



MEV Manifesto

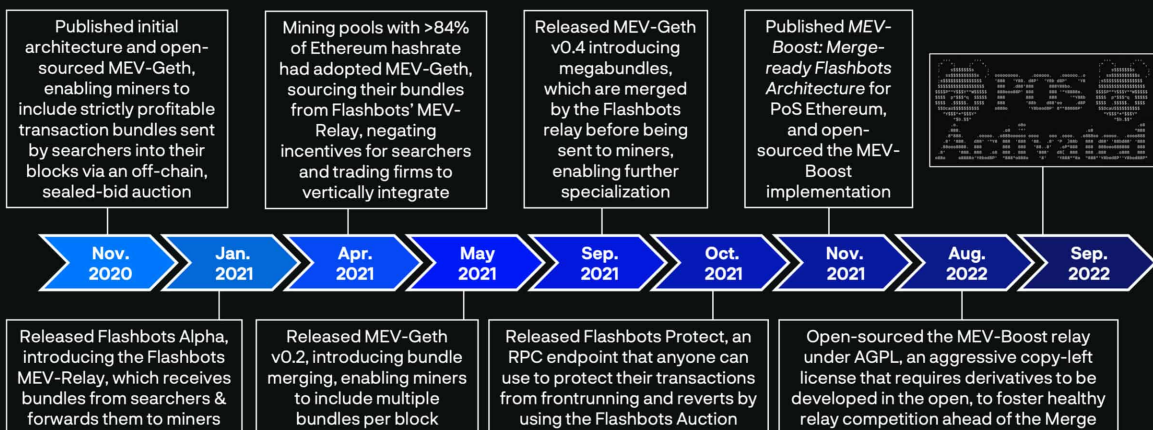
Premium · OCT 6TH, 2022



Jon Charbonneau



Flashbots History Milestones Over the Years



Source: Flashbots

Table of Contents

Introduction

- MEV Supply Chain

- Types of MEV

- Good vs. Bad MEV?

- MEV Utopia vs. Dystopia

Before the Merge

- Before Flashbots

- MEV-Geth

- Bundle Merging – Pre-Megabundles

- Bundle Merging – Megabundles

- Flashbots Auction & Privacy

The Merge

- Mechanics of the Merge

- Nakamoto Consensus Gasper

- Miner (Nakamoto Consensus) vs. Validator (Gasper) Censorship

Enshrined Proposer-Builder Separation

- Overview

- Two-slot Proposer-Builder Separation

MEV-Boost

- Overview

- Relays

- Flashbots Builder – New Profit-Maximizing Algorithm

- Choosing a Builder

- Relay Trust Assumptions & Liveness Risks

 - Malfunctioning Relays

 - Malicious Relays – Block Withholding Attack

- Relay & Builder Censorship

- Are Maximally-Decentralized Validators Necessary?

Conclusion

Introduction

MEV (Maximal Extractable Value) has been [loosely defined](#) as the value that block proposers can permissionlessly extract by reordering, censoring, or inserting transactions.

MEV is central to Ethereum block production. As a result, today's censorship issues are deeply intertwined with MEV research, and both will be discussed in this report. If you'd like my far more opinionated takes here, you can see my

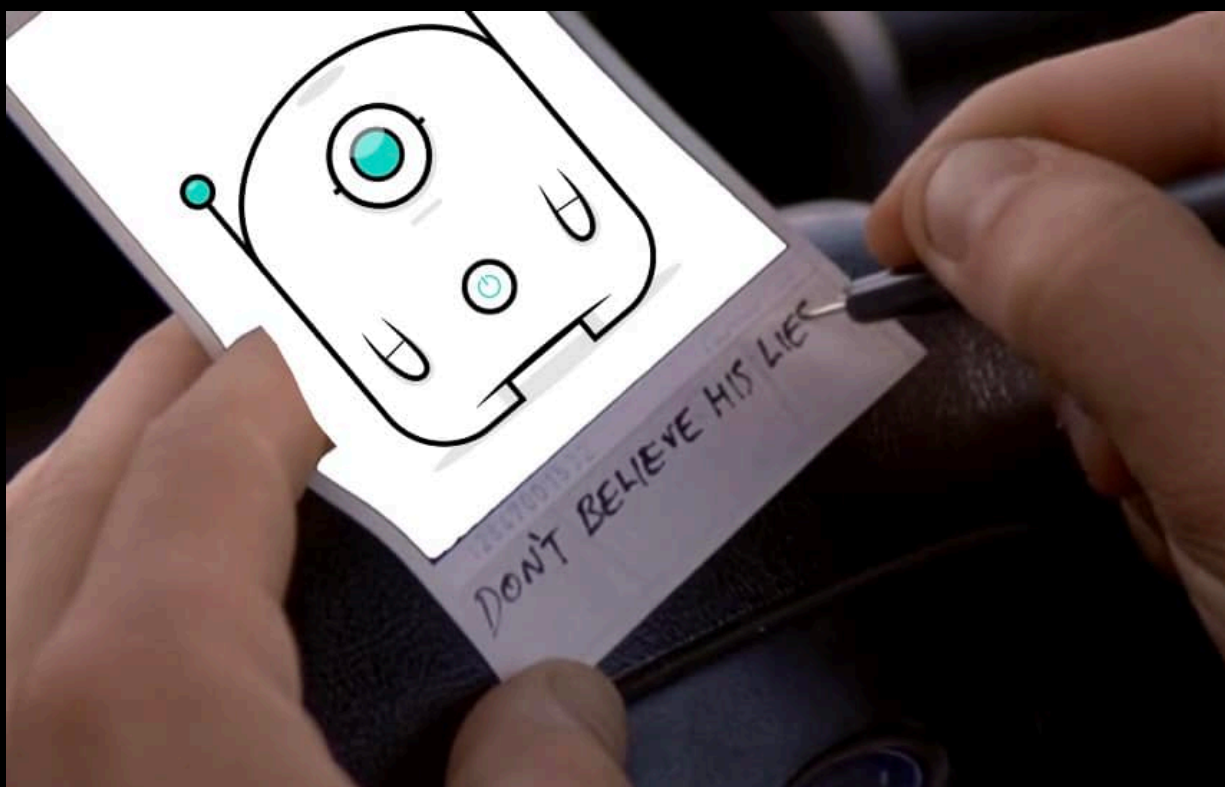
[recent article](#).

This report is more of a factual primer focusing on:

- **Overview** – MEV strategies and their supply chain.
- **Past** – How Ethereum initially reacted to the rise of MEV and built around it.
- **Present** – Ethereum block production changed drastically with the Merge, and MEV was central in these design decisions.
- **Future** – Ethereum faces many challenges (some old, and some new post-Merge) that will need to be addressed. I cover this a bit here then plan to elaborate in the future.

If you at any point require additional background on Ethereum’s mechanics, you can see my previous report [here](#). Some of its relevant sections have been incorporated and updated here as applicable.

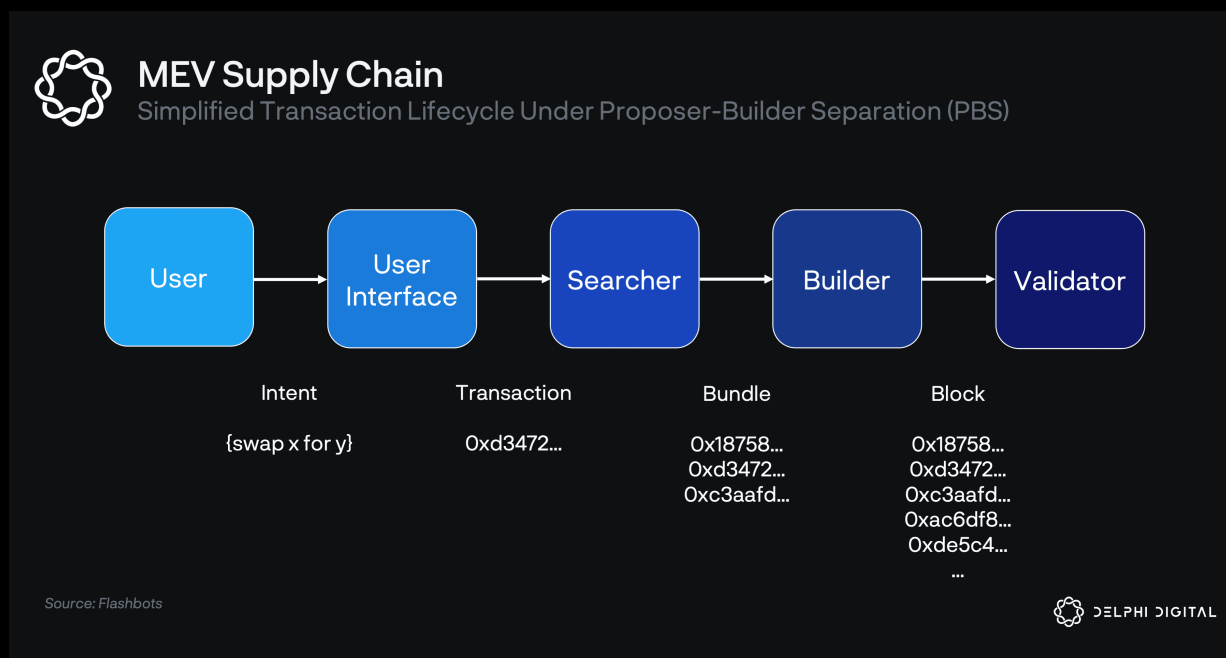
Just ignore all the times where I said that centralized block production is inevitable. Whoops. Lately, I’ve been thinking it might be possible to do better.



Here’s my recent [addendum to that piece on decentralized block building](#). It’s still unclear how this’ll shake out, but I think we’ve got a fighting chance at it.

