This article introduces Alternate PBS, a new PBS proposal forbased rollups. Alternate PBS allows MEV income generated on the based rollup to be captured by the rollup itself, as opposed to being captured through priority payments to the proposer of the sequencing chain. This addresses one of the biggest challenges faced by based rollups, as identified by based rollup guru Justin Drake at Devconnect, Istanbul 2023, a view which is already being reiterated. Alternate PBS also addresses another concern related to stealing based rollup blocks. Without careful construction, proposers of the sequencing chain can observe based rollup blocks built by unaligned rollup builders, and rebuild these blocks for themselves. This allows the sequencing chain proposer to receive the value of the built blocks without the cost of building (just copying).

# Motivation

Taiko is an example of a based ZK-rollup L2 blockchain. Taiko's L2 blocks are proposed and sequenced on Ethereum L1, with ZK-proven state transitions later posted to the L1. Based rollups, where L2 transactions are sequenced by an L1, naturally inherit many of the properties of Proposer-Builder Separation (PBS). The L2 builders compete for inclusion on the L1, which effectively acts as the block proposer/sequencer for the L2. Inclusion of L2 blocks in the L1 is controlled by the L1 proposers/block builders, who implicitly run an auction among L2 builders, with the winning bid guaranteeing inclusion in the L1. This provides strong guarantees of censorship resistance for the L2 blockchain, while allowing for a permissionless set of L2 block builders, and potentially many L2 block proposals per L1 block.

However, the competition for inclusion in the L1 block results in almost all L2 block-building revenue, including MEV, being paid to the L1 to guarantee orderly inclusion vs. other L2 builders. By allowing anyone to submit blocks to the L1 mempool, this also allows L1 builders and searchers to "steal" L2 blocks, by observing L2 block contents, and rebuilding them for themselves.

In this document, we propose Alternate PBS as an amendment to this design currently used in Taiko. Alternate PBS has the following properties:

1. L2 retains L2 block revenue.
2. L2 block stealing is mitigated.
3. Permissionless L2 block production.
4. L2 inherits L1 censorship resistance.
5. Multiple L2 block proposals possible per L1 block.

# Protocol Description

In addition to the permissionless roles of builders that currently exist in based rollups like Taiko (Taiko also requires ZK provers, although we generalize here to focus on based-rollups as a whole), we introduce a set of trusted proposers

. Access to the set of trusted proposers can be controlled via aProof-of-Governance protocol (alternatives such as PoS are also possible). These trusted proposers do not affect the safety or liveness of the protocol (provided certain precautions, as outlined later in this proposal, are taken).

For every L1 block, there is a designated L2 proposer from the trusted set with an exclusive right to propose an L2 block to the L1. This is enforced by an L1 contract deployed on behalf of the L2, which has knowledge of the proposer set.

Note:

This is an amendment to the current Taiko L1 smart contract. The contract on the L1 only needs to be aware of the set of trusted proposers. Another contract can enforce trusted proposer behaviour.

Trusted proposers are free to run a local instantiation of MEV-Boost, or build blocks themselves. If proposers are trusted for example, proposers can run the auction themselves. Otherwise delegated MEV-Boost using a trusted intermediary could be preferred.

In the case of based ZK-rollups, proposers must also provide a commitment from a prover, possibly the proposer themselves, to provide a proof for their proposed blocks to be valid. This commitment is typically a financial collateral that is returned when the relevant proof is provided. For increased decentralization of the proposer set, this could be outsourced through a prover market, as proposed already for Taiko.

In each L1 block, it is possible to override the exclusive right of the trusted proposer set by triggering an overridePermissions()

function. This allows any player calling overridePermissions()

to propose an L2 block to the sequencing based L2 sequencing contract instead of the trusted proposer for that L1 block. Calling overridePermissions()

requires the burning of a contract-specified _backdoorFee()

amount, which is a function of the number of times overridePermissions()

has been called in the last n

blocks (n

and _backdoorFee()

will be the subject of further analysis). _backdoorFee()

is paid to the L2 treasury.

