Tl;dr

A plan to reduce, and eventually remove, the role of the relay in MEV-Boost.

[

TEE-Boost architecture

1864×1122 104 KB

](https://collective.flashbots.net/uploads/default/original/2X/3/3c2cb9c0080888d54459f19a8d913ff37e8ff6d4.jpeg)

Intro

MEV-Boost relies on centralized relays (run by third parties) to connect validators with a market of block builders. These third-party relays provide essential guarantees like validating blocks, preserving privacy, and providing data availability. But in the long term, it's suboptimal for Ethereum to rely on a few centralized actors to serve 95% of blocks. For this reason, replacing the relay has been a top goal for the PBS community for over 2 years.

We believe there is a new and promising path forward. We propose TEE-Boost:

a change to MEV-Boost which allows validators to directly accept TEE* proofs from block builders. With these proofs, TEE-Boost can provide better guarantees than existing relays, without a centralized actor in the loop. This is akin to enabling free optimistic mode for all TEE builders.

TEE-Boost is an early design and comes with several open research questions. In particular, the current instantiation of TEE-Boost is focused on removing the trust assumption in relays during the getHeader

phase of MEV-Boost (ie. leveraging TEEs for privacy and integrity). But further research is needed on registerValidator

(multiplexing) and submitBlindedBlock

(data availability). We invite the community to join us in this R&D.

To help guide that conversation, this post will focus on a few questions:

1. Why do we want to replace the MEV-Boost relay
2. What is the architecture and purpose of the MEV-Boost relay today
3. How can TEE-Boost replace the relay during getHeader()

while preserving its guarantees

1. What open questions remain

*TEE = Trusted Execution Environment, like Intel's SGX or TDX

Why replace the MEV-Boost relay

Reducing the role of the relay can improve decentralization and increase revenue for Ethereum stakeholders.

Decentralization: 90% of blocks on Ethereum are served by ~5 centralized relays today. Removing these actors will protect Ethereum from risks of censorship and rent seeking.

Revenue: Performant relays have an average e2e latency of5-20ms in optimistic mode today. Skipping this step could remove 10s of milliseconds from block submission and make it possible to build more efficient blocks. Concretely, reducing the delay between bundle submission and block inclusion can reduce execution risk for cross domain searching strategies like CEX-DEX arbitrage. Because risk is typically priced into searcher bids, reduced risk would likely correlate with higher payments to validators.

MEV-Boost design

The MEV-Boost relay is an essential piece of infrastructure that provides several guarantees to builders and validators.

1. Privacy — Outsourcing validation and data availability to relays prevents validators from manipulating the contents of blocks before they are signed. This protects block builders and orderflow providers from unbundling attacks and

privacy leaks.

2. Validation — Relays ensure that the payload and bid returned by getHeader

are valid. Concretely, relays will resimulate blocks to detect errors or require builders to stake collateral for optimistic bids. This helps prevent missed slots.

1. Data availability — Relays ensure that the block payload is released to validators in response to submitBlindedBlock

. This helps prevent missed slots.

1. Safe multiplexing: Relays enable block builders and validators to register just once (eg. via registerValidator

) to access a large number of counterparties. This makes it much easier for new entrants to join the market, because builders don't have to rely on validators to manually add their endpoints and vice versa. Routing requests through a trusted relay also helps protect validators and builders from spam and keeps their IPs hidden.

[

MEV-Boost architecture

1000×1350 65.5 KB

](https://collective.flashbots.net/uploads/default/original/2X/e/e981453a7c80bef6f1a8b93727d6455d3365694f.png)

Any substitute for the relay would need to provide similar guarantees.

TEE-Boost design

TEE-Boost is a change to MEV-Boost which allows validators to accept TEE proofs directly from block builders.

TEE proofs

Block builders participating in TEE-Boost would run exactly pinned combinations of their system, open source code, and configuration. This produces a unique hash which can be attested and whitelisted by TEE-Boost.

Concretely we propose splitting the proving process into two phases. First, builders would submit attestations at the start of each epoch which validators running TEE-Boost verify. Attestations can be verified via a cloud provider's attestation service or locally for dcap attestations. This ensures that builders are running software which is known to produce valid blocks.

Then, builders generate a much simpler proof to accompany each block they submit. For example, builders can sign their block bid with a key that is only known by the TEE. This key could be randomly generated inside the TEE and the public part included in the builder's attestation. This allows validators to easily confirm that blocks are valid without needing to validate full attestations for every submission, reducing work in the hot path and the cost of spam.

Improved getHeader

flow

The key benefit of these TEE proofs is that they allow us to remove the trust requirement in the relay from the getHeader

flow. Concretely:

1. Assume that builder attestations were previously verified at the start of the epoch
2. Validator calls getHeader

in MEV-Boost

1. MEV-Boost calls getHeader

directly on TEE builders*

1. Each builder generates a TEE proof p

which proves that its block payload and bid are valid and includes p

in its getHeader

response to MEV-Boost

1. MEV-Boost validates each p

and selects the valid header with the highest bid to return to the validator

*Or through an aggregation layer, depending on the validator's preferences. See "other desired guarantees: safe multiplexing" for more.

TEE-Boost guarantees

Using TEE proofs in the getHeader

flow provides two important guarantees.

