MEV mitigation protocol based on SSS

Nowadays searchers do not have any guarantee that whenever they find MEV opportunities those won't be simulated and front-ran by the block producer.

Let's assume that the chain on which the MEV opportunity was found consists of a protcol built on top of it, which is able to operate on a shared ECDSA key using techniques (e.g. SSS).

The participants of such a protocol would be:

- Block producers
- Block producers verifiers
- Users/Searchers

Such a key could be generated and split accross the protocol where the number of it's block producers and cerifiers is known.

Whenever a searcher or a user wants to submit a transaction, it will have the ability to encrypt its content with the shared and known public key.

For the sake of simplification let's introduce something I call Banana Transaction

or BTx

.

The BTx

could have the structure below:

{ encryptedContent: { recipient, data, gasLimit, value,
}, gasPrice, encryptedContentHash, signature }

BTxns

might exist only in the mempool, those are not part of the chain.

The lifecycle of a block including BTxns

can be the following:

1. EOA submits a BTx

to a dedicated mempool.

1. The BTx

spreads accross the protocol through the p2p layer.

1. Block producer will pick up a BTx

from the mempool and form a block or append to an existing one.

NOTE: The transactions will be ordered by its gasPrice

1. Once the block is formed it will be proposed to the protocol participants.
2. The protocol network in response will agree on the block and decryptBTxns

.

1. The original block producer will include the block into the chain preserving the proposed order (sorted by gasPrice, if there are 2 or more BTxns

with equal gasPrice the BTx

with the bigger hash will be included on the top) and exclude invalid BTxns

(if any).

1. Whenever the protocol participants will observe that the included block did not preserve the proposed order the block producer will be punished (e.g. slashing).

This design could be widely applied to most of the chains because it does not require any hard fork.

The network/MEV mitigation protocol

can be setup by picking up the actors below:

- Block producer which will stake some tokens in order to participate in the protocol.
- Verifiers who ensure that the proposed order was preserved.
- Searchers/Users willing to submit BTxns

.

The value of posted stake by the block producer should be high enough in a way, that the loss because of slahing is bigger than the potential outcome from including another block than the proposed one.

Also the proposal assumes that Searchers/Users

are willing to pay higher gasPrice

than usual, in order to incentivise block producers to participate in the network.

Such a solution prevents:

1. Block producers from frontrunning MEV searchers.
2. Sandwiching bots from sandwiching users transactions.
3. Any sort of MEV done on top of the mempool(BTxns

mempool) content.

Weak points and solutions:

- DDOS - Sending BTxns

with invalid content within the encrypted section * Requires the BTx

origin to submit a valid hash cash solution on time of submission.

- Requires the BTx

origin to submit a valid hash cash solution on time of submission.

- How can a distributed protocol detect that the block producer changed the order?
- The participants can vote that the block producer cheated by reaching a majority ← This is weak cause the majority can be unfair and steal the stake deposited by the producer by enforcincg slashing?
- Include fraud proofs where the rlp encoded decrypted txns are sent to a smart contract, the contract will order them and recompute the block hash, check if the recomputed blockhash matches the hash of block producer by the block producer. If not the fraud is proven.
- The participants can vote that the block producer cheated by reaching a majority ← This is weak cause the majority can be unfair and steal the stake deposited by the producer by enforcincg slashing?
- Include fraud proofs where the rlp encoded decrypted txns are sent to a smart contract, the contract will order them and recompute the block hash, check if the recomputed blockhash matches the hash of block producer by the block producer. If not the fraud is proven.