Each L2 MEV-Boost auction must settle sufficiently long

before the proceeding L1 block is proposed to allow for propagation to the builder network, and inclusion in that L1 block. This notion of sufficiently long is at the discretion of the trusted proposer. The longer the auction is run typically corresponds to greater bids and auction revenue. This should be taken in consideration with the relevant risks of not propogating the L2 block(s) to the L1 mempool in a timely manner for L1-inclusion.

[

1710×449 59.3 KB

](https://collective.flashbots.net/uploads/default/original/2X/a/a8a7e2f62822f6003c78790f7747d353681f8b0d.jpeg)

)

Our proposed design only affects the rules of how L2 blocks can be sequenced in the L1. Although we would expect the trusted proposers to only sign-off on valid L2 blocks, this is not critical to the protocol. State verification happens after L2 blocks have been sequenced on the L1, as in typical based rollups.

However, depending on the trust placed in these proposers, a sequenced L2 block signed-off by the trusted set of proposers can be seen as verification that some/all of the transactions in a block will be executed. On the above scale, this moves our design slightly away from a pure based rollup. In a typical based rollup, sequencing a block on the L1 does not require any L2-related entities to sign-off on the block.

Our proposed design only affects the rules of how L2 blocks can be sequenced in the L1. Although we would expect the trusted proposers to only sign-off on valid L2 blocks, this is not critical to the protocol. State verification happens after L2 blocks have been sequenced on the L1, as in typical based rollups.

However, depending on the trust placed in these proposers, a sequenced L2 block signed-off by the trusted set of proposers can be seen as verification that some/all of the transactions in a block will be executed. On the above scale, this moves our design slightly away from a pure based rollup. In a typical based rollup, sequencing a block on the L1 does not require any L2-related entities to sign-off on the block.

## Permissionless PBS in the presence of a set of trusted players

Using a trusted set of players to enforce certain behaviours in a protocol should raise some concerns when trying to create a permissionless protocol. To ensure permissionless PBS for an L2, we need to ensure anyone can produce a block for the L2 using Alternate PBS. That is the function of the overridePermissions()

function. Any L2 builder willing to pay _backdoorFee()

in the L1 smart contract managing L2 block proposal can call overridePermissions().

[

image

1710×608 70.6 KB

](https://collective.flashbots.net/uploads/default/original/2X/4/4bda129157beb99e5914469e39d4f32182e7dbaf.jpeg)

An objective measure of the permissionlessness of _backdoorFee()

may be unrealistic. Thinking about its value on a scale, and what values it should have under various network conditions, will help to set the function appropriately.

An objective measure of the permissionlessness of _backdoorFee()

may be unrealistic. Thinking about its value on a scale, and what values it should have under various network conditions, will help to set the function appropriately.

With a parameter such as _backdoorFee(),

we have a lot of freedom to strike a desired balance between full permissionlessness (_backdoorFee() = 0

) and permissionless in name only (setting _backdoorFee() > feasible value of any L2 block).

Too high, and the trusted proposers can extract rents as builders become forced to use the trusted channel, while too low, and the permissionless channel will dominate, and most of the L2 revenue will be paid to the L1 (which this proposal is intended to address).

We envision a reactive _backdoorFee(),

which drops quickly when L2 blocks have not been proposed, but not so quickly so as to incentivize concurrent-block L1 strategies from L1 builders (e.g. censor L2 blocks in the current L1 block to drop _backdoorFee()

to 0 in the next L1 block, then extracting L2 value “for free” in that block).

Additionally, the level of trust in the trusted set will dictate the mechanisms we use to ensure they behave correctly. For example, the constraints on a proof-of-governance set will be significantly less than those required for a permissionless proof-of-stake set.