MEV Supply Chain

The oversimplified Ethereum [MEV supply chain](#) looks something like this under PBS:



- **User** – Express intent to enact some state transition (e.g., swap ETH USDC).
- **User Interface** – UI allows the user to easily encode their intent into a transaction the blockchain can understand. This includes the whole app layer (wallet, smart contract, and dApp UI) working together to express user intent. This layer decides where to route user transactions (e.g., to the public mempool or private channels such as Flashbots Protect).
- **Searcher** – Run MEV strategies (e.g., arbitrage, liquidations) and submit “bundles” of their transaction preferences to builders.
- **Builder** – Aggregate transactions from various sources and construct a block (previously mining pool operators, but a distinct new role has been introduced post-Merge).
- **Validator (Proposer)** – Perform consensus duties. Proposers and builders have historically been the same logical entity by default, but PBS strips them apart.

Types of MEV

I’ll cover the major categories which encapsulate much of the value captured, but

note a long-tail exists. The largest forms are becoming progressively commoditized (i.e., most profits get bid to the block producer), while the long-tail retains a higher margin for searchers.

Atomic Arbitrage

Arbitrage = Buying and selling an asset in different markets (or in derivative forms) to take advantage of differing prices.

Atomic = Entire transaction sequence successfully executes together, or they all fail together (no partial execution).

Example – A large ETH buy was just executed on SushiSwap. ETH is now \$1,000 on Uniswap, but it's \$1,010 on SushiSwap. MEV searchers can submit bundles to atomically buy ETH on Uniswap and then sell it on SushiSwap until the arbitrage is closed. This benefits market efficiency without harming users, and it can provide riskless profits to extractors.

Statistical Arbitrage

Searchers can also take on risk to probabilistically capture MEV profits when conducting statistical arbitrage.

Example – ETH is trading at different prices on an Ethereum L1 DEX and a rollup DEX. Searchers could submit arbitrage transactions across these domains, but they run the risk that one trade executes while the other leg does not. They no longer capture a riskless profit from a sequence of transactions executing atomically. However, you can turn this specific cross-domain stat arb into (riskless) atomic arb if you control block production across domains.

Liquidity Sniping

This is another specific example of probabilistic MEV capture. Extractors who are willing and able to effectively manage risk across position sizing and timing will

outcompete here.

Liquidity sniping entails purchasing some asset immediately after it's listed in a DEX pair and liquidity is added. The sniper then offloads the position over an extended period of time, warehousing risk in the interim. In normal conditions, the asset price will rise significantly from its initial listing price. However, this isn't guaranteed – the sniper is taking inventory risk as they offload it.



Frontrunning

Trades create a price impact – that's why you set some slippage tolerance of where you're willing to be filled. However, this opens up the ability to frontrun your trade.

Example – I want to buy 10 ETH for \$10,000, but I set my slippage tolerance to 2%. Searchers can see this in the mempool, then swoop in and take that liquidity in front of me, causing me to execute at my worst price and getting only 9.8 ETH.

Generalized Frontrunning

This is the attack popularized in [Ethereum is a Dark Forest](#). Generalized frontrunners can scan the public mempool for any transaction, simulate it with

their own wallet address swapped in, then frontrun the original transaction with their own if it's profitable. This reaches far beyond just simple DEX trading.

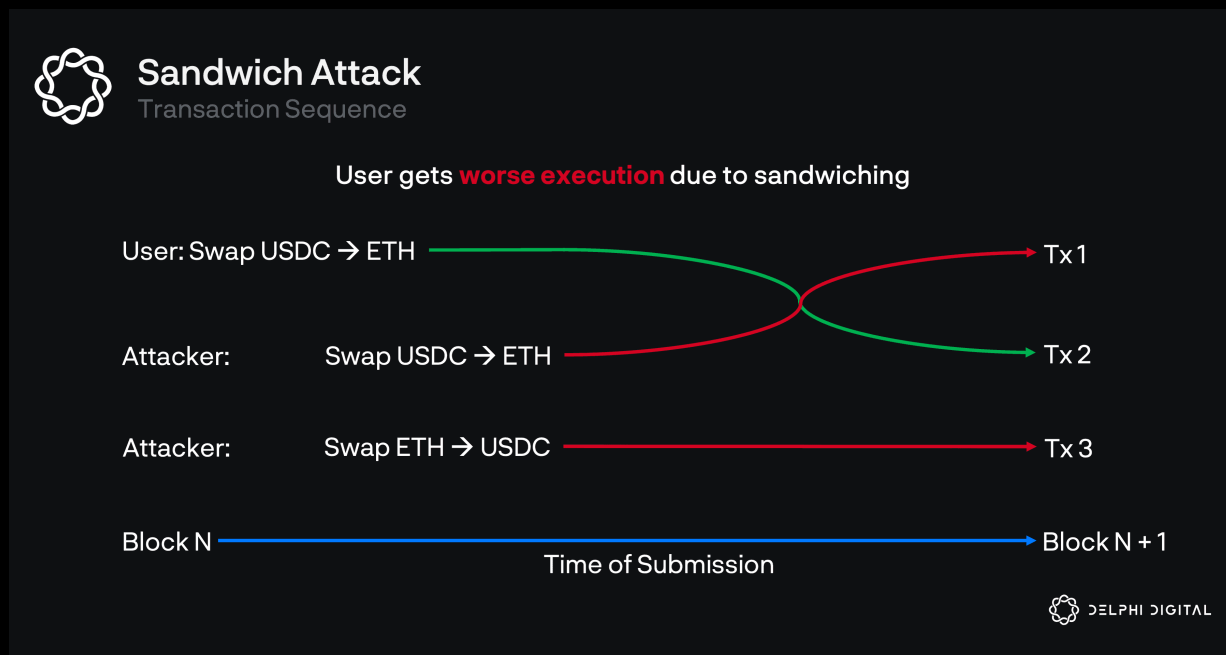
Backrunning

Intuitively, it's the opposite of frontrunning.

Example – After I executed that trade earlier of \$10,000 for 9.8 ETH, the price of ETH on that exchange is now ~\$1,020. However, the global market price of ETH is still sitting at \$1,000. Someone can profit by closing this gap – selling ETH right behind me at a premium until the gap has been closed back to \$1,000.

Frontrun + Backrun = Sandwich

Upon seeing a large trade order in the mempool, searchers could submit a bundle including (1) their frontrun tx, (2) the target tx, then (3) a backrun tx. They scoop up liquidity, allow the target to push up the price, then immediately sell it at a markup.





Liquidations

Open market participants are needed to liquidate under-collateralized loans. This helps to ensure a properly functioning DeFi market.

Example – User borrowed USDC against their ETH on Aave, but the price of ETH has since fallen. The borrower is now under-collateralized (i.e., USDC value is approaching the value of their ETH collateral, and is below the protocol's collateral requirements). Anyone is able to liquidate this loan – they can pay off the USDC loan as quickly as possible, claim the ETH collateral, and sell it. Well-designed protocols can auction off the right to liquidate this position. The loan is

closed, keeping the protocol solvent, and the liquidator earns a profit.

