PBS censorship-resistance alternatives

[TOC]

crLists

High-level idea:

We want to make it so proposers are able to combat censorship by forcing inclusion of some transactions, but we don't want to do it in a way which is bandwidth-intensive (i.e. can often result in redundant gossiping and/or block inclusion of transactions), which disadvantages altruistic proposers by requiring them to sacrifice MEV or which gives sophisticated proposers some power that they can exploit to extract more MEV than unsophisticated ones. When builders are not censoring, it would be ideal if the protocol would essentially behave as in the regular PBS proposal, with the proposers just choosing the most profitable block, and minimal overhead

The key to do this is to understand what censorship looks like after 1559, given a view of the mempool: we suspect a builder is censoring if there is blockspace which could be allocated to currently eligible mempool transactions and is instead left empty, i.e. if blocks are not as full as they could be given the current mempool. Obviously there is some degree of subjectivity given by the view of the mempool, but what we can do is let the proposer impose their view (or a slice of it) in advance, so that we can distinguish censorship from simple divergence of views of the mempool.

Rather than allowing proposers to directly force inclusion of their chosen transactions, we instead allow them to force builders to fully utilize the available blockspace: if they cannot do so, they must use the unused space for the proposer-selected transactions

Cost of censorship

For an overview about how the cost of censorship changes with PBS, if we don't assume the bribing model (where the whole validator set is bribable), see Vitalik's State of research post

crLists + EIP 1559 change this dramatically:

- On a timescale of a few slots, the cost of censorship is still linear in the number of slots for which you want to censor, but with a much better constant. Without crLists, a dominant builder only incurs the opportunity cost of not including the transaction, per slot. With crLists, they are forced to fill each block to its gas limit, which is many times more expensive
- On a longer timescale, the cost of censorship is exponential in the number of slots, because the basefee goes up and filling blocks becomes exponentially more expensive (and it prices out real transactions, leaving the attacker to fill entire blocks)

Design goals from Vitalik's post

- **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 (onceVerkle 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.

Original basic scheme

1. Proposer broadcasts crList, a list of transactions that the proposer sees that deserve to be included (ie. they have the correct nonce, sufficient balance and `maxFeePerGas >= basefee`). 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 only if all transactions in crList are valid, and the body 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

Notes:

- The proposer is free to choose any headers, maximizing payment, but publishing crList potentially restricts the set of headers they choose from. If we are not relying on a little bit of altruism, and instead want the proposer to be incentivized to include transactions in crList to get an extra payment, we might be worried about the possibility that the proposer publishes multiple crLists. For example, publishing an empty one together with the desired one would allow all builders to participate in the auction. We could introduce slashing for crList equivocation to get around this problem, though proposers could create trust-based relationships with a few builders to maximize profits while avoiding slashing.
- If we are concerned about such proposers, we might also be concerned about what would happen if builders openly published blacklists consisting of transactions they are not at all willing to include (for example because of compliance, sanctions etc...). Proposers could listen for such blacklists and make a maximal crList which does not intersect with any of the ones sent by the best builders, to maximize profits while trustlessly avoiding slashing.
- These issues with incentives can be mitigated in two different ways, which will separately be explored in this document (and which of course have tradeoffs compared to the basic scheme):
  - having a separate crList proposer, which does not have a conflict of interest driven due to wanting to maximize PBS auction revenue
  - having crList be published after a header has been chosen, so profit can be maximized independently of the list

Refined versions with no free data-availability given by the gasLimit field

Interactive

- In step 2, after making the block the builder filters crList by removing all transactions such that gasLimit < remaining gas, since they are not includable.
- The block body still includes a summary, containing all senders in the filtered list. The header contains a hash of the summary.
- In step 3, the proposer does the same filtering as the builder, since the gas used by the block is in the header, and accepts the header only if the hash of the summary matches
- In step 4, the extra verification is only that each sender in the summary is the sender of some transaction in the block

