Fair Exchange in Proposer-Builder Separation

A staker, acting as a proposer, in a proof-of-stake system, can earn revenue through four avenues: (i) as a block reward for proposing a block at a given slot, (ii) as a reward for identifying and reporting past malicious actions by other stakers, (iii) through fees paid with the transactions (iv) through block building by adding, deleting, or reordering transactions (also known as gains through miner extractable value or MEV). Earning revenue through the first three avenues is fairly straightforward and something that every staker can easily do by observing the state of the network and following the protocol. The last avenue of block building, on the other hand, can be sophisticated, due to which not all stakers may be able to maximize their revenue in an optimal manner. Having non-uniform revenue amongst different stakers is a risk, as it can lead to centralization. The approach of **Proposer-Builder Separation (PBS)** separates the task of block building from block proposing by creating a market of builders, each of which can perform the role of block building by optimizing for MEV. The proposer then proposes the block content of the builder that (generates and) bids the highest value. This process is touted to reduce the barrier to entry for stakers by obviating the need for a staker to own sophisticated infrastructure and still earn revenue closer to what it would if it did have the infrastructure (minus a small profit that the winning builder keeps for herself). The collected revenue through this bid can also perhaps be redistributed differently, e.g., paying the set of all stakers by burning the bid value, or somehow paying back the users.

The PBS Fair Exchange Problem

Separating the task of building and proposing so that it is performed by two different stakers brings inherent challenges that relate to the trust between the two. In particular, a key challenge relates to a fair exchange between the builder and proposer *where deciding the builder block, as is, at the proposer's slot is exchanged for fees bid by the builder and the builder's block content[^1].*

The fair-exchange problem has been studied extensively in the cryptography literature. It is known that even a mere fair-exchange of cryptographic strings among mistrusting parties cannot be solved deterministically[^2]. Therefore, below we will explore mechanisms for incentivizing participants to behave fairly in the PBS fair-exchange.

To understand the concern better, observe that any party that makes the first move may be at a disadvantage.

Builder's concern: leaking block content and losing fee

Builders invest effort into producing a block that generates the most MEV. For instance, the builder may identify an arbitrage opportunity for a token available at two different sources, and the builder may exploit this opportunity. Typically, this may involve adding transactions containing the builder's public key such that the builder buys and sells from the two sources and earns in the process. However, if the builder reveals the block content to the proposer as is, the proposer can perhaps identify these arbitrage transactions and replace them with its own transactions, thus stripping the builder of its gain. Thus, the builder is concerned about releasing its block before it knows that it will make it to a specific slot on the chain as is. Moreover, the builder may be willing to pay a fee to the staker in the form of a bid; however, if its block is not included in the slot, it would lose the fee too.

Proposer's concern: unavailability of builder data and not obtaining fees

If the proposer proposes a block without having access to the block content, then it runs the possible risk of unavailability of data, i.e., a situation where the network never learns what transactions were proposed in the block. This can happen, for instance, if the builder crashes or goes offline. Data may also be unavailable if an adversary easily performs an attack where it wins a bid but no block was built and released. In such situations, the proposer may not be able to collect transaction fees or other block rewards for performing its duties. Moreover, if a fee is supposed to be included as a part of the block itself, then the proposer does not earn the fee either.

Addressing the Fair Exchange

In this post, we will discuss potential solutions and the associated trade-offs to perform such a fair exchange in the context of HotShot as the underlying consensus protocol. We assume that the proposer has already selected a winning builder among the set of builders.

A quick overview of the HotShot protocol. Since our goal is to discuss the fair exchange problem in the context of HotShot, we will first provide a brief overview of the protocol in the absence of PBS. The goal is not to explain the protocol itself, but only to provide sufficient terminology and invariants that may later be used in this post.