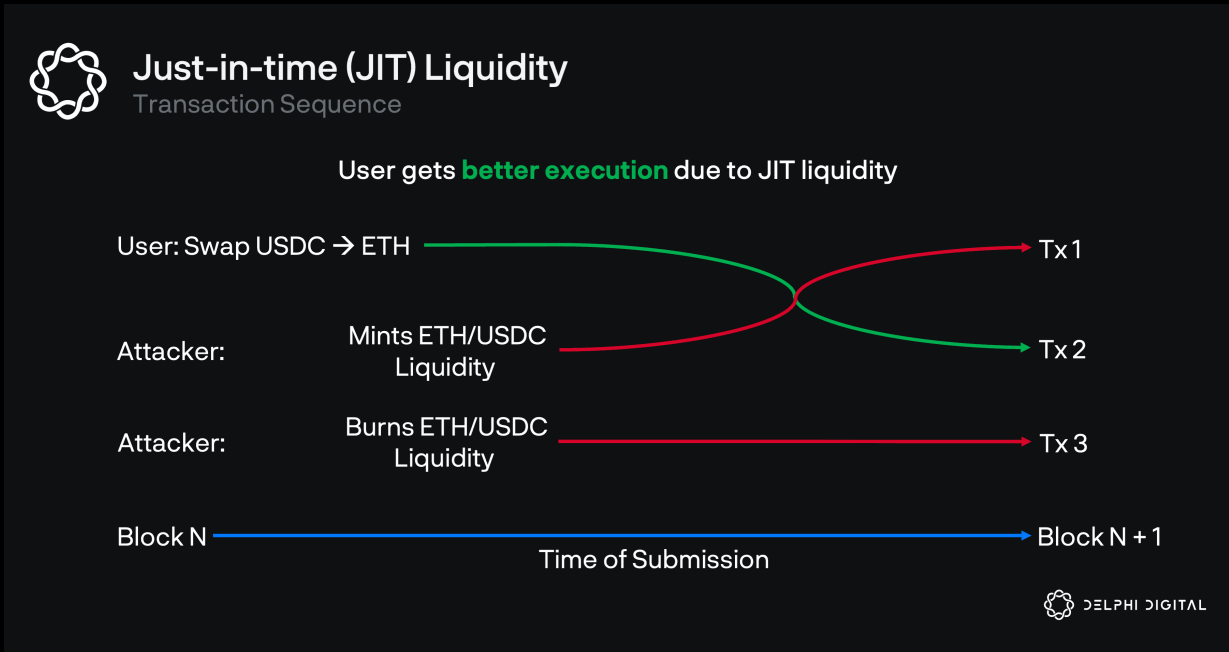


Just-in-Time ("JIT") Liquidity

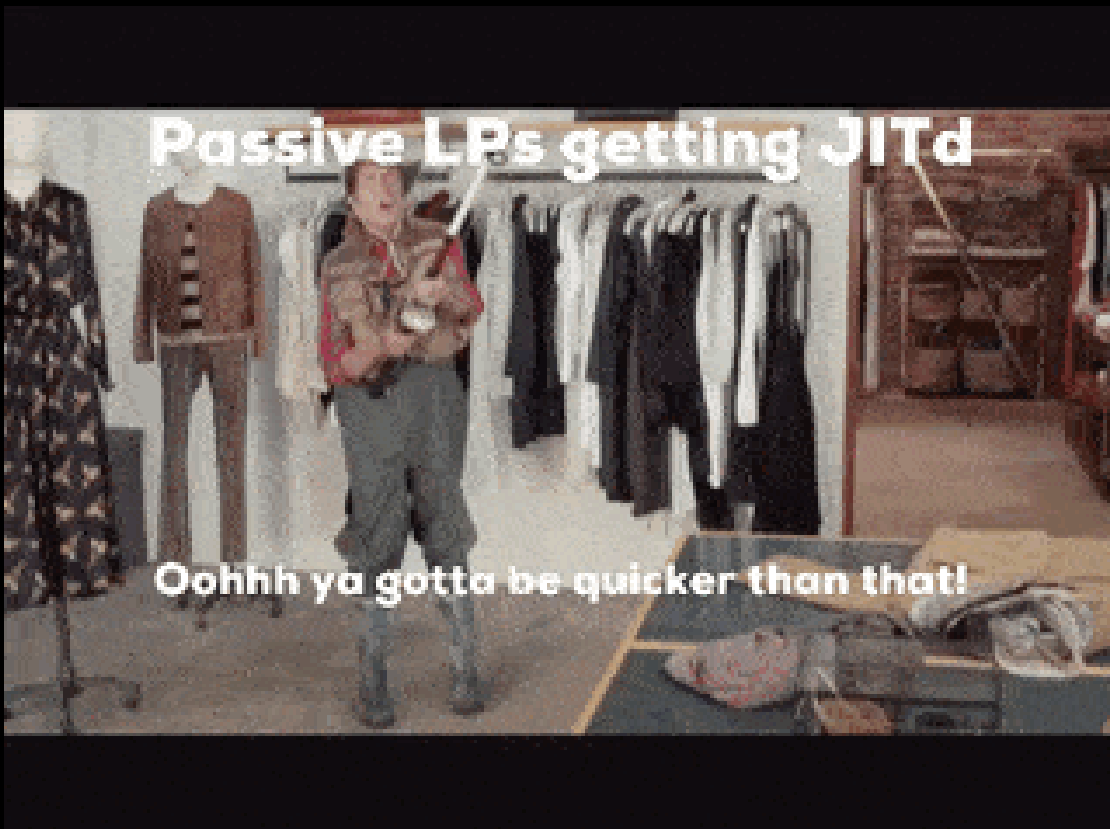
TLDR if you're unfamiliar with Uniswap v3's concentrated liquidity – LPs provide liquidity over a specific price range rather than in a pool altogether. LP positions are represented as NFTs. Range liquidity is far more capital efficient – deeper liquidity – less price impact from trades – less MEV extracted from users. In fact, JIT liquidity is one such MEV strategy which actually explicitly benefits user execution.

Rather than sandwiching a user with trades that push the price up then sell it

after for a profit, an LP could “sandwich” the target trade by minting and burning narrow ranges of deep liquidity immediately before and after it. This allows the active LP to jump in at the last second and capture the trade’s fees.



This reduces the fees gained by passive LPs, but it’s explicitly beneficial to user transactions. Users get better execution with lower slippage. Some argue that disincentivizing passive LPs leads to less liquidity.



However, note this is incredibly uncommon, with [<1% of all of Uniswap v3's volume being matched against any JIT liquidity even partially](#). This only occurs when a trade would have otherwise received incredibly poor execution relative to market prices. The “attacker” realizes that they can offer an even better price, and it's still worth it for them to capture the fees.

We could see this more frequently though, as Uniswap v3 (or similar mechanisms) move to cheaper fee environments (e.g., rollups). This JIT “attack” is pretty gas-inefficient, so reducing the cost to conduct it could make it a bit more prevalent. There are possibly some ways to prevent it, but the question is should we even try to? Dan Robinson gave a great rant on the topic [at SBC here](#), and he argued that we shouldn't try to prevent it even if we can.

One interesting area to explore here would be new mechanisms that allow LPs to compete on price movements. Currently, the price that an LP offers has nothing to do with whether or not their JIT transaction gets included in the block. Ideally, the LP that offers the most price improvement (quotes the best price) should outcompete other LPs and sandwichers. This would incentivize more LP participation and increase trader welfare even further. Uniswap recently discussed this idea in a [JIT blog post](#).

TWAP (Time-Weighted Average Price) Oracle Manipulation

Multi-block MEV is exacerbated by Ethereum's shift to PoS. PoW miners can't know the next block producer (unless conducting selfish mining), but validators are now chosen ahead of time.

At the end of epoch $N-1$, the validators for epoch N are all known (1 epoch = 32 slots = 6.4 minutes). The block producers for epoch $N+1$ are also generally known, but are subject to change in rare circumstances of validator balance changes (e.g., due to slashing).

A validator who knows they will control subsequent blocks (B_1 and B_2) can trivially manipulate TWAP price oracles to their benefit. Simple TWAPs such as Uniswap v2 just record the asset price at the end of each block for N blocks, sum the asset prices, then divide by N .

The validator could manipulate oracle prices at the end of B_1 and exploit the resulting MEV opportunities in B_2 without competition. For example, many lending protocols rely on price oracles from DEXs such as Uniswap to provide them the value of on-chain assets. This can be used to determine the health of collateralized positions and dictate when liquidations can occur.

A more detailed analysis of this topic can be found [here](#).

Good vs. Bad MEV?

MEV is often framed as bad, or at least certain types of MEV are framed as bad (e.g., frontrunning and sandwiches) vs. others being good (e.g., arbitrage and liquidations). However, this framing can often be detrimental and simplistic in my view.



JIT liquidity was one such example – it’s an MEV “attack” which can hurt passive LPs, but directly benefits users. What’s the right tradeoff to optimize for?

Sandwich attacks can actually be another interesting one. A [recent research paper](#) showed that sandwichers can actually improve network welfare and improve transaction routing.

The reality is we're still learning a hell of a lot about MEV and its implications. Gut reactions may not be the most appropriate courses of action. That doesn't mean we should have bad designs which screw users. Rather, we should closely consider the externalities of these design decisions. Generally be wary of any silver bullets which claim to "solve" MEV.




Source: H/T Fred for this meme. Check out more dank [MEV memes on the Flashbots Collective](#) [here](#). Optionally read about MEV stuff while you're there too.

MEV Utopia vs. Dystopia

Armed with a basic understanding, we can now lay out what we're trying to

achieve here. The below are some proposed goals of Flashbots which I generally like:




MEV Utopia vs. Dystopia

Some Potential Goals

	MEV Utopia	MEV Dystopia
TLDR	Permissionless, Efficient, Socialized	Regulated, Systemic Instability, Censorship
Validator / Bridge	Neutral	Financial Intermediaries
Liquidity	Passive	Just In Time
Externalities	Gas Efficiency / Privacy	Co-location / Spam / Asics
Competitive Edge	Creativity	Capital / Exclusive Access / Licenses
Bot Operators	Individuals	Billion Dollar Funds
Value Accrual	Users & Network	Bots & Miners
Consensus	Moves Forward	Moves Backwards

Source: Flashbots



MEV is created either directly or indirectly by user transactions. Ideally, we create systems where users can benefit from the MEV they generate themselves. This can be achieved by exposing less unnecessary MEV to be extracted in the first place, or via direct cashbacks/fee rebates to users. Rebates ≈ better execution.

To the extent MEV leaks out from users, it's beneficial to direct this toward validators to fund the chain's security budget. To keep the validator set decentralized, this MEV must be easily accessible in a permissionless manner. Additionally, the potential to capture MEV should scale linearly with percent of stake-weight (i.e., n% of stake should receive approximately n% of all MEV captured). Superlinear returns to stake-weight would pose a centralization vector.

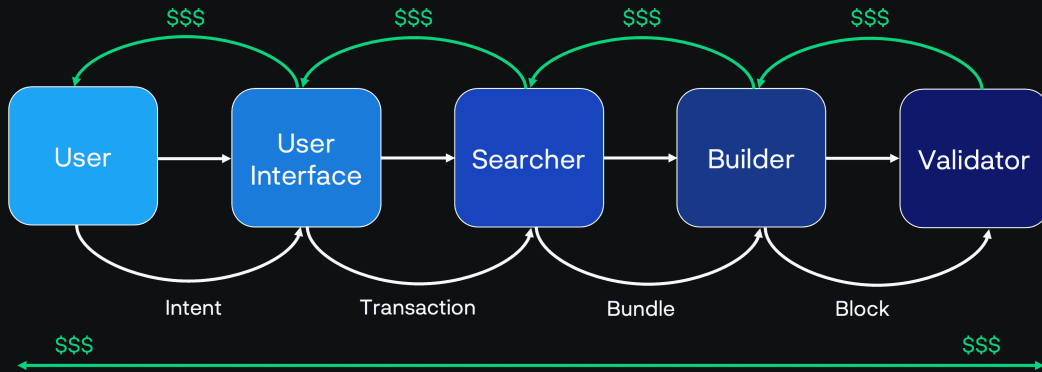
Ideally, independent entities fulfill each role and compete in an open marketplace. Rent extraction in the middle of the chain should be minimal, with value accruing to the edges:



MEV Utopia

Decentralized Block Building

Decentralized Block Building



Source: Flashbots



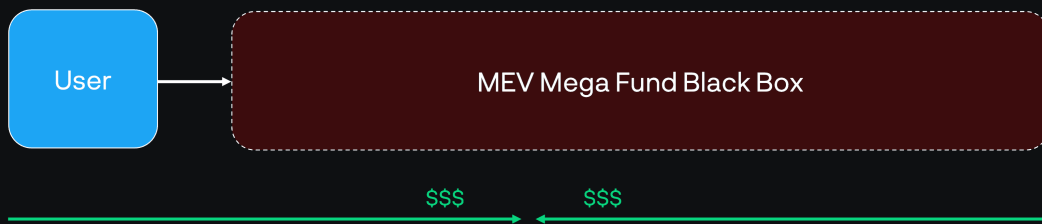
We want to avoid high-resourced gatekeepers becoming entrenched and extracting rent in the middle of the supply chain (e.g., to a dominant single block builder). This could take the form of a single large entity vertically integrating throughout the supply chain. They engage in anti-competitive behavior and increase barriers to entry for new participants:



