This post describes a possible alternative paradigm to two-slot PBS for designing proposer/builder separation mechanisms. The basic philosophy of this approach is simple: take MEV Boost and replace the relayer with a size-256 committee of validators.

- A builder, instead of sending their payload to a relayer, now erasure-codes their payload into 256 chunks (85 needed to recover), encrypts the i'th chunk to the public key of the i'th committee member, and sends this into a p2p subnet.
- Each committee member decrypts their message, and validates the proof. If the proof is valid, they make a pre-attestation to that payload's header.
- The proposer accepts the highest-bid header for which they found at least 170 signatures
- Upon seeing the proposal, the committee reveals the chunks, and the network reconstructs missing chunks if needed
- Attesters vote on the proposal only if the proposal AND the payload are available

Simplified diagram of MEV boost

Simplified diagram of this proposal

Interaction with sharding

The above diagram assumed a non-sharded chain. There are two natural ways to make this scheme work with the 32 MB payloads of sharding:

1. Require the builder to evaluate the degree-256 polynomial of EC points of the blobs at 768 other points (to preserve zero knowledge), and encrypt the full blob corresponding to each evaluation (or set of 3 blobs) to the corresponding committee member.
2. Apply the non-sharded scheme above only to the ExecutionPayload, and expect the committee to only care about the payload. Once the proposal is published, the committee would decrypt and reveal the ExecutionPayload and the attesters would data-availability-sample the corresponding blobs, expecting them to already be available.

(1) preserves pre-publication-zero-knowledge of the blobs, (2) does not [it only preserves pre-publication zero-knowledge of which blobs were chosen]. But (2) is much more efficient and lower-overhead than (1), and requires much less work from the builder.

DoS protection

When a builder makes a payload of size N, this imposes:

- O(1) load to the entire network
- O(N) load with a very small constant factor (the aggregate signature verification) to the entire network
- O(N) load spread across 256 nodes (so, roughly O(1) load per node if we assume there are as many builder proposals as there are validators)

The second can be managed by splitting committees into many subnets, and the third can be managed by making sure that there are not too many payloads (eg. allow each builder to propose a max 3 per block, and increase the min deposit to be a builder). As a practical example, if there are 100000 total validators, and there are 1000 size-1-MB payloads, then the per-validator downloaded data would only be 10 kB.

Advantages of this proposal vs two-slot PBS

- Single slot, requires less fork choice complexity
- In the same "format" as MEV Boost, requires much less transition complexity for an ecosystem already running on MEV Boost
- Avoids increasing block times
- Possibly higher security margins (1/3 instead of 1/4)?
- No block/slot required
- Easier backoff (just only allow proposals >= k slots after previous proposer to give time to decrypt and attest)

Disadvantanges of this proposal vs two-slot PBS

- Less convenient for SGX-based searcher/builder architecture, because there is no pre-builder-reveal attester committee that could be used to trigger decryption (such an architecture would have to use its own proposal availability oracle)
- Depends on majority being online to decrypt (possible fix: choose committee from validators present in previous epoch)
- Requires more cryptographic primitives that have not yet been implemented (standardized asymmetric encryption)
- Variant (1) of integration with sharding makes it much harder to make a distributed builder (because of the need to encrypt full bodies of the extension)
- Variant (2) of integration with sharding does not seem to preserve pre-publication zero knowledge of bundles (though maybe pre-publication zero knowledge of _the choice of which bundles_is good enough)