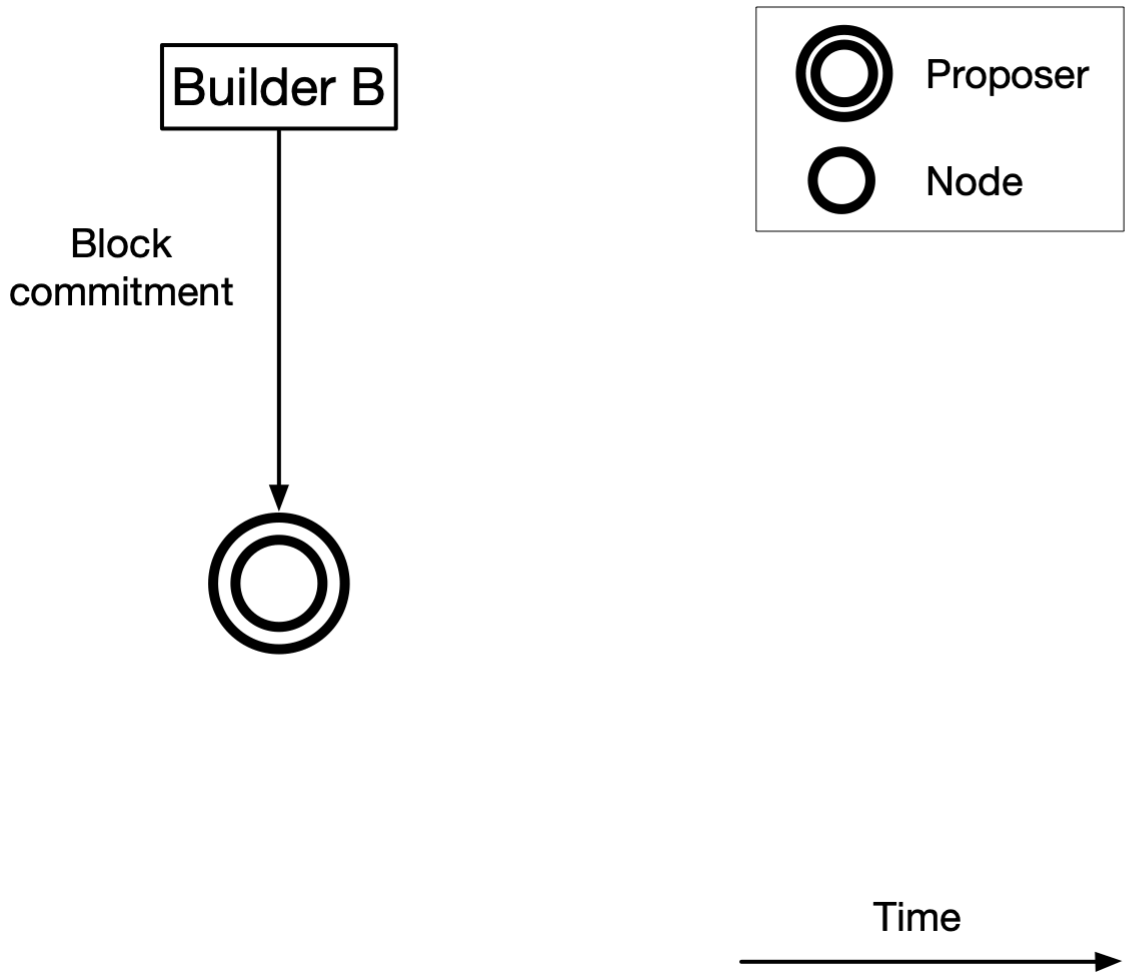
The HotShot protocol progresses in a sequence of views where, in each view, a designated leader (also called proposer) proposes a block to all other nodes.