MEV Dystopia

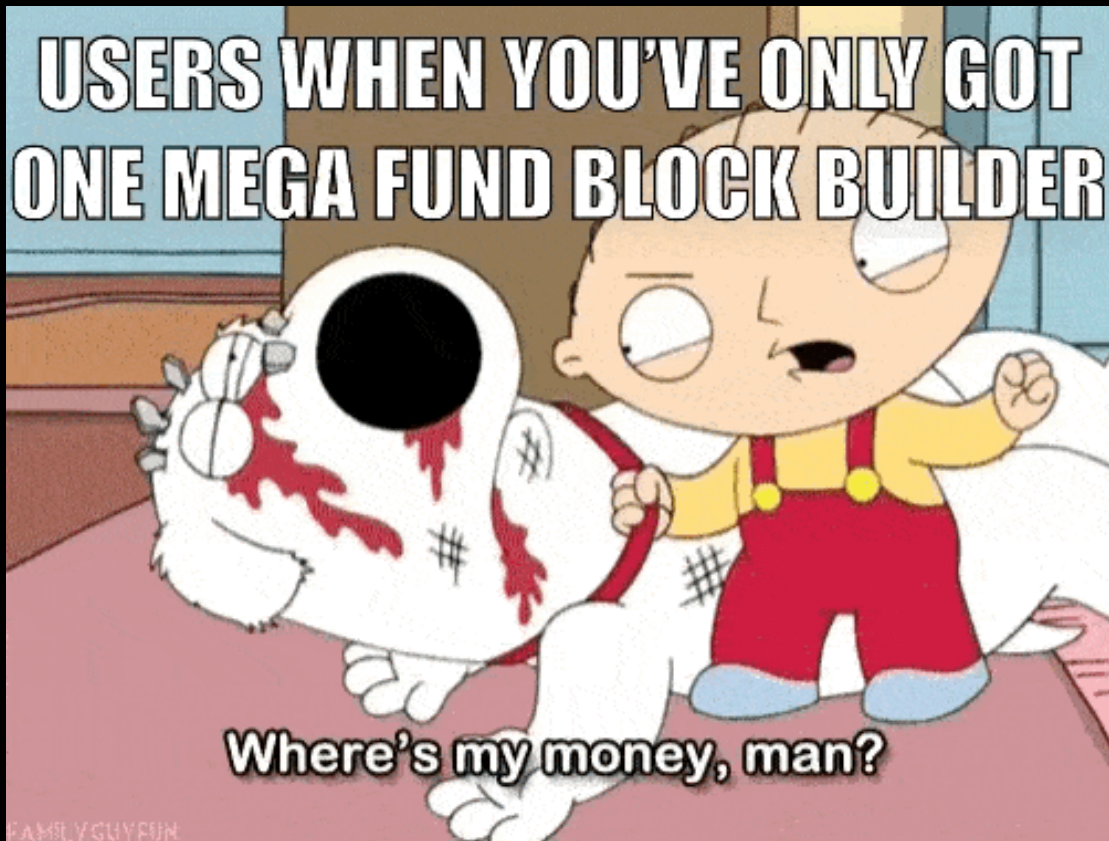
Centralized Block Building

Centralized Block Building



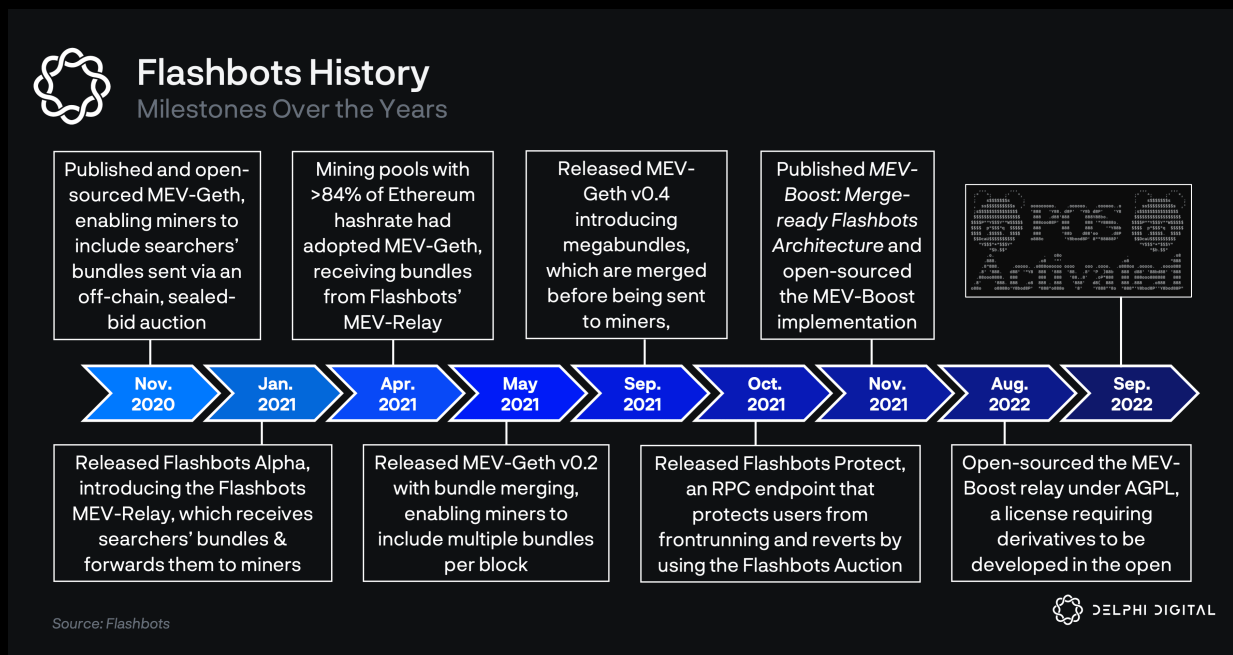
Source: Flashbots





Before the Merge

The system here has changed significantly over the years, but a brief history lesson provides necessary context for the current design and its goals.



Before Flashbots

Miners ran clients (primarily Go Ethereum a.k.a. Geth) which produced blocks with transactions ordered by gas price. Searchers would conduct “Priority Gas Auctions” (PGAs) to fight for blockspace – sending transactions to the public mempool and bidding higher back and forth in hopes of their transactions landing. Failed transactions would still land on-chain, driving up gas prices and wasting blockspace.

Searchers were unable to express preferences for bundles of transactions which could execute atomically. For example, to sandwich attack you could try to submit:

1. Frontrun transaction with a gas price slightly above the target transaction
2. Backrun transaction with a slightly lower gas price, then hope they all land together

Mining pools were beginning to strike exclusive deals with trading firms to extract MEV. This risked vertical integration of the block production supply chain.

MEV-Geth

Flashbots developed MEV-Geth (fork of Go Ethereum) to give miners an easy way to aggregate MEV bids, removing the need to vertically integrate. Providing the ability for searchers to clearly express complicated preferences via bundles then enabled an efficient auction, cutting down on the negative externalities of PGAs.

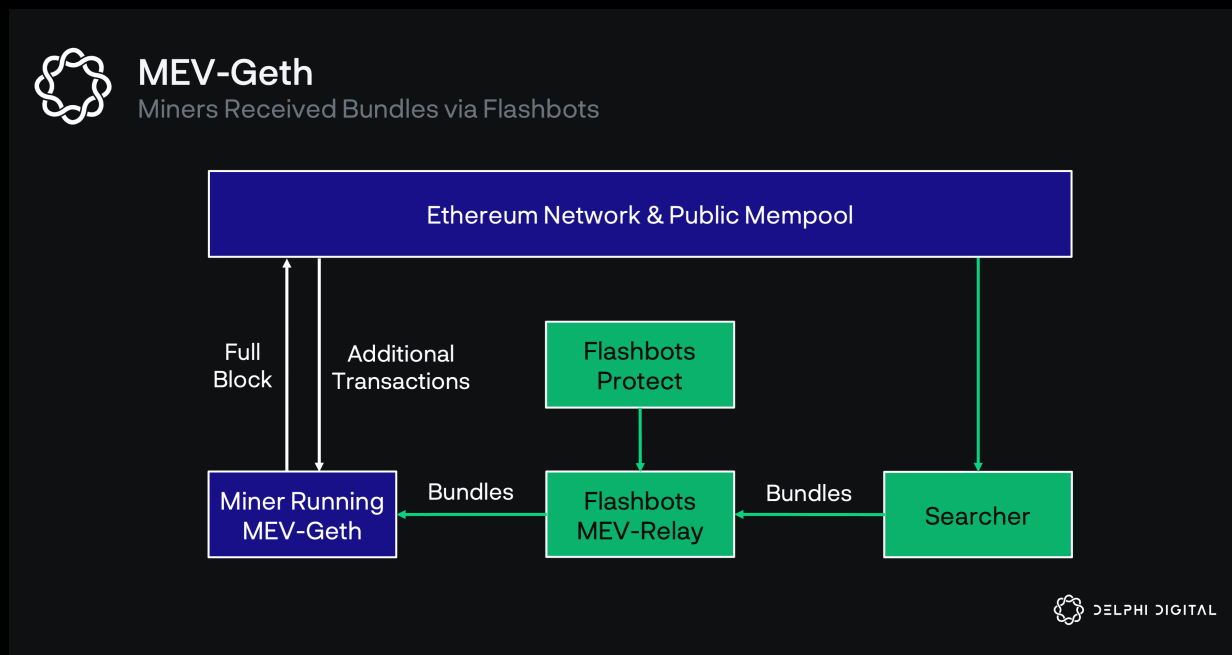
The Flashbots Auction uses an off-chain first-price sealed-bid auction. Searchers bid for their bundles to be included, and failed bundles never land on-chain or pay fees. Bundles can include:

- Preference over one or several transactions (can be searcher’s transactions and/or other users’ transactions in the mempool)
- Searcher’s bid – this can be paid to miners via a smart contract call to `block.coinbase.transfer()` or via priority fees/tips with their submitted

bundles

The process looks like this:

1. Searchers send bundles to Flashbots' MEV-Relay (relay.flashbots.net).
2. MEV-Relay simulates bundles and then passes them along to all whitelisted miners running MEV-Geth. MEV-Relay also provides DOS protection – searchers don't pay for failed bids, so anyone can spam with invalid bundles.
3. MEV-Geth selects the most profitable combination of bundles, then fills out the rest of the block with other mempool transactions.
4. MEV-Geth compares the block, including these bundles vs. a vanilla block with no bundles, then selects the more profitable one.
5. Mining pool operator sends the header along to the individual winning miner to attest to it with PoW, giving it weight in the fork choice rule.



