MEV: The Next Five Years

James Prestwich

Follow

--

Listen

Share

That’s right I just won’t shut up

Five half-weeks ago I wrote a blog post calledMEV: The First Five Years. It was, I expect, broadly shared and highly regarded. To be honest, I wrote that entire post as a lead-in for this one. Yeah, that seems a tad crazy, no? Who writes a few thousand words just so they can write more? Well. I have some thoughts

. I am full of opinions (among other things), and I am determined to share them with you.

Seeing MEV extraction evolve over the last year has, counterintuitively, given me a lot of hope about its future role in the crypto ecosystem. MEV’s story to date is about the professionalization of extraction. This post is about what professionalization means for the future of MEV-aware design.

Author’s notes:

This is (still) not an MEV primer. Many of those have been written, and most are better than I could produce. I assume you’re familiar with MEV. If you’re not familiar, this is a good time to go read a few blog posts and then come back.

This is a collection of musings and predictions on where I think MEV could go in the future. It’s completely biased. Y’all are welcome to agree or disagree. I’ll get pretty conceptual. However, I promise to be 30% less pompous than the previous post. This one is shorter, and there will be some fun concrete stuff towards the end :)

MEV: The Next Five Years

This is a prediction post. As I’ve done in the past, I’m gonna be calling some shots. I feel like I have a pretty good controversial predictions track record. Let’s schedule a few twitter fights for 2–5 years from now. I expect y’all to tell me how wrong I was. Probably can’t avoid the eventual public shaming anyway, so in the interest of my future ego, let’s start with some softballs!

The Spread of the 3-role MEV Market

During the bear market, there will be a few coordinated pushes to boot up MEV extraction supply chains in other ecosystems. Solana, Cosmos, Near, etc etc. From where we stand, it seems obvious that the Searcher-Builder-Proposer supply chain is going to be replicated in all of those ecosystems. Specialization wins in complex markets. SBP market structure dominates because the simple boundaries allow each role to specialize without coordinating with other roles. It’s hard to see other market structures forming in the current landscape.

However, some fee markets are sufficiently distinct from Ethereum’s. In chains likePenumbra, with batch auctions and threshold-encrypted transactions, the SBP market struct won’t form. Other chain teams (e.g. Osmosis) have already spent significant research time on MEV-fighting mechanisms. It’s unclear what the market structure for MEV extraction will look like on those chains. And it seems possible that no mature extraction supply chain will form there. If it’s sufficiently annoying to build the supply-chain (relative to the amount of MEV available), nobody will even bother.

MEV market structure is in tension with the protocol fee market structure, and bends it towards first-price auction. The tension is resolved (or institutionalized, depending on your perspective) by bifurcating the fee market into MEV and non-MEV. Normal fee markets will be pulled towards this structure with an SBP extraction supply chain. Abnormal fee markets may see a unique MEV supply chain if (and only if!) they generate enough accessible MEV to make development of the software worthwhile.

Resurgence of Backlash

Another easy call, in my opinion. We used to have excellent debates about whether MEV extraction was (morally or legally (discounting Avi’s whole Mango thing, I don’t think that oracle manipulation is MEV)) theft. Going into the bull market, a lot of that calmed down as everyone got busy stealing everything that wasn’t nailed down. Now that on-chain financial activity has chilled out a bit, I expect we’ll see a return to moralizing on the “ethics” of MEV extraction. CryptoTwitter will rerun some old

old episodes, and the fandom will have some fierce debates about who “deserves” the extracted value. However, Searchers won’t care. Extraction will continue.

MEV-Aware Hard Forks

At risk of repeating myself: MEV bends fee markets towards first-price auctions. Core devs already think about MEV with respect to protocol design. It’s a topic of conversation. It’s a concern. From my (informal) interactions with several ecosystems, core devs tend to lean towards one of two camps.

The first camp of core devs, as alluded to in the previous section, design mechanisms to raise the costs of MEV extraction. These look like batch auctions, threshold encryption, pre-consensus, etc etc. Mechanisms, they hope, can make MEV so prohibitively expensive to extract that extraction is not viable. If extraction requires collusion of the validator set in the face of potential slashing, it probably won’t get done at all. However, these mechanisms add costs or complexity to user interaction and protocol incentives. No specific mechanism design has reached broad consensus.