- The nodes determine whether the proposal is valid and, if so, send a vote to the leader of the next view.
- The next leader creates a new block extending the previous one to create the next proposal, and so on.
- To create a new proposal, in addition to the block content, each leader includes a quorum of votes (called quorum certificate or QC) from some previous view. If such a QC of votes from the immediately preceding view is not available, for instance, because the previous leader crashed, then it includes the QC from the highest view known to it.
- Each node also maintains the highest QC known to it; we call it being locked on the block based on the highest QC. The node will vote for a proposal containing a different block only if the QC included in it is not from a lower view than the QC formed for the locked block.
- Two blocks in consecutive views that obtain QCs decide (finalize) the first block and every block preceding it. A block is said to be decided (finalized).
- While QCs from two consecutive views are needed, intuitively speaking, the existence of a QC is a step towards a decision, and if one can ensure that the QC is received by all the nodes at an appropriate time, then one can guarantee that the block will eventually be decided.

Given this background, let us understand the space of solutions.

Strawman 1: Commit-and-Reveal

In this solution, the builder shares a commitment to its block with the proposer. The proposer proposes and eventually obtains a decision (finality) on the block with all stakers. The builder then reveals the block content associated with the commitment. The block content also contains fees (bid amount) to be paid to the proposer, and thus processing the block also settles a payment to the proposer.



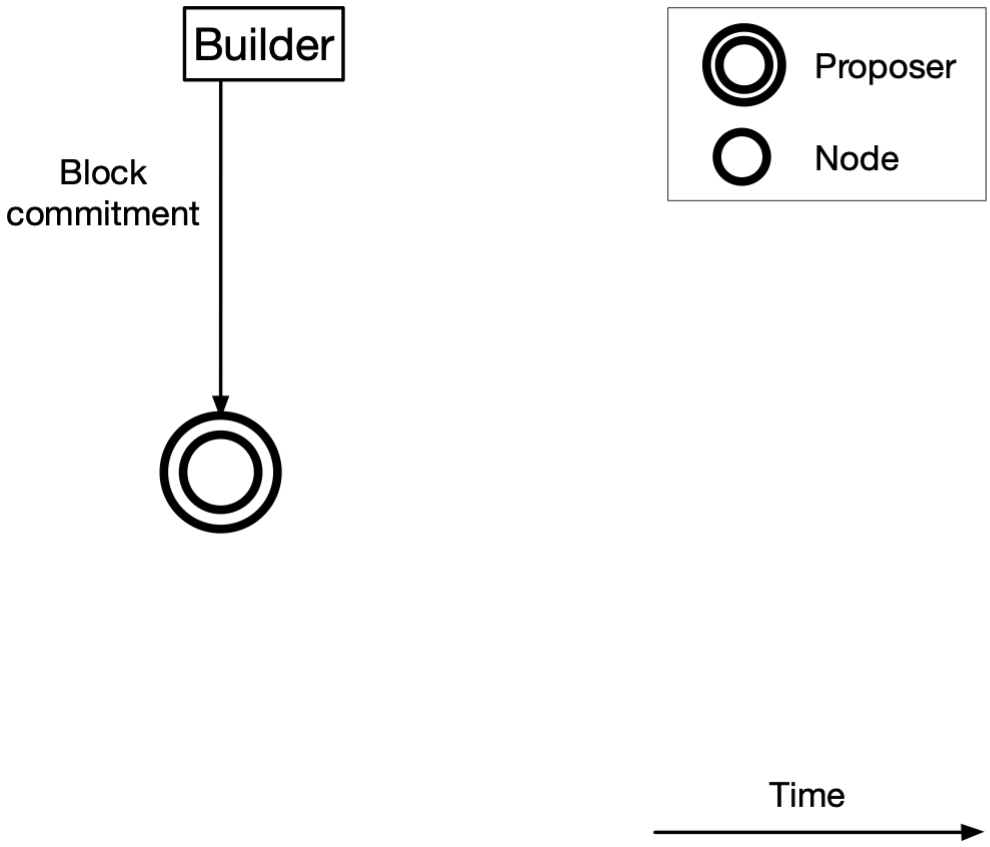
A key advantage of this approach is that the builder's block content is not accessible to any party until it has been decided, and thus the builder cannot be cheated against, e.g., by reusing the builder's block content.

