State of research: increasing censorship resistance of transactions under proposer/builder separation (PBS)

Special thanks to Francesco for coming up with the core idea in Solution 2, and thanks to Francesco, Justin Drake, Alex Obadia and Phil Daian for feedback and review.

How does proposer/builder separation (PBS) work?

In the current transaction market, the **block proposer** (today: a miner, post-merge: a validator) directly chooses which transactions to include in the next block by looking at which transactions in the mempool pay the highest priority fee. This puts the block proposer in a position to use sophisticated strategies to choose which transactions to include, or even include their own, to take advantage of opportunities such as DEX arbitrage and liquidations (hereinafter just called "**MEV**" for simplicity) to maximize their profits. The complexity of these strategies creates a high fixed cost in running an effective miner or validator, and advantages centralized pools that take on this task on behalf of their participants.

Proposer/builder separation (PBS) fixes this by splitting the block construction role from the block proposal role. A separate class of actors called **builders** build **exec block bodies** (essentially an ordered list of transactions that becomes the main "payload" of the block), and submit bids. The proposer's job is only to accept the exec block body with the highest bid. Notably, the proposer (and everyone else) does not learn the contents of any exec block body until *after* they select the header (and hence the body) that wins the auction. This **pre-confirmation privacy** is needed to prevent "**MEV stealing**", where sophisticated proposers detect builders' MEV extraction strategies and copy them without compensating the builder.

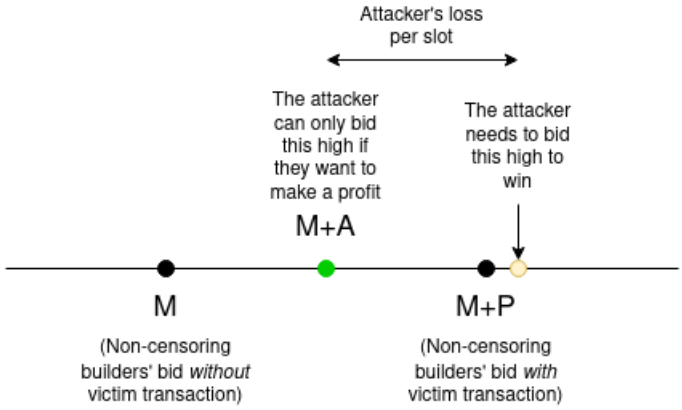
What are the censorship resistance challenges in PBS vs status quo?

First, consider the status quo scenario. Suppose that there is a 150k gas transaction, and the current gas limit is 30m (target 15m). Powerful economic actors are trying to censor that transaction. In the current transaction market, if that transaction is willing to pay the base fee and a priority fee above ~1-2 gas, proposers will be willing to include it; an average block has ~15m gas of slack space for them to do so. To censor a tx sender willing to pay X for inclusion, the attacker would need to raise the basefee to above $X / 150k$ per gas and hold it there. If they do this, they will price out most other users - after all, if the basefee is high enough to price out a transaction important enough to attract a targeted censorship attempt, it will be high enough to price out most other activity too! Hence, the attacker will likely have to pay for most transactions in the block themselves. Conservatively, **the attacker may have to pay up to $(X / 150k) * 10m = X * 66.7$ *per slot* (so $\sim X * 20000$ per hour)**.