Searchers sign the bundles with their private key, enabling them to build a reputation over time. This isn't the primary private key used for executing transactions, and it doesn't store funds. It's just used for identity. Better reputations give searchers higher-priority delivery of their bundles. Searcher reputation allows the MEV-Relay to better infer which bundles are likely to be profitable, even without being able to run a full simulation of all transactions.



Flashbots' MEV-Relay was dominant. However, others (e.g., bloXroute and Ethermine) also ran smaller relays.

Several trust assumptions exist in this supply chain, including:

	Miners	MEV-Relay	Searchers
Miners:	-	Miners trust relays to send them profitable bundles and not DOS them	Miners do not need to trust searchers
MEV-Relay:	Relays trust miners not to reorder, censor, or steal the forwarded bundles ⁽¹⁾	-	Relays trust searchers not to DOS them ⁽¹⁾
Searchers:	Searchers trust miners not to reorder, censor, or steal their bundles after being forwarded by the relay	Searchers trust relays not to reorder, censor, or steal their bundles	-

1) If abuse is suspected, the relay can cut off miners' access/blacklist searchers respectively

Bundle Merging – Pre-Megabundles

Bundle merging is the process by which some aggregator finds the ordering and number of bundles that optimize profitability.

MEV-Geth's alpha-v0.1 was pretty bare bones – only one bundle could be included in a given block. Alpha-v0.2 introduced multiple bundles per block. The bundle merging process began, giving miners control over the bundles they wanted to include.

MEV-Geth managed this process for miners by parallelizing it and choosing the most profitable combination of bundles:

1. MEV-Relay forwards bundles to miners running MEV-Geth
2. MEV-Geth simulates all bundles alone and figures out their gas prices
3. MEV-Geth places the highest gas price bundle (B_1) at the top of the block
4. MEV-Geth tries to merge additional bundles by simulating multiple bundles together one after another, looking for conflicts:
 - Simulate the second highest gas price bundle (B_2) placed after B_1
 - If B_2 reverts (e.g., targets same opportunity as B_1), then B_2 is discarded
 - Repeat simulation for B_3, B_4, \dots, B_N

Bundle merging is a very computationally-demanding process. Each bundle you attempt to merge requires an additional parallel computing process. For example, trying to merge up to 3 bundles requires 4 parallel computing processes:

- One creates a block with 0 bundles
- One creates a block with 1 bundle
- One creates a block with 2 bundles
- One creates a block with 3 bundles

At the end, the most profitable block of the 4 is selected. As a result, the median miner would only merge up to ~3 bundles per block, even though blocks frequently had >3 profitable bundles.

Note that bundles are actually rank-ordered here by gas price and not overall bundle profitability. It's not optimal, but this was chosen because it's computationally simpler to do. This put gas-intensive MEV (e.g., liquidations) at a disadvantage compared to less gas-intensive MEV (e.g., arbitrage).

Consider the following example:

	Gas Used		Gwei		Miner Payment
Bundle 1	100,000	x	10,000	=	1 ETH
Bundle 2	1,000,000	x	9,900	=	9.9 ETH

B_1 would win the Flashbots Auction because it bid a higher gas price (10,000 gwei), despite the fact that B_2 is more profitable overall (9.9 ETH vs. 1 ETH).

Bundle Merging – Megabundles

MEV-Geth’s alpha-v0.4 introduced megabundles – bundles which are merged before they’re sent to miners. This shifted the merging task upstream from miners allowing more specialized actors such as Flashbots’ MEV-Relay to further optimize their infrastructure and more efficiently merge bundles, resulting in higher profits.

This also allowed faster product iteration. Previously, updates would’ve required coordinating with the large majority of miners who were running MEV-Geth. Instead, Flashbots could now simply update the MEV-Relay’s megabundle code.

Megabundles moved a step closer toward outsourcing full-block building under PBS, but it still had limitations. Megabundles didn’t form entire blocks, and they still offered no privacy to searchers (relays and miners could still MEV-steal).



Flashbots Auction & Privacy

Flashbots has researched both cryptographic and crypto-economic proposals to separate block building and proposing (proposer-builder separation, a.k.a. PBS).

To ensure that all block proposers have access to the most profitable builder blocks, their ability to MEV-steal from builders must be removed by adding privacy to this relationship. Otherwise, builders would only provide blocks to large entities that they could trust and monitor (as was the case with PoW mining pool operators, and would instead be limited to Coinbase, Kraken, etc. in PoS).

Proposals to address this have included:

1. **Enshrined PBS (future Ethereum upgrade)** – A permissionless version of PBS built into the Ethereum protocol. First, builders provide only their block headers to the proposer. Second, the proposer accepts the header with the highest associated bid. Third, the full block body is only revealed after the proposer has committed to its header. If the proposer then attempts to MEV-steal and propose an alternative block, they can be slashed for double-signing.
2. **MEV-Boost (live)** – Interim version of PBS which still includes some trust assumptions in the stack, but removes the need for builders to trust proposers.
3. **MEV-SGX** – Uses trusted hardware to keep block contents private. Builders create blocks, encrypt them, and send them along with decrypted block headers to miners. If a miner finds a valid PoW solution, this can be input into their SGX along with the encrypted block to produce a decrypted and attested block. This design could be used similarly in PoS with some alterations, but more work is needed.

The Merge



Mechanics of the Merge

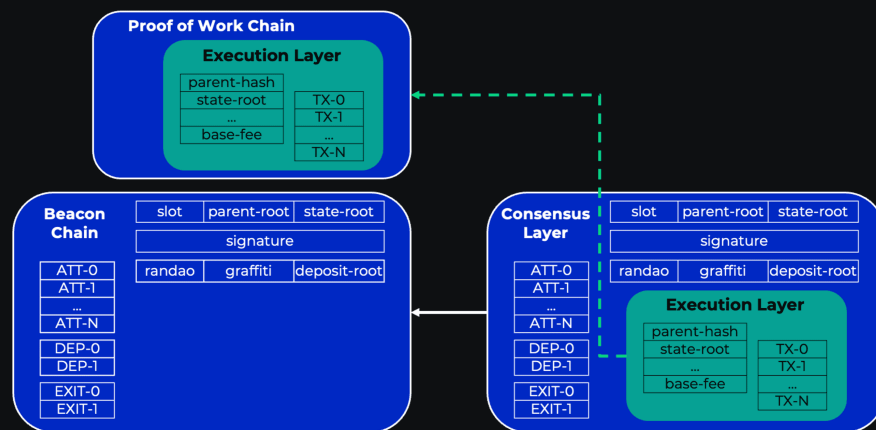
Prior to the Merge, full nodes ran one monolithic client (e.g., Geth) that handled both:

- **Execution** – Execute every transaction in a block to ensure validity. Take the pre-state run transactions transition to the post-state
- **Consensus** – Verify you're on the heaviest (highest PoW) chain with the most work done (i.e., Nakamoto Consensus)

At the Merge, the existing execution layer merged into the Beacon Chain consensus layer:



The Merge Underlying Mechanics



Source: Tim Beiko, Ethereum Foundation



Full nodes now run two separate clients under the hood:

- **Execution client (f.k.a. "Eth1 client")** – Eth 1.0 clients continue to handle execution. They process blocks, maintain mempools, and manage and sync state. The PoW stuff got ripped out.
- **Consensus client (f.k.a. "Eth2 client")** – Beacon Chain clients continue to handle consensus. They track the chain's head, gossip and attest to blocks, and receive validator rewards.

Clients receive blocks, execution clients run the transactions, then consensus clients follow that chain if everything checks out. All execution and consensus client pairs are interoperable.

Nakamoto Consensus Gasper

Ethereum swapped out Nakamoto Consensus for [Gasper](#) = Casper FFG (the finality tool) + LMD GHOST (the fork-choice rule) at the Merge.

Safety favoring algorithms (e.g., Tendermint) halt when they fail to receive the requisite number of votes ($\frac{2}{3}$ of the validator set here). Gasper is a liveness-favoring algorithm though, similar to Nakamoto – miners could drop off, and the chain kept going. The downside was that Nakamoto could only provide probabilistic finality – you assume after a sufficient number of blocks that a reorg

won't occur.

Gasper aims to achieve the best of both – it can achieve finality once the requisite votes are achieved, but it can remain live and continue an optimistic ledger even if they aren't (it just won't finalize).

Gasper achieves finality by checkpointing periodically with sufficient votes. Each instance of 32 ETH is a functionally separate validator, and there are already >400,000 Beacon Chain validators (though most are run by shared entities due to the 32 ETH cap). Epochs consist of 32 slots, with all validators split up and attesting to one slot within a given epoch (1/32 in each slot). The fork-choice rule LMD GHOST then determines the current head of the chain based on these attestations.

The biggest tradeoff of Tendermint vs. Gasper today is validator set limits vs. finality time:

- Tendermint – Small validator set, but fast finality
- Gasper – Large validator set, but slow finality (though single-slot finality is an open area of research)

In Gasper, a new block is added every slot (12 seconds), so epochs are 6.4 minutes. Blocks can become “justified” after one successful epoch, and “finalized” after two epochs with the requisite votes.

Miner (Nakamoto Consensus) vs. Validator (Gasper) Censorship

This consensus swap alters the conversation for how censorship can occur on Ethereum. In case you've been living under a rock, that's a pretty important discussion at the moment with all the Tornado Cash/OFAC shenanigans going on. Some framing:

- **“Weak” censorship** = Delayed but eventual inclusion. If, say, 50% of miners/validators don't include OFAC transactions, then on average they'll get included after 2 blocks (exactly 24 seconds in PoS, and ~24 seconds

under PoW). If 90% censor, they'll get in after 10 blocks (120 seconds), etc.