Validity-proof-based

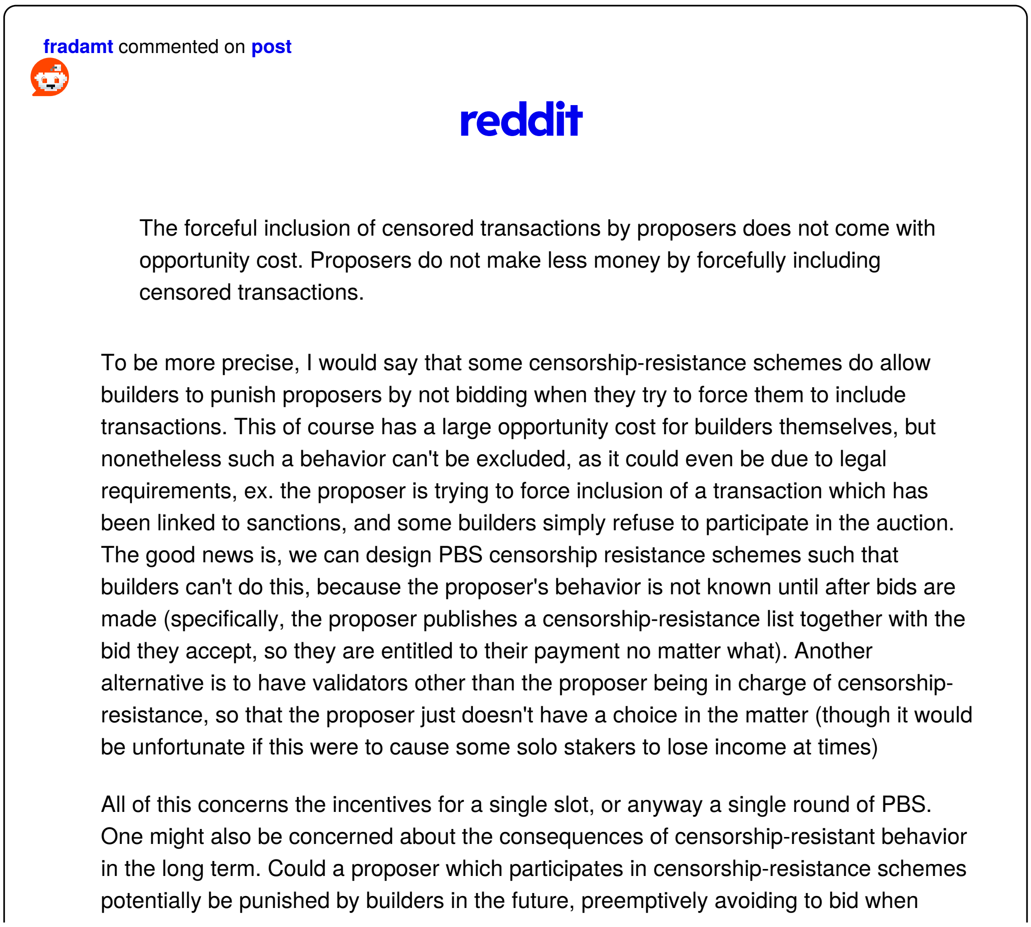
One way to do it without any interactive component would be for the proposer to sign a merkle commitment to the (sender, gaslimit) list, ordered by ascending gaslimit, ties broken by sender. The block would then contain:

- the commitment
- the first (sender, gaslimit) pair such that gaslimit > remaining gas
- a merkle branch proving that this (sender, gaslimit) pair is in fact the “next” pair in the list, after the ones which are already in the block

The builder would also have to include a bitfield marking which transactions are from crList, so that anyone can reconstruct the first part of the list, i.e. the (sender, gasLimit) pairs which precede the explicitly included one that is proven to be contained in the list.

(Doesn't quite work because you also need to prove inclusion of the whole list, and in the worst case you might need to include as much as twice as many (sender,gaslimit) pairs as there are in the list, so that a whole subtree can be reconstructed and a single merkle branch for it can be provided. What we can do instead is to have the merkle commitment be to a list of hashes of sublists. Say the ordered list of pairs is (p1,p2,p3...). We commit to (h1,h2,h3...) where hi = hash(p1,...,pi). Moreover, we commit to (p1,p2,p3...). With the second commitment we only prove the inclusion of the "next" pair, as before, and with the first commitment we prove the inclusion of the whole sublist which precedes it. Is there a simpler way which doesn't require hashing all these sublists?)