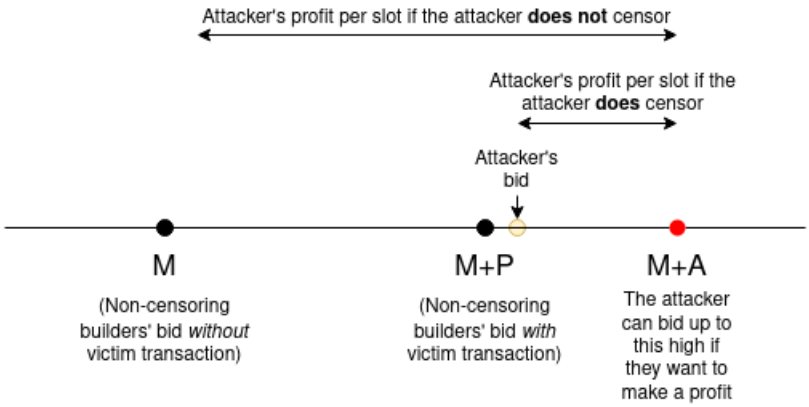
Now, let's explore the PBS case. Suppose that there is a builder that is consistently better than other builders, either because they convinced unsophisticated users to run wallets that *only* send transactions to them ("**proprietary order flow**"), or because they have better algorithms and data access for discovering MEV opportunities. Suppose there is some specific **victim transaction** that the builder ("the **censoring builder**") is trying to censor. Let:

- **M** be what the non-censoring builders would bid if there were no victim transaction
- **$P = X - basefee * 150k$** is the priority fee portion of the total txfee of the victim transaction
- **$M + P$** is the non-censoring builders' bid if the victim transaction *is* available to be included
- **A** is the censoring builder's **advantage** (difference between fees + MEV that they earn and fees + MEV that the best non-censoring builder earns)
- **$M + A$** is the censoring builder's bid, regardless of whether or not the victim transaction is available (as the censoring builder does not want to include the victim transaction)

If $P > A$, the non-censoring builders would be able to bid $P - A$ higher than the censoring builder's zero-profit bid ($M + A$). To keep censoring, the censoring builder would have to increase their bid to $M + P$, and lose $P - A$ per slot to keep outbidding the non-censoring builders (and additionally sacrifice the opportunity to profit A per slot, so total loss is P per slot). This is still expensive, but note that it is much less expensive than $X * 66.7$ per slot.



If $P < A$, the censoring builder wins the auction despite not including the victim transaction, though they do have to increase their bid to slightly above $M + P$ and thereby sacrifice P of profit per slot (the censoring builder's profit becomes $A - P$ instead of A).



Note that in both cases, the attacker loses P per slot for censoring - still a significant ongoing cost to keep just a single transaction out of the chain, but much less expensive than the cost in the per-transaction fee market case.

PBS censorship economics and block time

One important consequence of the above analysis is that in PBS, the cost of censorship is *per slot* (technically per block, but for simplicity we don't care about the difference, as almost all slots have a block in them on the Ethereum beacon chain in practice). If blocks come 10x less frequently, censorship is 10x less costly. If blocks were to come 10x *more* frequently, censorship would be 10x more costly. This is counterintuitive, but follows from the logic in the previous section. Note particularly that the status quo fee market is kind of like PBS with 100+ auctions for single-transaction placements running in parallel; however, it is a broken PBS market because it lacks pre-confirmation privacy.

So one natural question to ask is: can we try to extend PBS to get the benefits of very fast blocks, but have those blocks come *in parallel* instead of in series?

What are the design goals of any anti-censorship scheme?

- **DoS protection and no free data-availability**: the proposer (or the builder) should not be able to include data which no one pays for, as that could be abused by an attacker to bloat the chain or by rollups to cheat on transaction fees. For example, this means that if a block contains a transaction that turns out to be invalid because of insufficient balance or other transactions in the same block, the protocol must still charge the proposer base fees for it.
- **Minimal additional bandwidth consumption**: the mechanism should be efficient not just with on-chain data, but also with data in the p2p network. For example, having hundreds of redundant full bodies from different builders floating around the network is not realistic.
- **Does not re-introduce proposer centralization**: the whole point of PBS is that it doesn't require proposers to be sophisticated. We don't want to create a mechanism which reintroduces a benefit for proposers to be sophisticated and

hence an incentive for proposers to enter into further extra-protocol auctioning relationships or join pools.