The other group of protocol devs has a nuanced approach. They prefer elegant mechanisms and a more civilized chain with focus on minimization, mitigation and participation. First, how do you minimize the impact of MEV on users of the protocol? Second, how do you mitigate the impact of MEV on the protocol itself? Third, how do you ensure the protocol receives a fair share of the extracted value?

Consider, for example, EIP-1559. While it’s not explicitly an MEV EIP, it highlights the three concerns. EIP-1559 minimizes the impact of fee spikes on users, by allowing blockspace to expand and user fees to float with the basefee. It mitigates the impact of fee spikes on the protocol, by regulating the relationship between block capacity, gas usage, and basefee. And finally, it participates in the value capture by increasing basefee proportional.

Sophisticated, yet simple, mechanisms like EIP-1559 are hard to come by. They don’t really grow on trees. I think that they’ll follow in 1559’s footsteps, and target proposers’ rents. MEV’s shadow fee market benefits proposers and MEV creators to the detriment of all other users. Like pre-1559 fees, MEV block production and inclusion dynamics create a fundamentally unfair market. The market structure cannot be improved without direct interference. The protocol must control the market. End proposer sinecures. To that end, I hope that we’ll see a 1559-like mechanism targeting the relationship between builders and proposers.

Regardless of my political views vis-a-vis proposers being useless rentiers who deserve to have their revenues choked, my overall prediction here is pretty simple: within the next 5 years we’ll see multiple hard forks in different chain ecosystems that explicitly target the MEV market for a specific chain. They could be targeted at MEV extraction costs, or at MEV profit shares. But they will target MEV. Ideally we will see tight integration of MEV extraction supply chains with L1 designs.

Vertical Integration (in special cases)

A lot of noise has been made about vertical integration between proposers, builders, and searchers. The basic idea is that a proposer might keep an in-house builder, in order to capture the builder revenue, or a builder might prioritize specific searchers over others. Vertical integration could squeeze out smaller operations, and lead to toxic markets dominated by a few major builders.

I think this fear is a bit overblown. The protocol selects the proposers randomly-ish. And the markets for searching and building are fiercely competitive. Horizontal cooperation between proposers (who have all the power in the market) seems much more worrying. Price-fixing arrangements between proposers could be enforced by staker-activated soft fork. The protocol may have to become a proposer trust-buster to prevent toxic PBS markets from affecting user experience. Even worse, multi-block MEV could be extracted whenever colluding proposers have adjacent slots, which could have destabilizing effects on chain finalization.

Vertical integration will certainly occur in special circumstances. For example, bundles extract MEV in a vacuum, without considering other bundles. Some amount of MEV is created by the interaction between bundles. Builders have privileged access to this cross-bundle MEV, and will likely integrate an in-house, specialized, Searcher. There are likely additional special cases where a closer, more trusted, relationship between Searcher, Builder, and Proposer are economically viable. Looking forward to learning about those in the future.

MEV-Aware Application Design

This one’s cheating, as it’s already started happening. Teams likeCowSwap are already launching MEV-aware applications. MEV minimization will be the name of the game for some time, as we learn the tradeoffs. Research here is driven by a single, central question: What do we have to give up in order to remove MEV?

While protocols have a very constrained design space (it’s actually kinda hard to make a consensus protocol), applications have tons of options for minimizing MEV. And what works is application-dependent. We’ll see minimization mechanisms for DEXes (like batching) that won’t work for lending protocols. New research will drive some changes in each dapp vertical. However, new MEV-minimizing models need to compete with the momentum of existing applications. It’s still unclear what dapps will see wide user-adoption of MEV-resistant designs, and what dapps will accept current levels of MEV generation.

However, we’re not going to stop at designing protocols to minimize MEV. That’s myopic. Instead, MEV will become a tool in application design. Applications designers will choose when and how to generate MEV. We’ll come up with cool frameworks

for deciding when MEV is acceptable, and a standard set of tricks for adjusting the amount of MEV the system produces.