Builders punishing proposers for publishing crLists



fradamt commented on **post**

> The forceful inclusion of censored transactions by proposers does not come with opportunity cost. Proposers do not make less money by forcefully including censored transactions.

To be more precise, I would say that some censorship-resistance schemes do allow builders to punish proposers by not bidding when they try to force them to include transactions. This of course has a large opportunity cost for builders themselves, but nonetheless such a behavior can't be excluded, as it could even be due to legal requirements, ex. the proposer is trying to force inclusion of a transaction which has been linked to sanctions, and some builders simply refuse to participate in the auction. The good news is, we can design PBS censorship resistance schemes such that builders can't do this, because the proposer's behavior is not known until after bids are made (specifically, the proposer publishes a censorship-resistance list together with the bid they accept, so they are entitled to their payment no matter what). Another alternative is to have validators other than the proposer being in charge of censorship-resistance, so that the proposer just doesn't have a choice in the matter (though it would be unfortunate if this were to cause some solo stakers to lose income at times)

All of this concerns the incentives for a single slot, or anyway a single round of PBS. One might also be concerned about the consequences of censorship-resistant behavior in the long term. Could a proposer which participates in censorship-resistance schemes potentially be punished by builders in the future, preemptively avoiding to bid when

Key remaining issues

- lack of SSLE compatibility (besides the usual reasons for wanting SSLE, e.g. DoS protection, we just saw in the last section that SSLE is quite crucial to allowing proposers to safely participate in the crList scheme)
- lack of MEV smoothing compatibility

- having crLists be published by multiple validators, for added censorship-resistance. This would be especially useful if we are going to be doing DAS without sharding, which eliminates the added censorship-resistance given to data txs by the parallel shard proposers. It would also resolve the SSLE compatibility issue, because we could just not have the actual proposer be also a crList proposer.

The second and third problem are due to the same underlying cause:

- MEV smoothing requires a competitive builders market, which requires everyone to be on equal footing. Here, the proposer can eliminate all other builders from the competition by not publishing a crList, which prevents them from making a valid block. The issue is therefore availability of crList

- If we elected multiple crList proposers, tasked with publishing crLists before their assigned slot, we could not just require builders to include one crList for every elected proposer, because that would give any crList proposer grief the actual proposer. Again, the problem is availability of these multiple crLists.

In the next section we explore a way to resolve this availability issue, at first in a way which enables SSLE and smoothing compatibility, but which only utilizes one crList proposer

Using attestations to vote on availability

Inspired by the mechanism of MEV smoothing, we could attempt to use attestations to enforce a subjective condition which cannot be enforced by validity conditions. In this case, it would be as follows:

- Each slot has a crList proposer. The crList proposer at slot n-1 publishes their crList immediately after observing the block body, and anyway before some deadline (ex. 8s)

- Builders of slot n make sure their block is compliant with the crList, meaning there is no transaction which is excluded if remaining gas is less than its gas limit

- Proposer of slot n chooses a header, chosen builder publishes a body

- Attestations are made according to normal PBS specifications, except an attestor considers the body missing if it fails to comply with the published crList, if they have seen one before the deadline

Some relevant points:

- The risk is all on builders, and they can freely manage it by reducing their bids or not making any if they have reasons to believe that their body might be considered missing because of this extra attestation condition

- If a committee is corrupted and wants to make a builder lose money, they can do it with or without this attestation condition. Let's therefore assume a committee is honest. Moreover, assume that a builder has many nodes in the gossip network. For their body to be considered missing because of this condition, it has to be the case that >50% of the committee has received the crList before the deadline, but the builder has not. This is very unlikely if the builder is not being DDOSsed, in which case they could just avoid participating in the auction.

- If the winning builder makes a compliant block, there's no splitting of views at all: similarly to proposer view-merge, views are reconciled. Of course there's also no views-splitting if no crList is published. The only case in which it views can be split is if a crList is published and the builder which wins the auction hasn't seen it (or chooses not to include it, but this case is not important, as explained just below)

- Builders have no more power to split views than in normal PBS: publishing a body with no crList to split views is no different than just publishing the body close to the body publication deadline, which a builder can always do.

