By Alex Watts (@ThogardPvP) - FastLane Labs

Abstract:

Currently, the benefit PBS might provide to on-chain execution is limited by the lack of a reliable link between a block builder and EVM execution. The most direct link - the block.coinbase address - is set by builders but can be arbitrarily assigned; its association with the builder is not

verified. Requiring a signature verification (or similar) to prove ownership of the block.coinbase as part of the consensus layer's block validation process will unlock novel DeFi primitives and turbocharge the benefits of PBS to the execution layer.

Example Use Cases:

The consensus layer signature verification of a block's block.coinbase unlocks many novel protocol designs and decentralizes many existing ones. Many of these designs would operate by having builders stake through a block.coinbase smart contract - think on-chain, atomic EigenLayer for builders. Validation of ownership of the block.coinbase allows for fully in-protocol slashing / reimbursements. Some examples:

- A variant of PEP-C handled entirely by builders with execution guarantees (as opposed to inclusion guarantees)
- LVR mitigation w/ Uniswap V4 hooks can be handled without accidental censorship by retroactively netting any "misordered" txs against the builder
- Execution Futures (partial blockspace auctions) that guarantee inclusion of a predefined UserOperation prior

to Block N (or your money back) can be purchased atomically

from a builder's contract.

- State proofs can be posted by builders and acted on optimistically, with atomic slashing / reimbursement if the proof is invalid (this is what FastLane is most interested in for our upcoming Ethereum → Polygon atomic settlement engine)

Unfortunately, most of these are quite tricky to explain without first defining in detail the exact design of the underlying smart contract. So for this post, I want to stick with something relatively simple:

- Trustless Multi-tx Flashloans

OK, actually that isn't simple at all… but it's the most straightforward to diagram out the full flow and fully define all core functions.

Why Can't We Do Multi-Tx Flashloans Now?

tldr: because of reorgs and relays.

First, let's define a "multi-tx flashloan" as a permissionless loan offered by the builder to a user in a transaction and that is repaid in a separate

transaction, later in the block

.

It's true that builders currently control the contents of a block. If a builder wanted to, they could set a block.coinbase address to a smart contract that they control that has some of their collateral in it for lending. This contract could define a borrow()

function, and put a modifier on it that stipulates require(block.coinbase == address(this));

.

dApps could then call a borrow()

function on that block.coinbase contract, and the builder could only include the borrowing tx in their block if they also have a handleRepayment()

tx that fully repays the borrowed amount. They would only sign the block if they're fully repaid by the end of it.

Unfortunately, this could end up costing the builder a lot. As you've probably already surmised, an attack vector is created by the fact that any builder can build a block with any block.coinbase address. Should a "reorg" leak the contents of the builder's block, another

builder could copy the original builder's block.coinbase address and put it as the block.coinbase for their own block and then execute the borrow()

function, which would pass the require(block.coinbase == address(this));

check. This other builder wouldn’t need to include the repayment tx.

Technically, a reorg isn’t even required - anyone can spoof a block.coinbase address, meaning anyone could steal the builder’s collateral without even sending a tx to the builder. This could be overcome by the builder’s lending contract requiring that all multi-tx flashloan requests can only be issued after

a preapproval tx signed by the builder in the same block. The builder would initially be the only entity to have the full preapproval tx, but it would be vulnerable to a reorg.

But even if reorgs were “solved” and the builder used a same-block preapproval system, the builder would also need to trust that their PBS relay wouldn’t leak the transactions. A single rogue employee at one of these block relays could easily drain the builder of their entire collateral by initiating the borrow()

transaction themselves, intercepting the “preapproval” tx from the relay, and then making their own block without the loan repayment.

Terminology

The EVM’s inability to see future state leads us to one conclusion: only by consensus layer validation of the builder’s ownership of block.coinbase (or a similar EVM-accessible address) can builders atomically offer or accept guarantees that are dependent on a future event and that have monetary value to adversarial actors.

We refer to a builder’s capacity to preemptively confirm the fulfillment of their counterparty’s precomittment as “Builder-Observed Preindemnification

” (BOP).

We refer to the aggregated locations of the data needed to determine if Builder-Observed Preindemnification (BOP) has been achieved as the “BOP Surface

.”

We refer to a builder’s capability to atomically, trustlessly, and permissionlessly offer services in exchange for precommitments by leveraging Builder-Observed Preindemnification (BOP) as “Builder-Offered Optimistic Precommitment Settlement

” (BOOPS).

Full coverage of the BOP Surface is a precondition for BOOPS. Note that properly understanding any sequence-dependent aspects of the BOP Surface is critical to enable cross-domain BOOPS - we’ll write more on that subject in a future post.