Explicit MEV

The inevitable extension to MEV-aware application design is co-opting MEV, and using it as a tool. To everyone's surprise, the MEV supply chain provides guarantees that normal transactions can't access. MEV transactions never revert. Builders don't waste blockspace on unprofitable reversions! MEV transactions confirm quicker. Searchers convert that MEV into shadow fees, giving the transaction higher priority. Applications will start using MEV deliberately to access these features. I call this "Explicit MEV

".

Explicit MEV gives apps super powers. It allows them to opt out of 1559 tipping, and still have their transactions confirmed quickly. An app can open an MEV opportunity on purpose, and couple it to a critical system action. For example, we can add 0.1 ETH MEV to an oracle update. This MEV opportunity creates a race between Searchers. The first Searcher to confirm a transaction gets that MEV, and as a side-effect, fulfills the application's need. In pre-PBS MEV, this would result in a priority gas auction, which could be very costly to involved Searchers. However, the modern MEV supply chain removes the cost of losing this race. Searchers know that reverting transactions will be dropped by builders.

The logical conclusion of MEV-aware design is MEV-inclusive design. Applications will explicitly create MEV. They will incorporate MEV into their designs, and use it to buy privileged positions in the blockspace market. Explicit MEV transactions will effectively replace standard transactions for high-value use cases.

MevWeth

This is the part where I start putting my thumb on the scales.

Explicit MEV requires two things:

First, a standard for creating it. To avoid costly gas, all explicit MEV should use the same on-chain state, regardless of what contract creates it. This allows MEV to build up across many transactions, without complexity explosions. Searchers need a simple way to know that an application is creating Explicit MEV. Searchers should be able to look for specific, predictable state changes. They should not have to scan a large number of contracts to detect explicit MEV. If each contract managed its own Explicit MEV, searchers could not harvest it without a large number of state lookups and changes. A single, specific contract should manage Explicit MEV.

Second, a separate accounting system for it. Explicit MEV should be denominated in ETH. Because searchers take gas price risk, their payment should be in the gas asset. It simplifies everything immensely. However, this imposes a new requirement on contracts generating Explicit MEV: Contracts must track how much ETH they can spend on MEV. Accounting. Our ancient foe. A DEX accidentally spending user ETH on Explicit MEV would be disastrous. Therefore, any contract using Explicit MEV and ETH must keep a balance of MEV-able ETH and non-MEV-able ETH. Kinda gross.

So we need a standard for creating MEV, and an accounting system for MEV. The (pretty clear, obvious, intuitive) solution combines these needs. We can easily build a system that tracks user balances of MEV-able ETH. We know exactly how to build contracts that accept ETH and hold it for users. It's trivial to extend it to track and manage Explicit MEV.

We might call this "MEV Wrapped ETH" or MevWeth.

The code might look like an extension of WETH10 with an Explicit MEV management interface. It might be hosted on github and possibly deployed mainnet already. Maybe. Probably not tho, idk.

MevWeth is an ERC20 that provides a standard for:

- Creating MEV, via the addMev() family of functions,
- Retrieving MEV, via the getMev() family
- Adding Explicit MEV creation to contracts via the Mevitize.sol base contract

It's cool, you should check it out, but read the rest of the blogpost FIRST.

Negative MEV

Explicit MEV can't be separated from the transaction that creates it. It is atomic. Explicit MEV joins the implicit MEV in the transaction. Together they subsidize faster confirmation (and all the other benefits of the modern MEV supply chain). Certain classes of transaction, however, already have a lot of MEV. Liquidations, DEX trades, etc already generate sufficient MEV to get priority confirmation, without any explicit value creation. Adding any Explicit MEV to the total would be overpaying for access to the MEV supply chain.

These transactions drastically overpay anyway. Uniswap users don't sit down to calculate the MEV creation range of their transaction. They don't compare that to some concept of "priority fee market clearing MEV." Nobody sees MEV as priority fees (yet). And even if they did, there's no effective way for them to reclaim that value. Negative MEV can change all that.

