Encryption of bodies unfortunately prevents both relays and proposers to validate the blocks before the proposer has to sign the header. Thus, builders could trick proposers into proposing invalid blocks, resulting in reduced rewards for the proposer and (in the worst case) preventing the chain to make progress. There are two possible counter measures:

1. Pessimistic: Require builders to attach a zero-knowledge proof that the block is valid.
2. Optimistic: Require builders to be staked and slash them retroactively if they build an invalid block. The slashed amount is paid in full to the proposer.

Due to the complexity of the validity condition, the pessimistic option is likely impractical in the short term. In contrast, proving invalidity is feasible today as demonstrated by live and in production optimistic rollups. As long as proposers only accept blocks from builders who have staked more than the bid they promise to pay, they can be assured that they will never earn less than the promised bid.

MEV-Boost Without Relays

|--> Keypers <--| | | | | Users -----> Builders -----> Proposers -----> Network

In above proposal, the relay acts as a mere network hub facilitating communication between builders and proposers, with the only additional purpose of aggregating and filtering blocks. It can thus be removed and replaced by a gossip network on which any builder can publish blocks and proposers listen passively. As there would be no relay left, all of above mentioned attacks are prevented.

In order to protect against spam, nodes can filter blocks by builders who have not deposited a high enough stake or already sent too many proposals for the current slot. Even more so, blocks with bids not higher than earlier bids for the same slot can be filtered. This should reduce load on the proposer significantly who now cannot rely on the relay to pick a block proposal for them. Still, bandwidth requirements on proposers will likely increase by a factor on the order of the number of builders in the network.

The disadvantage over the previous proposal is that communicating via a central relay is faster, more efficient, and (depending on your trust assumptions) more reliable than through a gossip network. Thus, in this proposal, less MEV might be extractable (as builders have less time to optimize) and the risk of empty slots is increased (in case a block is delayed too much or even lost in the p2p network).

Keyper Committee

Threshold encryption requires a committee of nodes that produce keys. The system is fault-tolerant against failure of up to n - threshold

nodes, an arbitrarily chosen parameter. The bigger the committee, the higher the security, but the worse the performance and the bigger the risk of missed slots: A higher threshold means more nodes need to cooperate to produce a key, meaning less are allowed to be offline. We think a committee size of 50 to 200 nodes is reasonable, but even small committees of 5 to 10 nodes would already provide a benefit over plain mev-boost.

There are many options how the keyper committee can be formed. Ultimately, it can freely be chosen by builders and proposers and a situation in which multiple committees compete with each other is imaginable. The most straightforward way would be to start with committees managed by reputable entities in the ecosystem (Ethereum Foundation, Flashbots, BloxRoute, ZeroMEV, etc). Long term, DAOs might take over this responsibility.

Ideally, the committee is compensated. Since it provides a service primarily to builders (protecting them from MEV-stealing proposers), builders may set up a fund ensuring their existence.

Attacks and Counter Measures

In this section, we discuss two potential attacks by the threshold committee as well as a slashing mechanism to discourage one of them.

Attacks

The threshold encryption committee may attack in two ways:

1. Liveness attack: Do not generate the decryption key of a header (or do so late). This requires collusion of at least 1 - threshold

(e.g. 1/3

of the keyper set).

1. MEV-stealing attack: Generate the decryption key of a block that is not supposed to be decrypted. This requires collusion of at least threshold

(e.g. 2/3

of the keyper set).

The liveness attack leads to an empty slot and thus reduces the performance of the network. It can also be considered a griefing attack as neither proposer nor builder will be paid in this situation. However, the attack is not very concerning to us as there is no direct incentive rewarding the attackers and proposers themselves are already able to achieve the same for short periods of time. The only increased risk is that of long ranges of empty blocks, but this would be observable in the chain's history, so proposers can employ a simple circuit breaker style counter measure and fall back to local block building.

The second type of attack can be used for MEV stealing: Before a header is signed, the keypers decrypt the most promising block and create a copy of it that redirects all MEV to themselves and pays a marginally higher fee to the proposer. They send it to the proposer who will accept it and propose it to the network. More involved strategies are possible: The attackers can peak into the blocks of multiple builders and (block space permitting) extract the aggregated amount of MEV found in all of them. They might also introduce custom non-MEV-maximizing preferences such as targeted censorship. This attack would marginally benefit the proposers, but greatly harm builders who would not be compensated for their service anymore. The attack is incentivized by the extractable MEV minus the proposer fee.