- **"Strong" censorship** = Censored transactions aren't included on-chain. For both NC and Gasper this requires ~51% of miners/validators (simplifying here, proposer boost can change the math in Gasper) to not only censor OFAC transactions for their own blocks, but to also actively ignore all new blocks which include them. They would build directly competing blocks that leave blocks with OFAC transactions uncled.

If miners censor – you're kind of stuck. You could swap the hashing algorithm, but this hurts everyone and starts your chain security from scratch/makes it easy to attack again.

If validators censor – you can punish them. This can be done via:

- **User-activated soft fork (UASF)** – The censoring validators are inactivity-leaked out. In Gasper, you're penalized for not voting, and stake is burned rapidly if the chain isn't finalizing.
- **Hard fork** – Direct social slashing of the censoring validators' stake to burn their money.

UASF/slashing would be a rather extreme response to weak censorship. The red line to cross would likely be if validators conduct strong censorship. Luckily, this doesn't appear to be an imminent threat. Validators have seemingly taken the stance that they're not obligated to conduct such an attack (yay!).



The imminent censorship threat is weak censorship stemming from the relay/builder level. Flashbots gets the headlines for censoring (they run the largest relay and builders), but note that most relays/builders are currently censoring (they just have less market share).

Enshrined Proposer-Builder Separation

Overview

Miners and validators serve two roles by default:

1. Build the block
2. Propose it to the rest of the network

Block building is what determines how much MEV is captured, and doing this optimally requires sophistication. If validators are tasked with this, they'll become centralized. Sophisticated stakers would be able to extract more MEV → higher staking yield → increase their validator market share. Unsophisticated validators are incentivized to unstake and delegate to them (to receive that higher staking yield).

PBS addresses this – it separates “building” from validators’ responsibilities by creating a new specialized “builder” role. Builders can try to extract as much MEV as possible, then bid for the proposer (validator) to select their block. In a competitive market, builders will bid roughly the full value they can extract – most value flows to validators.

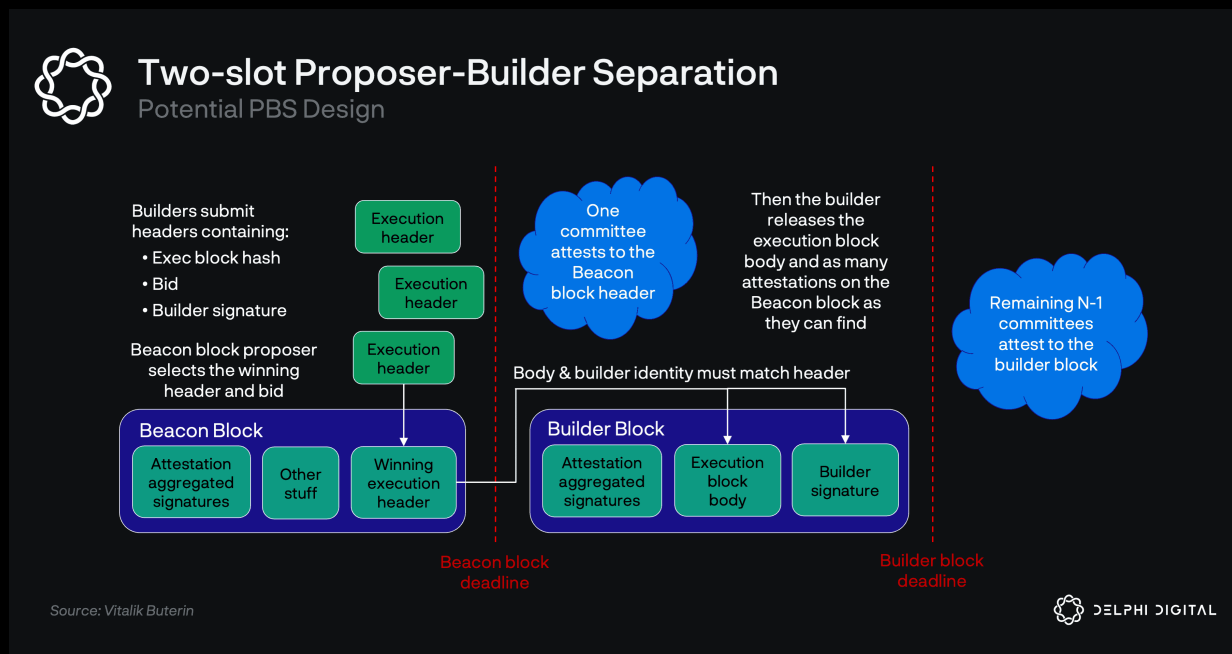
PBS then isolates the centralizing force of MEV extraction to builders, keeping validators decentralized. Validator sets require an honest majority, but builders have much weaker assumptions. In theory – one honest builder can provide liveness and censorship resistance, and two can provide an efficient market. In reality though, we should strive for much better than that.



Two-Slot Proposer-Builder Separation

PBS research is still a work in progress, but two-slot PBS could look like this:

1. Builders commit to block headers along with their bids.
2. Beacon block proposer chooses the winning header and bid. Proposer is paid the winning bid unconditionally, even if the builder fails to produce the body.
3. Committee of attesters confirm the winning header.
4. Builder reveals the winning body.
5. Separate committees of attesters elect the winning body (or vote that it was absent if the winning builder withholds it).



Unconditional payment – removes the need for proposers to trust builders. Once the proposer commits to a block header, they're guaranteed to be paid (even if the builder withholds the block body or sends an invalid one).

Commit-reveal scheme – removes the need for builders to trust proposers. It does this by removing validators' ability to MEV-steal. If builders were to share block bodies at the start, then another builder or the proposer could swap out juicy MEV transactions to capture them for themselves.

This is the type of trust that the MEV-Relay placed in mining pool operators – miners saw the bundle contents, so they could steal them. You could only send bundles to mining pools that could be trusted and monitored. This trust wouldn't

scale to a decentralized validator set, so validators must commit to the header without knowing the block contents. If they try to MEV-steal after seeing the block body, their original signature can be presented proposer is slashed for double-signing.

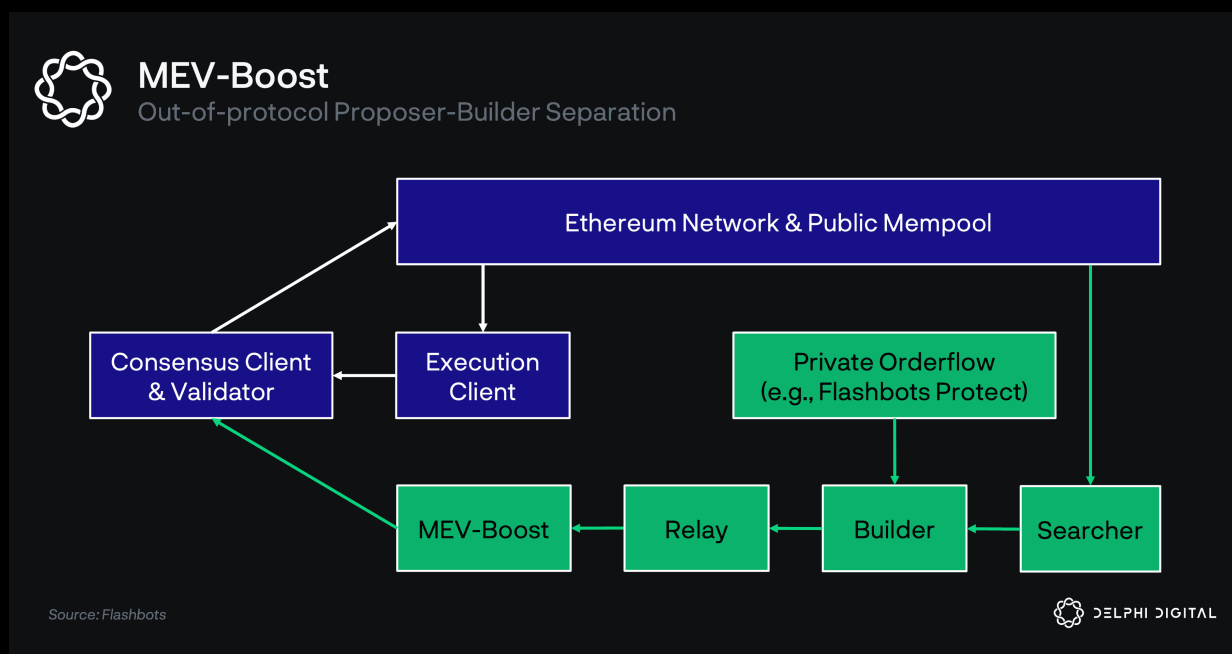
MEV-Boost

Overview

PBS is already here though – [MEV-Boost](#) is the stepping stone solution.

By default, proposers receive public mempool transactions directly into their execution clients package them into a block send them to their consensus client broadcast them to the network.

Alternatively, validators can run MEV-Boost (sidecar software) to query for outsourced block building. The process then looks similar to enshrined PBS, except there's a doubly-trusted "relay" sitting between proposers and builders. Note that validators still retain the option to use their own execution client to build blocks locally.



MEV-Boost patches two main shortcomings of MEV-Geth:

1. **Solo-staker participation** – As discussed, you could only send bundles to

mining pool operators that can be trusted and monitored. This doesn't scale to the full Ethereum validator set. MEV-Boost instead provides full blocks to proposers, who commit to the block header before the block body is revealed. This removes the need to trust them.

2. **Client diversity** – MEV-Geth was a Geth fork which nearly all miners ran. MEV-Boost is instead a sidecar piece of software which is interoperable with all consensus and execution clients.