- The attestations to the body cannot be used as a guarantee of availability of crList. The attestors might not even see a crList at all, perhaps because it is not available and it is instead only known to the builder, in which case the only relevance of crList to the attestations comes from the validity conditions revolving around the summary

- MEV Smoothing compatibility: the crList proposer could not publish the list and just send it to some builders, but this does not eliminate other builders from the competition, because attestors only enforce the presence of a summary if the crList has been made public sufficiently early

Possible downsides:

- If the crList is missing, how much does the expected payoff for the proposer decrease? (because of risk management from builders)

Version with multiple crList proposers

- crList proposers publish crLists at slot n-1, immediately after observing the block body
- Committee members of slot n freeze their view of the published crLists at some point in advance of the header publication deadline for slot n (ex. 8s, or 10s). They keep a record of the crLists which they have seen, and filter out the ones containing non-eligible transactions (invalid or not paying a sufficient basefee)
- Builders of slot n make sure their block is compliant with all published crLists, in the usual sense: either the block does not have sufficient gas, or they need to be included
- Proposer of slot n chooses a header, chosen builder publishes a body
- Attestations are made according to normal PBS specifications, except an attestor considers the body missing if it fails to comply with all crLists in their frozen, filtered view of the crLists.

Issues:

- It could be that different subsets of the committee vote against the body because of different crLists. We don't have the same clearcut guarantees that we have for one crList, i.e. that if you have $n$ nodes randomly distributed in the gossip network, your chance of not seeing crList when 50% of the committee has seen it is $2^{-n}$ unless you are being targeted

Possible improvement:

- Incentive scheme for crList inclusion. One idea is to have crList proposers earn a reward only if inclusion is within one slot, and builders earn a smaller reward whenever they include a new crList, no matter how old. This way, every published crList which contains a censored transaction adds to the cost of continuing the censorship: the opportunity cost of not including all crLists which contain the transaction grows linearly in the number of crLists, which should be proportional to the number of slots in which the transaction has been censored. Therefore, the total cost of censorship for $n$ slots should grow quadratically in $n$.
- make this be very efficient optimistically, when there's no attempt to censor. We can have a main crList proposer which publishes it first, and whose list is meant to include all eligible txs seen at least 5s before, and everyone else just signs that list if they are ok with it. If not, they publish their own (i.e. augment it with their txs). Can make this into a whole process which tries to produce a maximal list? (or one up to the gas limit)

crList released with header, on-chain mempool

- proposer chooses a header normally
- with the header, they publish a crList + summary, which here is just a list of senders. Transactions opt-in to be eligible for crList, maybe by specifying an extra fee.
- as in two-slot PBS, a committee attests to the header, but here an attestation to it is also an attestation to the availability of the list + summary
- builder publishes the full block, containing the exec body referenced in the header AND all remaining, still valid transactions in the crList, not executed. The summary is also included, and a validity requirement is that all senders are represented in the block, either in the exec body or among these other “on-chain mempool” transactions. These pay per byte immediately.
- these transactions form a “longer-lasting crList” in the original sense: if there’s enough gas in the following blocks, they

must be executed. When that happens, the first (per byte) payment is refunded

Notes:

- Opting-in has essentially only one risk: if you get included in the on-chain mempool, you pay for data upfront but your tx could in theory never be executed. In practice, this should almost never happen unless you set a very conservative maxBasefee, with no room for a few full blocks in a row.
- We could require that the next block executes these transactions, though without any ordering constraints, and have crList be restricted to a small gas limit. We could also require crLists transactions to have a maxBasefee high enough to be sufficient even after one full block, so that no further special treatment is required.
- The reason we need this on-chain mempool mechanism is that publishing a "traditional crList" (which gets executed immediately) with the header would let MEV extraction happen in the crList (ex. any MEV which "happened" after the publication of the header by the builder can still be extracted through the list)
- crList + summary are not referenced in the header, so the attestations do not at first glance constitute free availability: the proposer is free to sign any crList they want, and the committee does not attest to the availability of a particular one. On the other hand, the proposer could put slashable collateral in a smart contract in order to give guarantees of not publishing two lists...