Both attacks are difficult to carry out as they require cooperation in secret across a large set of nodes. If an attack is detected or suspected, proposers and builders can either replace the keyper committee or fall back to local block production. In addition, as described in the next section, the MEV-stealing attack can be punished using a slashing mechanism.

Slashing

A slashing contract enables punishment of the keypers attempting the MEV-stealing attack described above. The general idea is to slash keypers for releasing their decryption key share unprompted, i.e., if there is no corresponding signed block header.

Mechanism

The slashing contract stores the keyper committee as well as the master key they produce. Each keyper is expected to stake a certain amount of funds in the contract (alternatively, EigenLayer can be used). Before using their service for a slot, builders and proposers should verify that all participants of the keyper committee are staked and that the contract knows the encryption key.

Anyone can initiate a slashing event by providing the decryption key share of a keyper as well as the corresponding block hash. The contract verifies that the share is correct relative to the block hash as well as the master key and that it belongs to the accused keyper. Subsequently, a challenge period begins in which the accused keyper can exonerate themselves by providing the proposer-signed block header for the given hash. If they fail to exonerate themselves, a fraction of the stake is paid to the accuser and the remainder is burned.

In order to prevent griefing, accusers have to submit a small safety deposit which will be paid back if the challenge is successful. If the challenge is not successful, the deposit will only be paid back if the accuser can provide proof of equivocation of the block producer, i.e., a second block header for the same slot as the one the accused exonerated themselves with.

Examples

To illustrate the slashing mechanism, let us consider three cases:

1. Honest keyper, honest proposer

2. Dishonest keyper, honest proposer

3. Dishonest keyper, dishonest proposer

An honest keyper only releases their decryption key share upon receiving the corresponding block header. Thus, in case 1, the keyper will always be able to exonerate themselves.

A dishonest keyper attempting an MEV-stealing attack would release decryption key shares for blocks that the proposer will not sign (after all, the goal is to produce an alternative block). Thus, in case 2, anyone who gets ahold of the share can accuse them and the accused will not be able to exonerate themselves.

Arguably, another type of dishonest behavior is creating the decryption key early, even if the corresponding block is later proposed. While this in itself does not harm the network, keypers might use this strategy to evaluate if an attack on the block might be profitable even if taking their slashed deposits into account. However, for reasonable deposit sizes and in a competitive builder market these blocks are highly unlikely. Furthermore, the keypers risk being slashed if the proposer decides to sign a different block.

In case 3, the proposer can collude with the keypers. This gives them the ability to exonerate an accused keyper by signing the block for which they have been accused, even though another block has already been signed and likely been broadcast to the network. While the keyper will go free in this case, the proposer will be slashed according to the rules of Ethereum's consensus layer — likely a bigger punishment for the attackers. The accuser will still get their deposit back as they can present proof of the proposer's equivocation.

Data Availability

There is one caveat to the slashing mechanism: It requires a single honest party to have access to the decryption key shares. However, a perfectly coordinated attacker set would not leak the shares and, thus, even though the slashing condition would be fulfilled, no honest party would be able to submit the evidence to the contract. Fortunately, cooperation in large groups at this level of perfection is very difficult to achieve, especially if defecting is highly encouraged (the stake of all attackers would be paid out to the "rat").

Effects on Latency

Proposers want to make sure they can propose signed and decrypted blocks in time so that they are likely to be accepted by the network. Since generation of the decryption key is an additional step that takes a certain amount of time, they have to stop accepting bids earlier in the slot. Builders therefore have to submit their bids earlier as well, giving them less time to optimize routing (which can only start once the previous block is decrypted). Note though that the number of transactions they can consider does not change, so the total amount of available MEV does not change.

For the same reason, average latency experienced by users increases.

The extent of the latency increase is determined by the performance of the p2p network: It takes one message (the block header) from the proposer to threshold

keypers and one message each from threshold keypers back to the proposer. If the network is well connected, we hope the latency increase is acceptable.