However, this solution has three key concerns:

- The builder may cheat and not include a fee (or include a reduced form of fee) in the block content. One way to fix this concern is for the builder to prove to the proposer of the existence of a correct transaction, which can potentially complicate the protocol.
- A malicious builder may not reveal the content of the block after it has been committed. In fact, this can also be performed by a rational builder depending on the state of the system at reveal time, e.g., one of the transactors of the system learns that some of the transactions included in the block are not favorable and it bribes the builder to not release content. Whether or not the block content was revealed requires a separate consensus, which is cumbersome.
- In HotShot, blocks are proposed in a pipeline such that it takes two (consecutive) blocks to decide the first one. Hence, if the block content is not released before deciding, then the second block must be proposed without knowledge of the contents of the first. We say builders of this block have access to antiquated state of the system. This also disallows them from accurately determining whether transactions that they subsequently add are valid, or figuring out the gains through MEV for the block they build. Consequently, they are also unable to accurately determine bids for the subsequent blocks.

Strawman 2: Share Block Contents with a Committee of Stakers

This solution aims to address the first two concerns of the previous strawman. In this solution, the builder reveals the block content at the same time as revealing the block commitment with the proposer. However, to ensure privacy of the block content, it instead secret-shares the block with a committee of (or all of the) stakers.



Under an honest majority assumption, secret-sharing ensures that a minority corrupt subset of the committee cannot learn the block contents, but a majority subset of the committee can indeed reveal the block contents. This majority subset reveals the content after the block commitment has been decided. This solution has been suggested by multiple academic and deployed works such as Ferveo, Fino, Shutter DKG, and others. To pay the fees, the builder sends an unconditional transaction along with the block content to the proposer.

This solution is at the other extreme from Strawman 1, and attempts to address its concerns. However, this solution has a different set of concerns.

- This solution requires an honest majority assumption to maintain the privacy of block content. Observe that learning and modifying the block content through a collusion of parties can be done covertly; in fact, recently this has been formally shown by Wadhwa et al. in the presence of a rational committee. Thus, when the stakers in the committee are rational, this does not provide privacy to the builder.
- Sending an unconditional transaction as a fee to the proposer implies that the builder ends up paying without having any guarantees of the other side of the exchange.
- In addition, the concern with the antiquated state holds in this solution too.

An alternative to addressing the first concern is to protect the content of the secret shares using a TEE (or perhaps rely on a TEE entirely to protect the entire block content) as suggested in Wadhwa et al. This solution fixes the privacy concern related to the rationality of the committee. However, this mandates a new assumption plus the need for specialized hardware for every committee member.