Incentivization

To best possible incentive system for crLists would be one where proposers are guaranteed a payment for each transaction which they include. Some notes about this:

- This is not possible when the list is published with the header, because the builder's block might include transactions which make crList transactions invalid, and the proposer has no way to know which ones. We can't a posteriori exclude transactions from the builder's block, but still force the builder to pay for the full PBS auction bid
- In the other schemes, we could in principle have guaranteed payments. We would need to require that crList transactions take precedence over other transactions from the same senders (not ordering precedence, inclusion precedence). Moreover, we would need to have some simple way for payments to be made even when the block is full and the transactions are not executed (such as inclusion of a small signed message just for the payment)

Data-txs and crLists

- basic scheme: ok if there is a sharded data mempool which allows the proposer to do sampling to check for availability. If something is only partially available and builders do not have time to reconstruct it, the proposer loses the slot.
- crList proposer, committee enforces list based on its timely availability: trickier. Now the whole committee has to do sampling for all the data txs in crList, even if they are not included. Also, what happens with partially available data which needs to be reconstructed?
- list published with bid, committee enforces availability of list when voting on the header: same as for the previous point

In general, the issue with any scheme which does not rely on the proposer to check availability out of self-interest is that requiring availability sampling from the whole committee means having higher consumption of "availability resources" than what we allow in a slot. If we allow X MBs of data, we might need to do sampling for X + Y, with the crList being capped at Y MBs of data. If crList has very low resource caps because it is just for censorship-resistance, that's probably ok. If it is meant to be used for fast confirmations, then heavily limiting its resources also heavily limits its usefulness (makes it expensive/unusual to get fast confirmations). A potential solution is to adjust the "slack multiplier" between target and max consumption of sharded data to account for the real max given by crList, but that means reducing the normally available supply of sharded data.

In either case, the proposer effectively gets some dedicated availability resources when sharded data is maxed out. They're not equivalent resources of course, since there's no immediate execution.

Notes:

- if you have to do the crList availability sampling at a different moment than the rest of the sampling, is it ok? If some data tx is not available at first but the builder includes it, you have to check it again as part of the 2D sampling of the block, but this is what you would have anyway done normally. It's still X+Y but you do X and Y in different moments
- to do availability sampling of crList data txs, you have to do it for single ones (no 2D construction).

Multidimensional EIP-1559 to the rescue

With multidimensional EIP 1559, this is less of an issue, if burst limit for sharded data >> target. Say it is 6x the target. We could restrict it to 5x, and still be able to process blocks where the limit is hit AND we need to guarantee the availability of a crList with sharded data up to target, or restrict it to 3x and be able to deal with availability of a crList with 3x target etc… Ideally, we set a lower burst limit which still has the property that it is hit only in a negligible percentage of blocks, and such that the remaining slack (real burst limit - burst limit we set, which is a limit on crList) is sufficient to provide fast confirmations (through crList and on-chain mempool) to all but a negligible percentage of transactions (i.e. such that only a negligible percentage of blocks has more includable shard data than what fits in crList). For example, if burst-limit/2 is sufficient for all but a negligible fraction of blocks, it would work as the limit we set

Not only it less of an issue in terms of availability resources, it is also much less of an issue in terms of proposer economics. Yes, proposers may at times control some extra availability resources, but crucially this would only be the case when the burst limit we set is reached.

Secondary auctions

- Proposer publishes a header containing one `main_exec_header` and some number of `secondary_exec_header`, up to `max_secondary_exec_bodies`
- All exec headers are published
- Attestations are similar to normal PBS, distinguishing between three cases, and we only add a condition to the second one:
  - No header
  - Header but no main body. This case includes a published main body which fails to include timely published secondary exec bodies from previous slots. This could for example mean all secondary exec bodies which have not yet had a chance to be included (because the chain has not progressed since their slot) and which have been seen (and passed an availability check on the data they reference) at the latest by the exec body publication deadline of their slot.
  - Both header and body

As should be evident, this is pretty similar to the two crList schemes which use attestations to vote on availability. There is however some important differences:

- crList proposers whose crList is not timely published risk at most losing a small reward, whereas secondary builders whose bodies are not included lose their bid. Therefore, they have a much greater incentive to publish timely and to spread the body widely
- the censorship-resistance provided by the secondary bodies is more bribe-resistant: to exclude secondary bodies, one needs to bribe entire committees rather than a few crList proposers. Moreover, if we added MEV smoothing to this, bribes would need to be very high in order to offset the fact that bidding for secondary bodies would inevitably go down if timely published secondary bodies are being made to pay without being included

Exec body partitioning

A natural question is how the exec body would be partitioned. Would we have n equal exec bodies, 1 main and $n-1$ secondaries? Would we instead have the main body totalling $1/2$ of the available gas, and the secondary bodies filling up the rest? Is this going to lead to more centralization because the only way to control the whole block is to be in control of choosing the whole sequence of bodies?