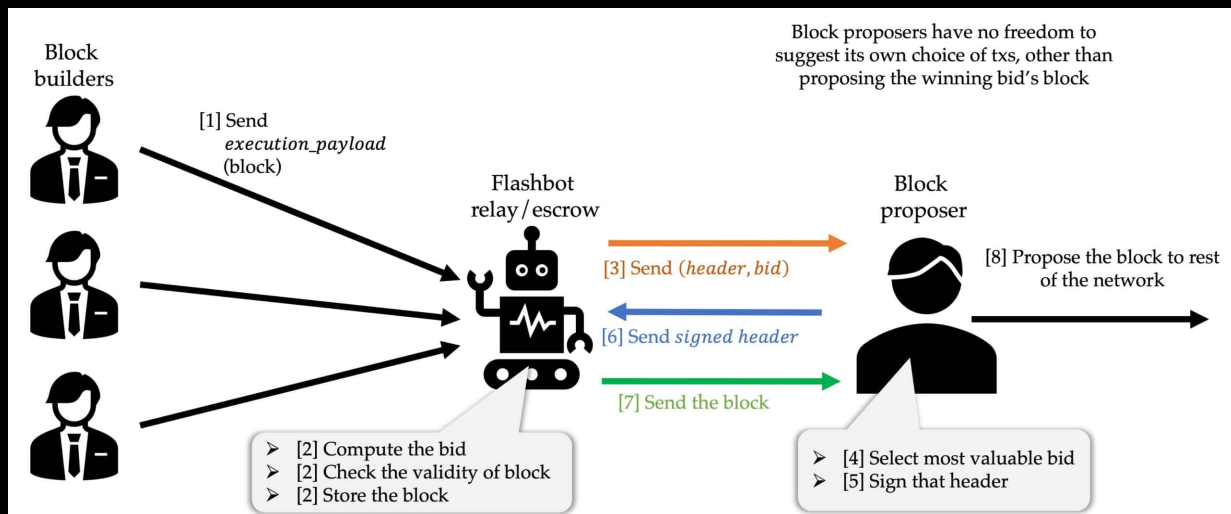
Note that MEV-Boost is neutral infrastructure. It doesn't censor transactions or sandwich trades. It simply allows proposers to outsource building:

- **Validators** – Free to run MEV-Boost or not. Free to use it alongside whichever clients they wish. Free to connect to as many or as few relays as they wish. MEV-Boost is effectively a relay aggregator that chooses the most profitable bid from the proposer's selected relays.
- **Relays** – Free to be run with whichever limitations or strategies they wish (e.g., censoring/non-censoring, "fair" ordering/max-profit, etc.). Relays can accept blocks from builders as they wish.
- **Builders** – Free to run whichever strategies they please and send blocks to any relay that's willing to accept their blocks.

MEV-Boost will continue to iterate and ship new features over time, moving it closer to enshrined PBS. One such goal could be to transition this process into a gossip protocol. However, it doesn't currently work this way because it requires further changes to the gossip network for the client, and the goal was to launch while keeping MEV-Boost as simple as possible.

Relays

Relays are intermediaries which are trusted by both proposers and builders. They receive builder blocks and escrow them before sending them to the proposer. For a given relay, the process can look like this:



Source: EigenLayer

A note on some terminology that may be confusing. This “relay” is different from the “MEV-Relay” (also commonly referred to as simply “the relay”) referenced earlier under PoW.

- The “relay” in the context of MEV-Boost is used to relay blocks from the builder to the proposer
- The “MEV-Relay” accepted bundles from searchers to forward them to miners

However, [nothing changed from the searcher perspective of bundle submission here](#). The Flashbots builder hosts all the same endpoints as the PoW MEV-Relay API. The original URL for MEV-Relay (relay.flashbots.net) now just points to the Flashbots builder.

Flashbots Builder – New Profit-Maximizing Algorithm

Searchers still send and simulate atomic bundles as they had before, and they can use the same bidding strategies in the auction process. There are some exciting changes though, as the Flashbots builders run new and improved merging algorithms.

Recall that in PoW, bundles were inserted at the top of the block, displacing transactions at the bottom of the block as needed. For a bundle to be profitable, it had to have a higher effective gas price than the transactions it would displace

at the bottom of the block.

More profitable transactions/bundles will still generally be favored by the block-building algorithm in PoS Ethereum. Transaction/bundle profitability is determined by fee per gas used, priority fee, and direct validator payments.

The new Flashbots builder algorithm is designed to produce the most profitable block possible. The practical implications include:

- Algorithm now ranks and includes bundles optimizing for overall block profit (no longer based on effective gas price).
- Bundle ordering by effective gas price is no longer guaranteed.
- Top-of-block execution is no longer guaranteed.
- Other transactions (e.g. from the mempool) may land between separate bundles (but not between transactions within a given bundle). For example:
 - Searcher A has bundle with transactions A_1, A_2
 - Searcher B has bundle with transactions B_1, B_2
 - Mempool has transactions M_1, M_2
 - Block may be built with the following ordering: $A_1, A_2, M_1, M_2, B_1, B_2$
 - Block may not be built with the following ordering: $A_1, M_1, A_2, M_2, B_1, B_2$

So bundles remain atomic, but they no longer need to be ordered directly one after the other at the top of the block.

Choosing a Builder

Searchers would ideally connect to every available builder to maximize their inclusion rate. A profitable bundle is of no use if the builder you sent it to loses the auction.

However, searchers need to trust the builders they're connected to. Builders have a clear view of the bundles submitted to them, and are able to abuse this (frontrun, censor, MEV-steal, leak data, etc.).

So there's a balance for searchers between choosing many builders to maximize bundle inclusion, and choosing only one builder to minimize trust assumptions.

For builders to maximize inclusion over time, they need a mix of:

- Building the most profitable blocks
- Using a relay that many proposers are connected to – if the current proposer isn't running MEV-Boost and connected to the relays that the builder sends blocks to, then block profitability doesn't matter

Searchers should also consider more idiosyncratic builder differences. One builder's algorithm may result in slightly higher inclusion rates for certain strategies, some may not allow front-running, others may censor bundles with OFAC transactions, etc.

Relay Trust Assumptions & Liveness Risks

Enshrined PBS removes the need for relays due to the "unconditional" nature of the payment via the in-protocol auction as described earlier.

For now though, relays exist, and they must be selected carefully. Proposers and builders use whichever relays they deem trustworthy and effective. MEV-Boost provides some strict improvements to trust assumptions in "stage 1 PBS" (outsourcing partial block production in PoW), but relays are still trusted in several ways:

- **Builders** – Trust relays to pass along their block if it's the most profitable valid block. Dishonest relays could choose a less profitable block (e.g., from their in-house builder), leak the block information, and MEV-steal.
- **Proposers** – Trust relays to accurately assess builder blocks, send the most profitable valid block, escrow the block until the proposer signs it, reveal the block body upon the proposer's commitment to its header, then pay the proposer accordingly.

The relay is also of course trusted and competing on overall performance (e.g., uptime and latency).

Malfunctioning Relays

Remember that the relay interacts with the proposer in two steps:

1. Provide the proposer with their bid and block header
2. After the proposer has signed the block header, the relay provides the full block body

In the example below, R_M = relay which is either malfunctioning or malicious. Let's see what happens if they fail at either stage:

1. **Step 1 failure** – The relay goes offline. Validator may propose a block from another relay or build it locally. If this is less profitable than what R_M could have provided, the validator and builder who made the actual most-profitable block will lose out on some money, but that's it.
2. **Step 2 failure** – The validator is forced to have an empty slot. Proposing an alternative block at this point would mean the validator has double signed, and they could be slashed for doing so.

The second case is the really bad one, so that's the one we're more concerned about. Validators can always fall back to local block production if all relays go offline (or if they don't have a relay listed in their MEV-Boost configuration). Concretely, the risk is that:

1. Validator is connected to R_M (not necessarily exclusively),
2. R_M is the highest bidding relay, so the validator selects its block, and
3. R_M sent the block header which the validator signed initially, but then it fails to reveal and publish the entire block

We've already seen this happen [with bloXroute recently](#). Their relay malfunctioned for several hours, directly causing 88 proposers to miss their slots. So malfunctioning relays are bad, but they're still not the end of the world for two reasons:

1. It's likely that a given relay isn't always the highest bidder
2. They can rectify the issue once they realize it

Similarly, [bloXroute had issues with sending invalid blocks to proposers](#) due to

the fact that their relay wasn't simulating blocks from known builders. 15 proposers missed their slots. bloXroute has rectified these issues – they repaid proposers for their missed slots, and have started to simulate transactions. This highlights the very-real risk though.

Malicious Relays – Block Withholding Attack

However, a popular malicious relay is of greater concern. If R_M is connected to many subsequent validators, then intentionally conducting a block withholding attack will result in a complete liveness failure of the Ethereum network. Neither of the mitigating factors for malfunctioning relays apply:

1. Malicious relay can lie and always bid the highest – this is a free attack as they aren't committed to paying the proposer until they reveal the full block
2. Malicious relay will intentionally continue the attack, not rectify it

There are simple local solutions for validators – track recent performance of the relays you've used, and disconnect if something is wrong. This may be useful for large validators (e.g., Coinbase) who propose many blocks and have recent data, but it doesn't do much for smaller validators who propose very infrequently. It also doesn't help any other validators if only Coinbase realizes the problem and disconnects.

So we need a global solution, of which there exist two leading ideas:

Relay Monitor – A trusted third party monitors all relays. If R_M withholds blocks, provides invalid ones, or pays proposers less than they had initially bid – the monitor can alert all proposers "remove R_M now." It wouldn't have the power to make validators connect to some other relay, so the worst a malfunctioning or malicious relay monitor could do is cause validators to fall back to local block production, not miss slots. Flashbots recently announced they would [issue a grant to develop such a relay monitor](#).