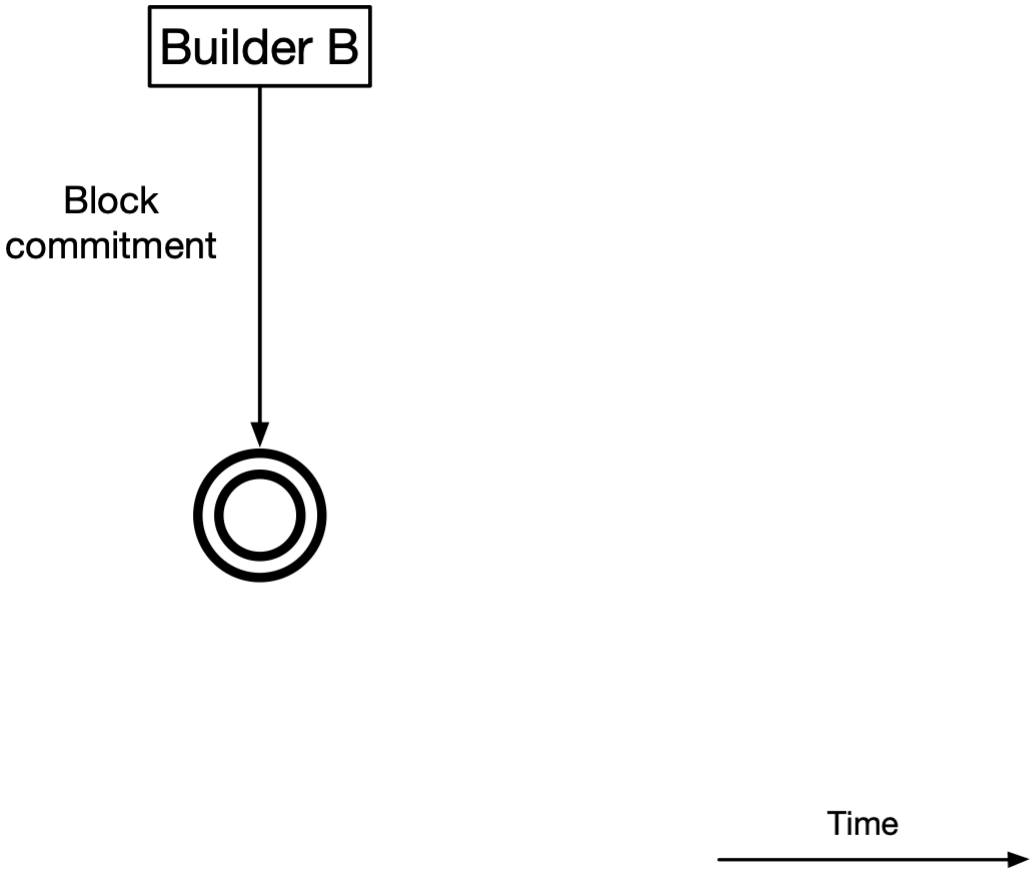
A Potential Space of Solutions

The two strawman solutions protect one of the two parties while exposing the other to obvious attacks when the protected party is rational. Independently, a common concern with both approaches is access to only antiquated information for subsequent builders. This primarily happens because the block content is released only after the commitment is decided. On the other hand, if a builder reveals the block content earlier, it runs the risk of losing block privacy in the event that the decision fails. Thus, there appears to be a tradeoff between (i) obtaining block content privacy and (ii) access to the most recent state for other builders for subsequent blocks. We will now explore a set of solutions where a trade-off between the two properties is possible in the context of HotShot.

As mentioned earlier, a HotShot block is decided when it and its successors include blocks that obtain quorum certificates (QC) in two consecutive views. A key invariant maintained by the protocol is that, whenever a node decides (by obtaining the second QC), 67% of the stake (and thus 34% honest stake) must have locked on the block based on the first QC. The protocol can decide on an equivocating block only if at least 34% of the stake maliciously does not respect their locks, in which case, the violation would be detectable. In other words, if 34% honest stake is locked, i.e., has access to the first QC before the parties quit their view, then the proposal is in effect, irrevocable and will eventually be decided by all nodes. Thus, from a builder's standpoint, the goal is to somehow ensure that 34% honest stake has access to the first QC before the nodes quit their view. Said another way, if 34% honest stake has access to such a QC, then no conflicting block can ever be decided (and thus stealing the block content is not useful). Thus, the builder can safely reveal the block contents even before the block has been decided if it can ensure this invariant.