Privacy

: Because the TEE proof ensures that the block is valid, the builder never needs to reveal the block payload during the getHeader

phase. This is a strict improvement over the privacy guarantees we have today (where centralized relays see the block payload at this phase).

Validation

: TEEs like SGX and TDX have strong integrity guarantees. The risk of a malicious relay submitting invalid blocks is likely higher than the risk of spoofing TEE proofs, which would require multiple actors with weakly correlated incentives (eg. the hardware provider and cloud service provider) to collude.

What happens if TEEs break?

If a TEE is compromised, TEE-Boost will still have better privacy and integrity guarantees than MEV-Boost does today.

There is a risk of leaking data through side channels from the TEE enclave to the TEE operator. Specifically, block builders operate the TEEs in TEE-Boost and could gain access to private data during a privacy leak. But private data is never shared with a relay in TEE-Boost, even if the TEE is compromised. This is an improvement over the status quo where block builders and relays both have full access to private transaction data.

There is also a risk of the hardware manufacturer and cloud provider colluding to break the integrity of a TEE. If this happens, the TEE could serve invalid blocks that appear to have valid TEE proofs. This could cause missed slots the same way that bugs in MEV-Boost relays do today. But while it's equally likely for TEE builders and MEV-Boost relays to have bugs, it's more difficult to compromise the integrity of a TEE because multiple different actors need to collude. This likely makes TEEs more secure in the adversarial setting.

Other desired guarantees

While this design is a step forward, it does not yet address all guarantees that relays provide today. In particular, TEE proofs on their own do not inherently ensure data availability

of block payloads or provide safe

multiplexing

between builders and validators.

There are a variety of ways we can extend TEE-Boost to provide these guarantees. The core improvement to the getHeader

flow is compatible with many different implementations of data availability and safe multiplexing. We will suggest a few likely paths forward but are interested for input on these designs.

Safe multiplexing

It must be easy and safe for new builders and validators to register to produce and receive blocks from TEE-Boost. Historically, relays have served as a convenient registry to connect new builders and validators and protect them from leaking IPs. How do we facilitate easy, safe counterparty discovery without relays?

One approach to address counterparty discovery is to run a lightweight service which aggregates a list of TEE builder endpoints. This service would allow aggregating block builders in a single place that can easily be referenced by validators. In the short term we could repurpose centralized relays to provide a lightweight aggregation function; in the medium term this could be implemented as an on-chain contract. Either of these designs would also need to be accompanied by a method to safeguard validators from leaking their IP.

Another approach which could help protect validators (and builders) from directly leaking IPs is to create a gossip layer that stores the best headers submitted by TEE builders. Validators could pull headers from this gossip layer instead of querying builders directly. This could potentially improve safety at the cost of higher latency to gossip bids. In practice there may be some proposers that are comfortable directly connecting to builders and others that prefer to go through an aggregation layer like this.

Data availability

In TEE-Boost, block payloads are not shared with a trusted third party. This means builders could withhold payloads after relays sign the header and cause missed slots. What is the best (ideally decentralized and private) alternative to ensure data availability of block payloads?

One approach is to require builders to post block payloads to an encrypted Data Availability (DA) layer and include a proof of availability in addition to the proof of validity p

. This would come at the expense of additional trust assumptions about that encrypted DA layer and added latency from posting and generating availability proofs.

Another option would be to use a TEE escrow which requests the block from the builder before the validator signs the header. The validator would only sign the block once it's in this escrow (which can be hosted by the validator or a third party). The builder would accept a TEE proof from the escrow instead of validator signature as a requirement to release the block.

Further research questions

A non-exhaustive list:

1. What are the latency impacts of different data availability solutions? How can we minimize the added latency that DA adds to the hot path?

Requiring a proof for DA could add an additional round of communication to the hot path of TEE-Boost. In the worst case, this could counteract the latency benefits of TEE-Boost. Does a local escrow at the validator improve this? What else should we consider?

1. Does TEE-Boost expose validators to increased risk of DoS from builders? How much risk is acceptable?

In TEE-Boost, validators still control when they call getHeader

but the MEV-Boost sidecar may need to process more bids (one per builder instead of one per relay). How significant is this issue? Are there simple tactics that validators could employ, or that we should build into MEV-Boost, to mitigate this?

1. Does TEE-Boost expose builders to increased risk of DoS from validators? How much risk is acceptable?

Block builders are typically more sophisticated and capable of handling spam because they process large volumes of bundles from searchers. That said, are there any novel attack vectors that this opens up? What mitigations should we consider?

1. What are the risks of exposing validator IPs? What's the best way to mitigate these risks?

When

TEEs are performant enough to use in practice today. Flashbots has been building blocks in TEEs (both SGX and TDX) on mainnet for over 1 year, and and benchmarks show that TDX can perform at near-native speed. It would be reasonably straightforward for these TEE builders to generate proofs of block validity and for MEV-Boost to verify them.

More broadly, TEEs would help decentralize the current implementation of MEV-Boost and ensure that future upgrades can be decentralized. They make it possible to add new functionality without adding new trust assumptions or intermediaries. Given the readiness of the technology, and the rising interest in adding new functionality to PBS actors, we believe that now is the time to focus on developing this mechanism.

We're looking for motivated community members to collaborate with us on open research and design questions. If you're interested in contributing, consider applying for a grant through the Flashbots Research Proposals program or the PBS Foundation.