- **Ideally, allow proposers to be stateless**: validators being able to be fully stateless (once Verkle trees are also included) would be a significant boon for further decentralization and scalability.
- **If we rely on altruism, don't make altruism expensive**: we generally can rely on assumptions that at least a few percent of proposers are altruistic and will ignore bribe offers and accept censored transactions. However, this assumption is less realistic if doing so is expensive in-protocol. Proposers should not have to sacrifice large amounts of revenue to help ensure censored transactions get in.

Solution 1: can we run many pre-confirmation-private PBS auctions in parallel?

Suppose that there is an auction for a **primary exec block** and N-1 auctions for **auxiliary exec blocks** (so N auctions total) running in parallel for each slot. The blocks are *executed* in series (primary first, then auxiliary in order), but they are *agreed upon* in parallel.

Suppose that a transaction sender is willing to pay P in priority fees. To exclude that transaction, a censor would need to be willing to pay the controller of each auxiliary exec block auction P+1 per slot, so they would have to pay (P+1) * N per slot.

The attacker has to make a bid *per slot*, and not only when an auxiliary inclusion is attempted, because if attackers only automatically outbid auxiliary inclusions, proposers would start submitting fake self-bids to force the attacker to outbid them.

But there is a problem: how would the auxiliary exec blocks be constructed? The builders constructing primary exec blocks ("**primary builders**") have a relatively easy time doing so, because they see the pre-state they are building on top of: it's the post-state of the previous block. The builders building the auxiliary exec blocks ("**auxiliary builders**"), on the other hand, have no such luck: whatever block they build will be executed on top of the primary exec block (and lower-index auxiliary exec blocks) *in the same slot*. Hence, the auxiliary builder does not know what state they are building on top of when they are choosing what transactions to include.

If an auxiliary builder includes a transaction that was already included in a previous block, they have to pay gas for the data and the execution up until the point of signature and nonce verification, but they do not get paid the fee from the transaction. Hence, they would lose money (**it's not safe to design a protocol that waives the data+verification gas fee in this case, because that could be abused by rollups and other store-data-on-the-blockchain use cases**).

This opens the question: **do auxiliary builders have a safe strategy for including transactions in auxiliary exec blocks**? This strategy should be profitable (in expectation) in the normal case and provide protection if a censoring primary builder is explicitly trying to attack them. As an example of an attack, a censoring primary builder could censor transactions until they expect auxiliary builders to start including them, and then include those transactions suddenly. If they keep doing so, they could cause auxiliary builders to lose money; once the auxiliary builders lose too much, they will drop out of the market, giving the censoring builder a free hand to censor transactions.

A minimum viable auxiliary builder strategy

Remember that in the "happy case", auxiliary blocks would be empty, as everything that wants to be included would be included in the main block. So the strategy only needs to activate in the case where censorship really is happening.

Let:

- N = the total number of auctions happening per slot (including primary and auxiliary)
- P = the transaction's priority fee
- B = the current base fee

First, for simplicity, assume N = 2 (so, only one auxiliary exec block per slot). Assume also P = B (censorship victims always pay 2x to attract auxiliary builders and avoid censorship). Here is a proposed auxiliary builder strategy:

> If a auxiliary builder sees a transaction that was first received k slots ago, was not included yet, and has been valid for inclusion since then, then the builder should include the transaction in their proposed block with probability $P_{include}(k) = 0.00000001 \cdot 2^{max(k, 1)}$.

Suppose the censoring primary builder accepts victim the transaction after k slots. Either k = 0, in which case they only have a 0.00000001 chance of "trapping" the auxiliary builder (meaning, the censoring builder publishes in the same slot as the auxiliary builder, causing them to lose money), or k > 0, in which case the probability the attacker traps the auxiliary builder equals the probability the auxiliary builder includes the victim transaction *before* the attacker (as $P_{include}(k)$ = $\sum_{i=0}^{k-1} P_{include}(i)$ for $k \ge 1$).

In the latter case, the auxiliary builder on average breaks even. This is because in the case where the attacker traps them,

they lose B, and in the case in which they include a transaction they gain P, and we assumed $P = B$ above. Reducing the exponential base to below 2 makes the math more favorable and ensures that the auxiliary builder is *profitable*. We could extend to arbitrary priority fees by setting the base to $1 + \frac{P}{P + B}$.