Attempt 1

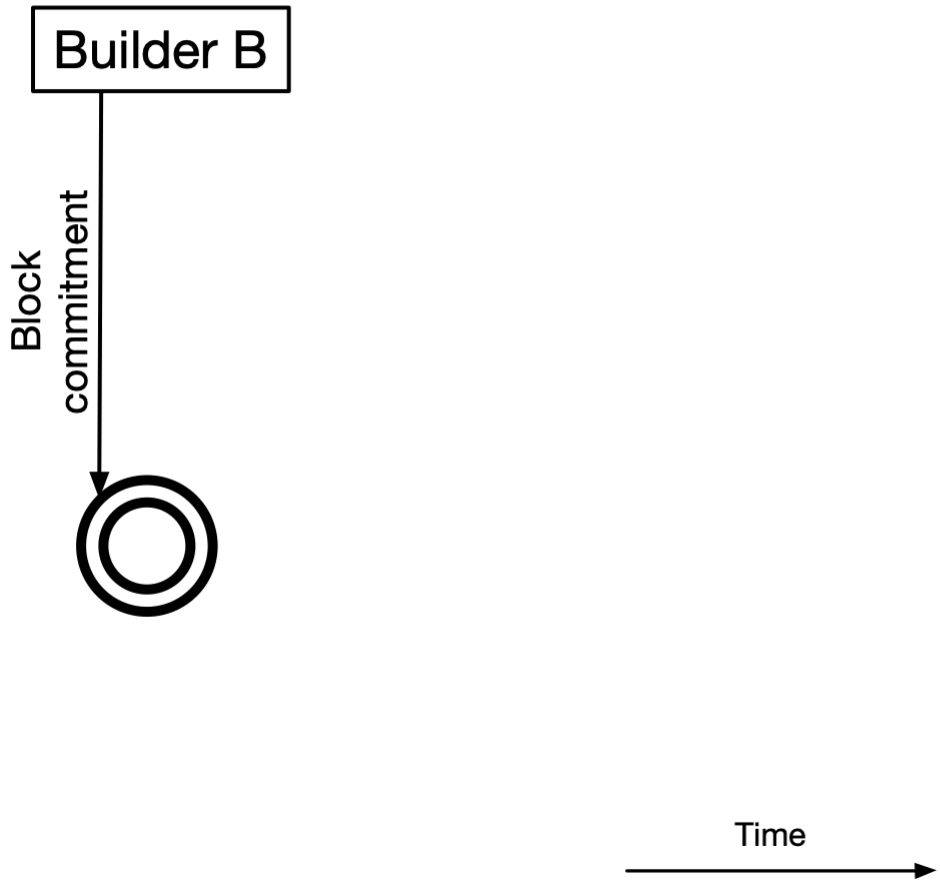
A potential way to ensure this invariant could be the following. A proposer can propose a block commitment for a block, obtain a QC for it and share this QC with the builder. If the builder shares this QC with all the nodes before they quit the current view, then the block will eventually be committed, and it is safe for the builder to reveal the block contents as well. Observe that this modification ensures that the reveal happens one step earlier than with the strawman solutions.



However, this does not entirely achieve our goal of builders having access to the most recent state. In HotShot, a proposal in the next view contains a QC from the previous view and commitment for a new block data. Thus, if the data release happens only after the QC is formed, the new block formation did not have access to this information.

Attempt 2

It turns out that we can take this idea one step further and reveal the block contents to all validators (and other builders) even earlier. In this solution, a proposer essentially delegates its proposal duties to a builder. The builder shares the block commitment with the proposer. The proposer then sends a signed proposal containing the builder's block commitment to the builder. If there is a sufficient amount of time before nodes quit the view, the builder shares the proposal as well as the block content with all the nodes. Once it obtains a QC corresponding to the block, it immediately shares the QC with all the nodes to ensure that they lock on it (and the block will be irrevocable). Paying the bid fee, i.e. rewarding the proposer, can be achieved in the following manner: the proposer includes the agreed-upon bid in the proposal, and the builder includes a corresponding transaction as a part of the block content. Each of the voting nodes can indeed verify that such a matching transaction exists to determine block validity.



Observe that this solution resolves the antiquated state problem and ensures that other builders have access to the most recent state for building subsequent blocks. Moreover, by delegating the proposal responsibility to the builder, the builder is essentially responsible for atomically performing the fair exchange (or not). In other words, the builder can ensure that its block contents would not be stolen by other nodes to create conflicting blocks and at the same time, the voting nodes can ensure availability of the data and the fact the proposer gets paid.