### Handling multiple L2 blocks per L1 block

Proposers, whether those elected from the trusted set, or those paying _backdoorFee(),

can propose arbitrarily many L2 blocks, within reason, per L1 block. Each L2 block must contain a prover commitment, meaning each block should be worth at least the cost of proving to the proposer (who is tasked with finding a prover) and the additional L1 cost associated with a larger L1 transaction. With respect to the proposer-run MEV-Boost auction, each additional L2 block therefore discounts the bid by an amount specified by the proposer.

To allow for multiple L2 blocks to be constructed per L1 block in an L2-specific MEV-Boost module, we introduce a proof_cost()

function. proof_cost()

is a monotonically increasing function which takes as input a number of L2 blocks required to be proven by an L2-defined deadline (as is defined currently for Taiko for example), and outputs a price for which the designated proposer/prover will provide those proofs.

Bids to MEV-Boost must be accompanied by a commitment to pay proof_cost()

for the number of L2 blocks attached to each bid. This allows for bids to be considered independently of the number of L2 blocks to be proposed, within some reasonable limit. For example, the L2 blocks’ contents must fit into the L1 block.

## Properties of Alternate PBS

Now let’s revisit the properties outlined at the beginning of the document, and how Alternate PBS inherits them.

1. L2 retains L2 block revenue:

By allowing a trusted set of L2-native entities to run an auction for block building, the L2 is able to retain block revenue within the L2 ecosystem. Either: 1. These proposers controlled by governance and trusted by the protocol to run an auction and report the winning bids correctly, in which case the revenue of the auction can be routed to the L2 treasury.

1. These proposers are not controlled by governance (~decentralized) and not trusted. In this case, L2 proposers capture proceeds for themselves, importantly still keeping the proceeds in the L2 ecosystem. If this were expected, _backdoorFee()

can no longer depend on the reported proceeds from the trusted auction.

1. These proposers controlled by governance and trusted by the protocol to run an auction and report the winning bids correctly, in which case the revenue of the auction can be routed to the L2 treasury.

2. These proposers are not controlled by governance (~decentralized) and not trusted. In this case, L2 proposers capture proceeds for themselves, importantly still keeping the proceeds in the L2 ecosystem. If this were expected, _backdoorFee()

can no longer depend on the reported proceeds from the trusted auction.

1. L2 block stealing is mitigated:

When blocks are proposed through the trusted proposer set, only one block proposer can proposed per block, preventing block stealing. However, when overridePermissions()

is called, block stealing is back on the menu for L1 builders and searchers.

1. Permissionless L2 block production:

By allowing anyone to override the trusted set by paying _backdoorFee()

(and some payment to the L1 proposer to insure inclusion), we are able to keep the entire protocol permissionless.

1. L2 inherits L1 censorship resistance:

As in 3, the trusted set can be overrode when censorship is suspected. This can even be enforced by removing _backdoorFee()

when multiple L1 blocks are added without containing any L2 blocks.

1. Multiple L2 block proposals possible per L1 block:

Addressed already in the Handling multiple L2 blocks per L1 block

section above).

## Conclusion

We propose a design framework for a permissionless PBS protocol for L2 blockchains. This framework facilitates permissionless block building and ensures network liveness, while also retaining revenue generated from L2 block production within the L2 itself. This is done by creating two complimentary channels for block production, one trusted and one permissionless. The intention is that the trusted channel will be responsible for producing most of the blocks, while the permissionless channel ensures that L2 blocks can always be force-included in the L1. In the presence of such a permissionless channel for block building, the trusted channel will be forced to perform as expected, or risk all block builders only using the permissionless channel, and L2 block revenue being paid entirely to L1 proposers and builders, as it is currently done.

For any questions related to R&D for PBS on L2s, please reach out to the team:

flashbots

- product management: @Tomasz
- design review: @dmarz , @Quintus .

Nethermind:

- engineering: Jorge Mederos, Paweł Nowosielski, Jinsuk Park.
- mechanism design: @CTra1n .
- architecture review: @swp0x0 .
- project management: Cameron Loo.