I think a solution which avoids having to decide on some arbitrary partitioning of the block, and also prevents further centralization vectors, is to allow the main body to take up the whole block, and to allow secondary bodies to be as small as they want, but capping their maximum number. This way, we get these properties:

- No further centralization vectors: if one exec body spanning a large portion of the block makes a higher payment than a sequence of small exec bodies, the proposer will choose this. Controlling the proposer does not allow for more extraction
- No damage to censorship-resistance, with the definition we have been using: if a builder can produce a body which fills up the whole block and does not leave space to secondary bodies, we don't consider this censorship. As with the crList scheme, EIP 1559 eventually forces blocks to not be full, and allows any censored transaction to be included through secondary bodies.
- Even a single transaction can be included this way, wrapped as a secondary body, if there aren't many secondary builders bidding at once

Sources of secondary bodies

Where will secondary bodies come from? Who is going to be taking the risk of including transactions which might already be included in previous bodies? Will there be secondary bodies only when there's censorship, making them easy targets?

Generally, what makes great secondary builders is having a constant stream of transactions which are not available to all builders, because these can be safely published. These frequent builders can also include censored transactions when there's a need for that (though how to incentivize them to do this is not straightforward. Vitalik showed there's at least a profitable strategy for including censored transactions when there are only two builders, but the multi-builder scenario needs further analysis). Potential secondary builders are therefore:

- Rollup operators! These could publish their data txs (ex. type 3 txs inthis proposal) in secondary bodies
- Public endpoint operators: for example, backrun-as-a-service operators, frontrunning protection services, off-chain matching engines, off-chain DEX aggregators etc...
- Wallets

I think it's hard to imagine a world in which users don't have a lot of options for how to submit their transactions, besides the public mempool. This is already a reality today, and I think it will be work more and more that way as MEV increases and off-chain competition for transactions intensifies, through incentives for end users.

EIP 1559 basefee updates

How do we update the basefee since the actual gas used is only known to the chain after the main body of the next slot is published?

One natural option would be to have the update happen in the main body of the following slot, after the execution of all previous secondary slots. The only downside is that the update cannot be known with certainty by users when they are submitting transactions. Most of the time, all secondary bodies should be published on time, and one can be almost sure that any accepted main body would have the expected basefee.

Splitting views by publishing partially available data

Say a secondary builders only publishes data on some subnets, enough to reconstruct the block and to pass some availability checks but not enough to pass all availability checks, so that some committee members would require this body to be included in the next main body and some wouldn't. Can this be a cheap way to split views?

Enabling widespread mempool-privacy and minimization of exploitative MEV

Until there's cryptographic primitives which allow for transactions' content to be private prior to inclusion, it should be expected that users will want to rely on trusted parties to provide frontrunning protection as a service (ex. Flashbots Protect). Moreover, transactions (and their orderflow/MEV) are valuable, and users will want to not only minimize exploitation but to maximize the value they get for their transactions, essentially by auctioning them off. Users can therefore greatly benefit from a competitive market for transactions, by foregoing the public mempool and sending their transactions to whoever provides the most value to them.

On the other hand, foregoing the public mempool, or generally limiting accessibility of one's transactions, comes with the trade-off of increasing time to inclusion: if there's many builders and I only send my transaction to one of them, I'll have to wait for them to win a block to get included. If I send it to 5 of them, I'll likely get included much faster but perhaps not by the builder which provides most value. Even worse, I am now trusting 5 builders to not frontrun me.

If the trade-off is bad enough, users will rather circulate their transactions widely, even at the cost of some value. If most users do that, "friendly" builders don't have any edge in the PBS auctions, and will lose more often that not to exploitative builders. We therefore should try to make it so being picky about who gets your transactions does not impact time to inclusion very much

With many bodies being includable per slot, we can enable many private mempools to coexist without much impact for the time-to-inclusion of their users