There are a couple of implicit assumptions that this solution makes. First, to ensure that there is sufficient time for the builder to make the proposal and obtain a QC and so on, the builder needs to be an integral part in knowing the consensus state (even if it is not a voting node). Second, the builder can guarantee the formation of the QC and send the QC to the nodes only when the network is synchronous. Under asynchrony, the fair exchange guarantees may not hold (even if safety and liveness for the consensus will still hold), i.e., no QC is formed but a subsequent leader uses the block content. However, in most practical situations, it may be reasonable for a builder to know a bound on the time it takes for its message to reach other nodes. Finally, while a rational builder is incentivized to perform this exchange (which is why it bid in the auction in the first place), a malicious builder can decide to not propose at all.

Raising the Barrier to Entry for a Malicious Builder

The last assumption w.r.t. a rational builder deserves more thought. Why is this assumption reasonable, or what is the barrier to entry for a malicious builder? Clearly, if the builders are entirely permissionless, then the system can suffer from a Sybil attack by builders posting high bids but not performing subsequent actions. Thus, the builder needs to lose some money for such behavior but at the same time not be punished when the proposers are malicious. A way to achieve this would be for builders to deposit a small amount of stake into the system before participating. Whenever they bid, they essentially make a promise to lose the small stake if they get chosen by the proposer but their block is not committed. In a scenario where their block is committed, they would still pay the proposer their bid through the block content, and the protocol proceeds as described earlier. Otherwise, if their block is not committed, the small amount of stake can be burnt. Observe that a rational proposer does not have an incentive to make a builder's stake be burnt, e.g., by delaying the sharing of the signed proposal. This is because the proposer only earns more, i.e., gets the bid fees, if the block is committed. Thus, burning the small amount of stake deters builders who intentionally do not release blocks but does not affect honest builders.

Comparison and Compatibility with Ethereum's Approach to PBS

MEV-Boost. Many Ethereum validators run an instance ofMEV-Boost, which acts as an out-of-band version of PBS. Their solution addresses the fair exchange problem by relying on a trusted intermediary called a relay — the relay is responsible for holding custody of the block, validating it, and making the data available after the proposal containing the block header is revealed. Until then, the proposer does not have access to the block contents. Their approach also does not wait for the proposal to be committed before revealing. Thus, given their reliance on the GHOST protocol for proposing blocks within an epoch, it is potentially possible for a proposed block to not be decided (finalized) after the data contents are released. Thus, builder's block privacy is not maintained under all circumstances.

Enshrined PBS. There are many proposed Ethereum Research posts to enshrine PBS (e.g.one, two). While these solutions may differ in their details, the key goal in these posts is to reduce the trust provided by relays in MEV-Boost. At a high-level, the solution approach breaks the 12-second slot time into multiple parts where (i) the proposer first releases a proposal, (ii) a set of committee members attest to the existence of this proposal and also observe for equivocation, (iii) the builder releases the block contents or a proof of equivocation, and (iv) a separate committee attests to the fact that the builder has released (or not released) the block contents in time. In essence, their approach is similar to the one described in this post where if synchrony holds, parties (builders, proposers) would have sufficient confidence to take the next step. However, the relationship between the votes by these intermediate committees to the commit of a block is currently not entirely specified. In contrast, in our solution, since we decide within two views in the optimistic case, so far as synchrony holds, a builder can ensure that its proposed block will be added to the chain and decided at the specified slot.

[^1]: Well, there are also other challenges/questions such as whether the separation affects the properties of the underlying consensus mechanism, whether the users are better off with or without PBS, etc. but we will not discuss those questions here.

[^2]: Even, S., and Yacoby, Y., "Relations among Public Key Signature Systems", Technical Report #175, Computer Science Department, Technion, 1980.