Negative MEV

is Explicit MEV below zero. An on-chain action with Negative MEV demands an MEV payment via MevWeth. It cannot be completed without prior payment. It is important to understand the existing power relationships in the MEV supply chain. Users and applications create the value being extracted, but do not yet participate in the extraction supply chain. Proposers squeeze the other supply chain participants for rents. Negative MEV is acknowledgement that Proposers are not ultimately in control of this process. Proposers rely on the value users and applications create.

By specifying Negative MEV, a user or protocol can recapture MEV created by valuable actions. We can grab MEV from Proposers' grubby little hands and give it back to the people who created that value. Negative MEV provides a simple & direct way to wield power in the MEV supply chain. With Negative MEV, users become participants in the value extraction process, rather than targets of it.

For example, a Uniswap user might defend against MEV by narrowing the slippage range of their trade. However, this often makes the trade worse or less likely to complete. Negative MEV provides an out-of-band way to improve the trade outcome without changing the trade itself. Instead of narrowing the slippage range, the user can attach a small amount of negative MEV to the transaction. This effectively provides a fee rebate via interaction with the Searcher. The user gets a better outcome. The trade itself is unaffected. Atomic coupling of the trade to a payment to the user allows the user to explicitly share in value that was ultimately created by the user.

Negative MEV will change how protocols gate access to high-MEV processes like liquidations. It will change how users perform high-MEV actions like DEX trades. And (hopefully) it will change the power dynamics of the MEV supply chain. Which brings me to my last prediction

MevWallet

I thought it sounded cool, so I built it.MevWallet is a smart-contract wallet that allows users to attach explicit MEV to transactions. It follows pretty standard EIP-712 meta-transaction semantics, and allows users to specify a — positive or negative! — tip on their transactions. Once a transaction is signed, anyone can broadcast it. MevWallet will verify transaction details and enforce a few basic conditions, then execute the transaction. Like any smart contract wallet, the transaction executes from the wallet, not from any EOA. While msg.sender will be the wallet, tx.origin will be the Searcher

After the transaction is executed, MevWallet meters the gas usage and adds in the tip. If the result is positive, it compensates the Searcher in MevWeth. If the result is negative, it retrieves some stored MEV from MevWeth. That's right, you can pay a negative transaction fee. If the transaction generates enough MEV to totally offset execution, the user can get paid to make it. By demanding to be compensated for the MEV they produce, users can recapture value they would otherwise have ceded to block proposers via the MEV supply chain.

MevWallet transactions support timelocks and deadlines, as well as nonce-based replay protection. This combination allows for complex time-based fee twiddling. For example, a transaction could generate more Explicit MEV over time until it is confirmed. This would create a Dutch Auction among Searchers, ensuring that the transaction gets confirmed at the lowest price available on the market. Plenty of other interesting fee escalator models can be implemented with no on-chain overhead

. Because the fee auctions take place among Searchers, and reverting transactions don't confirm, the user and Searcher bear no additional gas cost, no matter how complex the gas auction becomes.

MevWallet has libraries in TypeScript and Rust. New MevWallets can be deployed via aminimal proxy factory. The code is permissively licensed and available on GitHub. The main thing MevWallet is missing right now is a way to get transactions seen by Searchers. I'm still noodling on the best way to do this. I considered a simple CRUD API, and a specialized mempool, but I'm not sure that's a workable solution long-term.

I'm putting up a 5 ETH rewards pool for bug disclosures and contributions to the MevWallet ecosystem. MevWallet is unaudited, barely tested code. I'm interested in code reviews, design ideas, bug reports & fixes, and educational content. File issues please. For critical issues, DM me on Twitter :)

MEV: The Next Five Years

I expect and hope that the next five years of MEV are as surprising as the first. As MEV research exits adolescence and enters maturity we should welcome it into our application design. We should look for its strengths, and help it contribute to our applications and protocols. MEV is not theft. It's not an enemy to be defeated. It is an integral part of our community of practice.

MEV's story belongs to the profit margins now. That makes it predictable. Profit margins breed stability & reliability. The more money to be made via the market norms, the more sticky the norms are. The story of the next five years will be cementing the existing, stable, supply chain. And leveraging it to build systems that could never exist before. Let's get on it.

Acknowledgements

Everybody. I’ll see you all in hell (but in a good way).