Third of the Offslack Specs

.

What

Relays in the MEV Boost protocol serve two crucial roles during the slot:

1. Timeliness Committee* (n = 1)
2. Data Availability Committee* (n = 1 pessimistically, n > 1 optimistically)

*Liberal use of the word “Committee”

These two “committees” prevent against two specific attacks:

- Block withholding:

The builder submitted a block bid but failed to reveal the block body within the required timeframe (Data Availability Fault).

- Block unbundling:

The validator takes transactions from a builder’s submitted block and inserts it into the canonical chain via another block, potentially at risk of equivocation. (Perhaps akin to Safety Fault?)

Multiple attempts at expanding values of n in the permissionless setting have been met with concerns of increased latency via the introduction of synchronous communication or a failure to detect faults in the above two roles. Let’s explore how we could use TEEs to lessen the importance of the relay’s role.

How

Instead of accelerating even further down the “optimistic” path, Trusted Execution Environments (TEEs) could offer a safer guard rail to lean upon. Idea 1 talks about how we could verify proofs in MEV boost that a block was built in a TEE. Idea 2 is to embrace different trust models for different types of validator sets. And idea 3 is about how we could take Idea 1 a step further by putting the entire MEV Boost sidecar on a TEE and using it as an n = 1 timeliness and DA committee.

Idea 1: MEV boost Verifies TEE Proofs

Very simply, the validator turns on an experimental TEE proof flag and a boostrap service on their MEV boost side car. Every epoch the bootstrap service is queried to get a list of builder URIs, pubkeys, and attestations of the satisfaction of some TEE registration process such as in SUAVE Sirrah. Once bootstrapped, the MEV boost side car proceeds as normal by multiplexing to list of relays and builders when ever asked to.

[

Screenshot 2024-04-20 at 6.56.10 PM

1332×996 59 KB

](https://collective.flashbots.net/uploads/default/original/2X/1/102f87951d50850fa09b368f7d3e6c295d1d4790.png)

- Get Registry:

MEV-Boost requests a Bootstrap Service for Builder Pubkeys which have been created inside of an SGX as verified via attestation. This service could take various forms and is thus not elaborated on here.

- Get Header:

Same as normal, but query a Builder directly which implements the Relay API.

- Get Payload:

Same as normal, but query a Builder directly which implements the Relay API.

Ideally, the bootstrap service could also tell TEE builders which signatures to trust, like those only in large validator sets.

This idea, unfortunately, does not satisfy our concerns that the Builder could purposefully withhold the block after signing the header (DA fault) or incur sufficient latency so as to not propagate the block in time (timeliness fault). But we actually do not have a solution to this problem with relays either at the moment. Typically, these types of disputes are resolved over an exchange of logs in Telegram channels, and this idea does not solve that. In the presence of reputable actors it simply lowers some dependence on the relay and cuts out a middleman, plug builder Pubkeys in with caution.

Idea 2: Rainbow MEV Boost

Idea 1 in the list above doesn't do much to resolve our issues with fault detection, but what if we never have an efficient solution for this until something like an in-protocol attester committees? Given the permissionless setting is so hard, one idea is to reduce the capabilities of our adversary model by embracing different models of trust. For example, a newly registered solo staker can be trusted much less than a reputable validator pool. A TEE builder could make sure to only build for friends or large validator pools, and those with a higher risk profile could chose to service arbitrary validators.

Idea 3: MEV boost in TEE

Could we take Idea 1 further? What if we put the MEV Boost sidecar in a TEE and use it as it's own Timeliness and DA oracle? This would mean that TEE builders send entire blocks to the TEE MEV boost sidecar so that it knows what is available and on time, and releases to validator only after signed header. This adds more responsibility, but it mitigates our issue of knowing whether the builder payload was available and timely to mev boost. The TEE also provides basic guarantees against unbundling by still requiring the validator's signed header signature.

Discussion

One pro of these approaches is that direct connection to Builders allows for later block bid updates. A con is we lose the nice connective tissue properties of relays i.e. providing Builders access to more Validators and Validators access to more Builders. And lastly one improvement is that the bootstrap service in Idea 1 could post the builder pubkey registry onchain.

The core idea of this proposal is direct opposition of one of the initial goals of PBS which is to remove the advantage large staking providers have in block construction. It is my opinion that we have have received 80% of the benefit already with this direction as in solo stakers have a very competitive route of requesting blocks from centralized relays. By expanding to a heterogenous zone of trust model, we can benefit from the social reputation risk that large operators face in acting maliciously in the MEV boost protocol and both reduce centralization via lessening role of the relay and two potentially allow for even later updating of bids within the block.

Idea

Description

Trust Model

Problems Addressed

Pros

Cons

1

Validate Builder TEE proofs in MEV-Boost

Block Builder TEEs

None

Cuts out middleman (unsafely), lowers dependence on relay

Doesn't solve fault detection fully

2

Rainbow MEV-Boost

Trust in Validator Varies by Reputation

Manual Fault Attribution

Attack Space Reduction

Efficient fault detection still a challenge

3

MEV-Boost sidecar in TEE

Validator TEEs

Bid Withholding, Timeliness Fault, Unbundling

Maybe solves our problems?

Adds more responsibility to sidecar