The $N > 2$ case is trickier because of the possibility of interference: what if a transaction gets included in two auxiliary exec blocks at the same time? However, if we assume that auxiliary builders use independent random strategies, the chance of a collision is manageable (a simulation script suggests it's close to $\frac{1}{4}$ if the base is 2, regardless of the number of builders).

In practice, the above numbers are almost certainly too pessimistic. Censored transactions will probably be included by auxiliary builders after only a few slots, and auxiliary builders will react dynamically to attacks. Additionally, transactions cost less when they fail than when they succeed, so in practice P would only need to be a few percent of B for many transactions. Rather, the above is intended more as an existence proof that auxiliary builders *can* be profitable using a fixed specific strategy even under attack.

Why not let auxiliary builders choose which transactions to exclude via a bitfield after the primary exec block is known?

A bitfield still carries enough information for an attacker to encode instructions to a smart contract for how to exploit MEV opportunities. Additionally, if there are *multiple* auxiliary builders, they would have to provide their bitfields sequentially. Hence, this would simply convert the auxiliary auction into a series of primary auctions.

Solution 2: can we still use proposers ("hybrid PBS"), but only for inclusion-of-last-resort?

A natural alternative approach to protecting against censorship is to make a hybrid between the status quo transaction fee market and PBS. This approach would lean on proposers for censorship resistance. Only a few proposers need to actually use this scheme; even if 95% of proposers are somehow successfully bribed to pretend to see no transactions and only accept PBS blocks, the remaining 5% mean that on average any victim transaction would still on average be included after 20 slots.

The scheme works as follows. During each slot:

1. **Proposer broadcasts `crList`**, a list of transactions that the proposer sees that deserve to be included (ie. they have the correct nonce and sufficient balance in the state and sufficient priority fee and max-basefee to be included). The `crList` can only contain one transaction per `sender`. The proposer also signs and broadcasts a **`crListSummary`**, which includes the `tx.sender` and `tx.gaslimit` for every `tx` in the `crList`.
2. **Builder makes a proposed body**, which must include the `crListSummary`
3. **Proposer accepts winning header**
4. **Builder publishes body**. Verification of the body requires checking that for each `(sender, gaslimit)` in `crListSummary`, either `block.gasused + gaslimit > block.gaslimit` or `sender` is the sender of some transaction in the block

A more sophisticated alternative to further save space is to require `crList` to be sorted in descending order of `tx.gaslimit`, and then require the body to only include (i) a Merkle proof into the first tx in the `crList` that cannot get included, and (ii) a bitfield of which txs in the block are in the `crList`. Anyone can verify this by reconstructing the portion of the tree before the Merkle proof and checking that it fits with the Merkle proof.

Note that in either solution, the if the proposer constructs `crList` incorrectly, the builder potentially cannot safely propose an exec block body. In this case, the builder should simple not propose anything, just as they would if the proposer had never broadcasted `crList` at all.

Hybrid PBS and statelessness

A major secondary goal of PBS is to allow validators to be fully stateless (once Verkle trees are also included). PBS accomplishes this because only builders need to actually create bodies; builders would also be tasked with attaching witnesses to their blocks, allowing validators to statelessly verify the blocks.

To extend this property to this scheme, each transaction would need to come with a simple witness proving the balance and nonce of the transaction sender. If users are stateless, this witness could be added by intermediary nodes (eg. builders could volunteer to do this).

How would hybrid PBS interact with layer-2 account abstraction (eg. ERC 4337)?

ERC 4337 fundamentally changes the game theory of censorship resistance because of decouples the Ethereum-level

concept of a *transaction* from the economic concept of an *object of user intent*. Ultimately, we are trying to secure the censorship resistance of objects of user intent. When the two are the same (as they are today), this boils down to securing censorship resistance of transactions. In ERC 4337, however, a transaction becomes a *wrapper* containing many objects of user intent (in ERC 4337 called *user operations*).