Multi-Tx Flashloan Flow

We revisit many of the same strategies as outlined above, only now it’s trustless because we know

that the contract at block.coinbase will only execute if the entirety of the block was built by that specific builder.

[

MultiTxFlashloan

1623×1234 218 KB

](https://ethresear.ch/uploads/default/original/2X/1/1206481e3b29d000b3d78e41a72e6394b13bd3a2.jpeg)

The key breakthrough here is that the builder is empowered with the knowledge that their smart contract will only execute when the builder has signed and validated the entire block. The entirety of the BOP Surface is covered, allowing for trustless BOOPS.

While it’s true that this means that the builder’s smart contract “trusts” the builder’s backend, this should not be an issue for builders. Handling the trust assumptions between a smart contract and a backend is an experience shared by every MEV searcher on every chain… and seeing as how most top builders are

current searchers (Rsync, Beaver), former searchers (Titan), or have former searchers on the team (Flashbots, bloXroute), I do not think this will slow them down.

Potential Implementations

There are two primary paths to implementation:

1. A consensus layer verification of a signature directly from the block.coinbase address.
2.

    - This is the most straightforward implementation, but it limits the capacity for block.coinbase to be a smart contract until EIP-5003 (or similar) is implemented.

3. This is the most straightforward implementation, but it limits the capacity for block.coinbase to be a smart contract until EIP-5003 (or similar) is implemented.

4.
    - Smart contracts wishing to use BOOPS can call a BOOPS aggregator contract that different builders can opt into, maintain collateral on, and post data to.

5. Smart contracts wishing to use BOOPS can call a BOOPS aggregator contract that different builders can opt into, maintain collateral on, and post data to.

6.
    - The downside to this approach is that builders can't easily integrate their own backend-specific logic or iterate quickly on novel, builder-specific BOOPS services. Furthermore, external contracts wishing to integrate the BOOPS services may need to redeploy if the aggregator contract itself is upgraded with additional functionality.

7. The downside to this approach is that builders can't easily integrate their own backend-specific logic or iterate quickly on novel, builder-specific BOOPS services. Furthermore, external contracts wishing to integrate the BOOPS services may need to redeploy if the aggregator contract itself is upgraded with additional functionality.

8.
    - In the future, we intend to add this functionality to the FastLane MEV contract on Polygon PoS. Validators can opt-in to using their accumulated MEV rewards as collateral for various BOP-enabled services.

9. In the future, we intend to add this functionality to the FastLane MEV contract on Polygon PoS. Validators can opt-in to using their accumulated MEV rewards as collateral for various BOP-enabled services.

10. A consensus layer 4337-like validation of a signed "BuilderOperation" by the smart contract at block.coinbase

11.
    - Consensus clients would need to engage the EVM (probably stateful) to call a validate()

function on the block.coinbase address with the BuilderOperation as the argument.

- Consensus clients would need to engage the EVM (probably stateful) to call a validate()

function on the block.coinbase address with the BuilderOperation as the argument.

-
    - While this may be more complex for the consensus layer to implement, it would be substantially easier for builders and dApps to work with.

- While this may be more complex for the consensus layer to implement, it would be substantially easier for builders and dApps to work with.

-
    - Aggregator contracts as described in path #1

would still work but may need additional, embedded safety checks

- Aggregator contracts as described in path #1

would still work but may need additional, embedded safety checks

-
    - Builders can converge on a shared BOOPS interface, but iterate more rapidly on their offerings.

- Builders can converge on a shared BOOPS interface, but iterate more rapidly on their offerings.

-
    - Builder value becomes about more than just CEX/DEX arb.

- Builder value becomes about more than just CEX/DEX arb.

-
    - dApps wishing to call block.coinbase directly may need additional safety checks of their own - they can't assume that the builder contract would have any dApp-aligned safety features.

- dApps wishing to call block.coinbase directly may need additional safety checks of their own - they can't assume that the builder contract would have any dApp-aligned safety features.

Other Thoughts

Assuming SGX can be fully trusted, a limited version of this should be possible on SUAVE for BOOPS that have a BOP surface entirely inside of the SGX.

I also want to state for the record that I tried to come up with less silly-sounding acronyms for these terms, but they are literally the only things I could think of that accurately describe the complexity of what's going on in less than a dozen words.

Please let me know if you can poke any holes in the logic or uncover any attack vectors on the multi-tx flashloan system as diagrammed above… This kind of design is always

an iterative process and I'm sure there's at least one or two things I've overlooked. Please break it so I can rebuild it.

Thanks for reading!