Whenever MEV-Boost calls *submitBlindedBlock* to a relay, it also sends a request to the relay monitor including:

- The *SignedBuilderBid*
- The relay it originated from
- The *submitBlindedBlock* body

The relay monitor also requests the payload from the relay, allowing it to check whether:

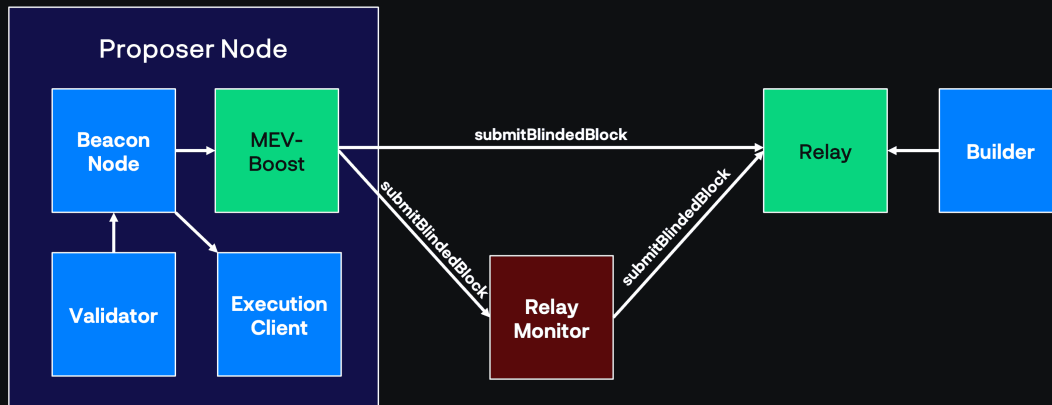
- The payload is withheld
- The block matches the bid

If the relay monitor observes any issues, it will propagate this to all proposers.



Relay Monitor

Mitigating Liveness Risks of MEV-Boost



 DELPHI DIGITAL

Circuit Breaker – This serves a similar function of alerting proposers to disconnect from relays, but it's run as part of the Beacon node rather than relying on a trusted third party. Proposers just watch for some threshold of recent slots being missed since the last valid block, in which case they fall back to local block production. This differs from the relay monitor in that it entirely cuts off outsourced block production for the proposer, not just one relay.

The best solution of course is enshrined PBS – relays are no longer needed, and the unconditional payment would make this attack prohibitively expensive for builders to conduct. It may even be possible to implement a crude form of PBS simply by adding a new transaction type to Ethereum which allows builders to offer proposers this unconditional payment, as [Phil](#) has proposed previously.

Summing up where are now vs. enshrined PBS:



Enshrined PBS vs. Current MEV-Boost

Major Trust Assumptions^(1,2)

	Enshrined PBS			Current MEV-Boost		
	Proposers	Relays	Builders	Proposers	Relays	Builders
Proposers:	-	Relays Removed	No Trust Needed (Unconditional Payment)	-	Proposers Trust Relays	No Trust Needed
Relays:	Relays Removed	-	Relays Removed	No Trust Needed (Commit-reveal)	-	No Trust Needed
Builders:	No Trust Needed (Commit-reveal)	Relays Removed	-	No Trust Needed	Builders Trust Relays	-

1) Major trust assumptions for proposers – counterparty’s ability to underpay the proposer or not pay them at all (e.g., not pay the promised bid), or to make them miss their slot **and** not be paid (send invalid blocks or withhold the block without unconditional payment)
 2) Major trust assumptions for builders – counterparty’s ability to leak their block information and/or MEV-steal



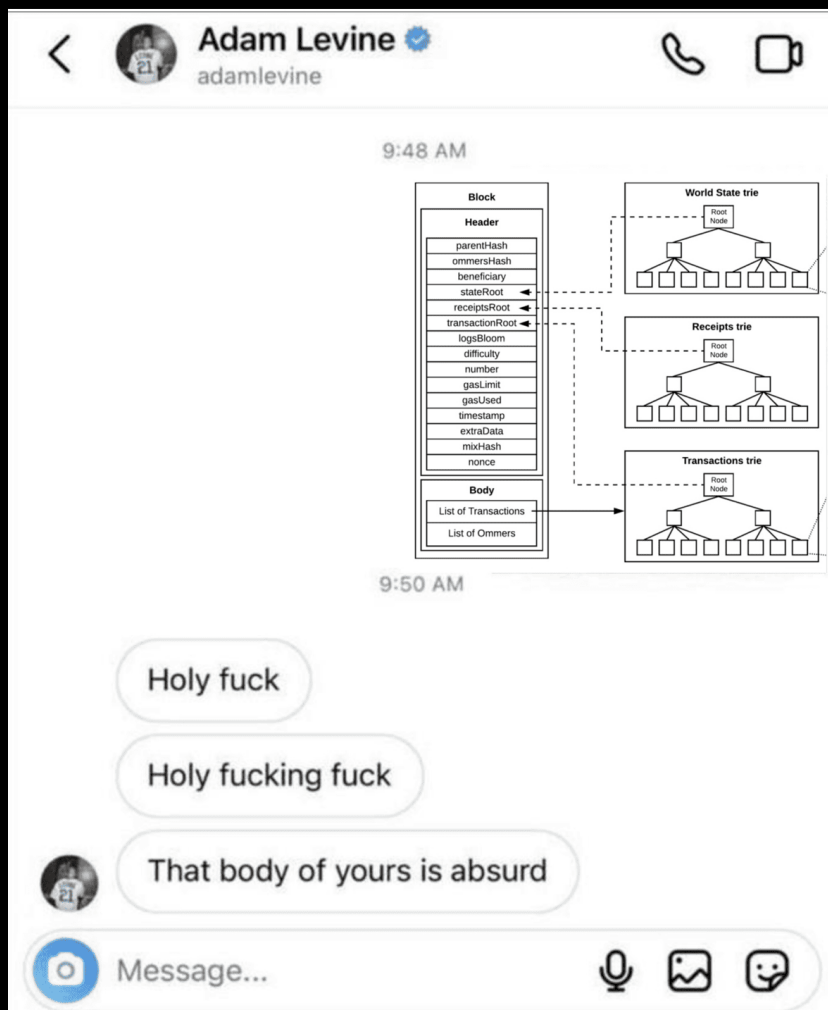
Relay & Builder Censorship

Now we’ll finally chat a bit about all the headlines. We discussed validator censorship earlier, but that’s not the problem right now. Censorship is currently stemming from relays and builders.

Recall that the MEV-Relay only sent bundles to the miner in PoW. They were never sent full blocks. Flashbots never included OFAC-blacklisted transactions in these bundles, but it didn’t really matter because miners could just include them anywhere else in the block.

This is very different from the function of “relays” in MEV-Boost, which send full blocks to validators that they blindly commit to. The full block body is only revealed *after* the proposer has committed to it. As currently constructed, the proposer is unable to do anything if the block revealed is censoring.

WHEN THE RELAY REVEALS THE BLOCK BODY



BUT IT'S GOT NO OFAC TRANSACTIONS

However, full block outsourcing now introduces serious censorship risk – proposers who accept blocks from censoring relays will be de facto censoring. They no longer retain the ability to append transactions in the way that miners did. If a censoring relay/builder is the most profitable, then the proposer is forced to choose between being:

- **Economically rational** – accepting the highest value block, even if censored
- **Altruistic** – accepting a lower value block which does not censor

To be clear, this risk was known when Flashbots was building MEV-Boost. However, the Ethereum community gave a clear objective that the validator set needed to remain decentralized. As mentioned, this full-block commit-reveal

scheme was needed to ensure that relays don't have to trust validators in the way that a handful of mining pools could be. Otherwise, you'd end up with a centralized validator set. So this censorship vector was deemed a necessary tradeoff.



So is it game over? Will relays and builders just censor forever? I don't think so. We can save her.

I expressed some of my stronger personal thoughts in that [article](#) I referenced

earlier, but the reality is this is still very open research. I'll have more to come here.

Are Maximally-Decentralized Validators Necessary?

Now that we understand the risks imposed by relays and builders, let's circle back to validators. Today's MEV-Boost censorship issues arise due to the need to remove trust in proposers. This allows for the possibility of having a bajillion validators participating. But is that necessary? Zaki has noted this tension recently:



Zaki 🧬 🍷 @zmanian · 1d



The design of mev-boost was to enable home stakers to capture MEV.

The design choice to prioritize the economics of home stakers has worsened censorship resistance of Ethereum as a whole because it requires the builder to force the block producer to sign the entire block.



Zaki 🧬 🍷 @zmanian · 1d



Replying to @zmanian

The reality is known entity reputation and getting rid of mev boost would allow the vast majority of Ethereum blocks to be built in a more censorship resistant manner.

This is a lesson that the dogma of home staking == censorship resistance is not always correct.

The primary goal of decentralized validators is to provide censorship resistance.

Decentralization is more a means to an end than some inherently great property. However, we see today that you can still easily have censorship even with decentralized validators.

If validators were limited to large entities that relays/builders could trust, today's issues would be avoided. As in PoW, relays could just send them full blocks in the clear, and validators would be free to append additional, otherwise censored transactions. If Coinbase misbehaves and starts stealing MEV from the relays, then they'd simply be cut off.

Even a smaller set of validators (ballpark ~100 in Tendermint usually) can still be held accountable by the same mechanisms described earlier. If they're clearly conducting strong censorship, the community can always fall back to a UASF / hard fork to punish them.

The flip-side as I mentioned is that these aren't necessarily mutually exclusive. Changes to the base layer and supporting infrastructure could bring back proposers' agency to force in censored transactions without needing to trust them.

However, these solutions are difficult, and they're not live today. The message is simply that decentralized validators aren't a panacea in isolation.

That's really the heart of the issue here. The weak censorship we see today hasn't arisen due to the shift from PoW to PoS. It's here because of Ethereum's desire to support a decentralized validator set which doesn't need to be trusted. Ultimately, this tension will need to be resolved through better mechanism design if it's to be successful.

Conclusion

Thoughtful protocol and app-layer design can certainly minimize unnecessary MEV exposure and return value to users. However, I'd be skeptical of any silver bullets claiming to "solve" MEV. MEV exists, and ignoring it doesn't solve anything. Left unchecked, it can introduce perverse incentives. The core protocol needs to be built with these incentives in mind.

This discussion is deeply intertwined with that around censorship resistance, including new vectors for censorship at the relay and builder levels. There are many ways to fight this threat though, and that should be front of mind for all researchers and developers.

Personal note – apologies for the long delay between reports. This report is only a brief primer – MEV has a bunch of fascinating areas of open research, and I've fallen down a bit of a rabbit hole lately. Following MEV reports will hopefully come a bit quicker.