In "regular" PBS, this distinction does not matter, because the protocol only directly interacts with exec blocks. Builders can create an exec block containing many ("regular") transactions, or they can create an exec block containing one transaction wrapping many user operations, or a mixture of both. The mechanics in both cases are exactly the same; there is little difference between a wrapper and a wrapper of a wrapper.

If auxiliary exec blocks are used for censorship resistance, the mechanics continue to be the same. **If hybrid PBS is used, however, then the treatment of status-quo transactions and ERC 4337 user operations diverges**: censorship resistance of status-quo transactions is ensured directly, but censorship resistance of ERC 4337 user operations is only provided for indirectly.

Either ERC 4337 or hybrid PBS would need to be modified slightly to ensure it inherits censorship resistance in a hybrid-PBS world. To understand why, first consider a flawed argument for why ERC 4337 will retain censorship resistance:

> "Transactions" carrying ERC 4337 user operations map directly onto the concept of "auxiliary blocks" described above. If a given ERC 4337 user operation is not included by a primary censoring builder, then anyone (we'll call them "the relayer") can create a transaction (a pseudo-auxiliary-exec-block) including that user operation, and expect profit from doing so. Sure, the censoring builder could include the user op at that exact time and deprive the relayer of their revenue, but not always, and so on average the relayer will be profitable for the same reasons as explained in the auxiliary builder strategy section above.

The key flaw in this argument is that the "not always" is false. This is because of a fundamental difference between the auxiliary exec block case and the ERC 4337 transaction case. In the auxiliary exec block case, the primary and auxiliary exec block bodies are being committed to *at the same time*. Hence, neither side has an ability to see what the other is doing and front-run them. But in the ERC 4337 transaction case, the winning builder *can* see the transactions, and so they *can* frontrun them.

Solution 1: modify ERC 4337

We can modify ERC 4337 as follows: **user operations will need to have an ability to specify a single relayer that is allowed to include them**. This protects the user operation from being "stolen" (though note that the censoring builder *colluding* with a user could still drain money from auxiliary builders, because the censoring builder could front-run the relayer with a *different* user operation with the same nonce).

Solution 2: modify hybrid PBS

Require the transactions in the crList to be executed *before* all other transactions. This prevents the builder from frontrunning. However, actually using this feature would be more costly for the proposer because it could reduce the revenue that they get from the MEV auction, as the transactions in the crList could themselves be MEV-extracting. Hence, going down this route could create more incentives for proposers to be more sophisticated in crList construction, which brings back centralization incentives.

Conclusions

PBS is an actively evolving research area. Mitigating validator centralization risks that come from MEV is very important for any blockchain protocol that seeks to retain its decentralization. However, preventing builder centralization from turning into a different type of censorship vulnerability is equally important.

At this point in time, I would favor hybrid PBS in the short term. Hybrid PBS gets the best-of-both-worlds between the status quo and the "pure" PBS model:

- Even a single altruistic proposer can include any transaction that other proposers and bundlers are censoring, so **we retain present-day censorship resistance**
- The priority position needed for MEV extraction is auctioned off to a separate class of specialized actors so validators do not need to touch it, so **we mitigate MEV-driven validator centralization risk**
- Proposers do not need to personally execute blocks, so **validators can be fully stateless**. This means that **it becomes safer to significantly increase the L1 block size**.

In the longer-term future, concerns with hybrid PBS and the desire for more abstraction are good reasons to also explore some notion of auxiliary exec blocks, that get constructed in parallel with primary exec blocks.

Additionally, **the arguments in this post present some interesting reasons to potentially consider eventually recognizing ERC 4337 user operations at the protocol layer**. In the short term, ERC 4337 should stay as an extra-protocol ERC because it's much faster to innovate there. In the longer term, however, it's clear that enshrining it and explicitly recognizing objects of user intent at protocol layer opens up more options for ensuring that they are protected from censorship.