MEV - Maximum Extractable Value

TL;DR Research field into the effects of defi on the core decentralisation & censorship resistance of protocols.

Miner-extractable value (MEV): We introduce the notion of MEV, value that is extractable by miners directly from smart contracts as cryptocurrency profits. One particular source of MEV is ordering optimization (OO) fees, which result from a miner's control of the ordering of transactions in a particular epoch. PGAs and pure revenue opportunities provide one source of OO fees. We show that MEV creates systemic consensus-layer vulnerabilities. - ***Flashboys 2.0 (2019)***

Key ideas:

- Validators/miners have the power to decide ordering and inclusion of transactions.
- They can extract extra value from ordering transactions, or selling the right to order transaction.
- This creates consensus-layer vulnerabilities.

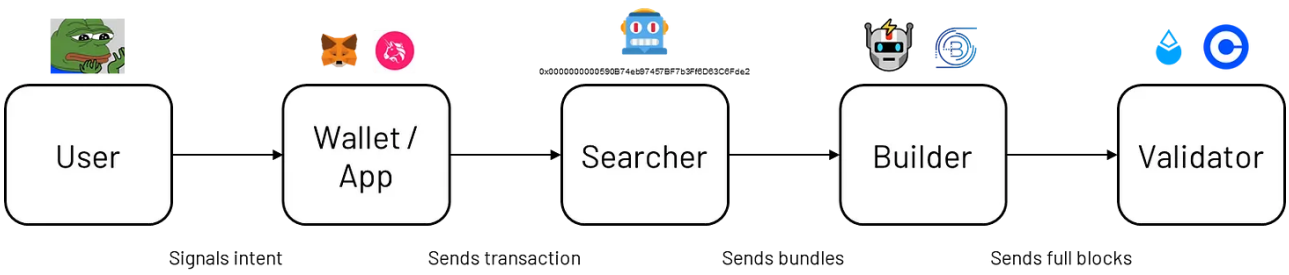
This opaque, highly adviserserial environment is often described as the Dark Forest - Ethereum is a Dark Forest

Ethereum Architecture:

Ethereum is suspectible to MEV in the a few ways:

1. All transactions are available in a public mempool before they are mined
2. All transaction data is public
3. Transactions can be cloned
4. Slow / each block is 12 seconds
5. There a lot of complexity (Defi ect)

Transaction Flow



User: Anyone who would like to express and submit intent to change the state of the blockchain.

Wallet / Application: The user interface that translates user intent into a blockchain transaction.

Searcher: Entities that monitor the mempool and submit transactions to extract MEV.

Builder: Aggregate transactions from various sources to create a full block, ideally one that maximizes rewards.

Validator: Perform consensus duties, such as proposing and attesting to blocks.

Is MEV unique to Ethereum?

No, hypothetically MEV can also be seen on Bitcoin. The incentives to censor Lightning channels or to double-spend colored coins are technically MEV. However, Bitcoin is inherently less exposed to MEV than blockchains like Etheruem.

The reason for that lies in the complexity and "statefulness" of the respective blockchain:

The rate at which MEV accumulates on a given blockchain is generally proportional to the complexity of its application-layer behavior.

Ethereum is a general purpose blockchain so cannot bound this complexity.

MEV incentives cannot be easily mitigated without altering Ethereum's UX.

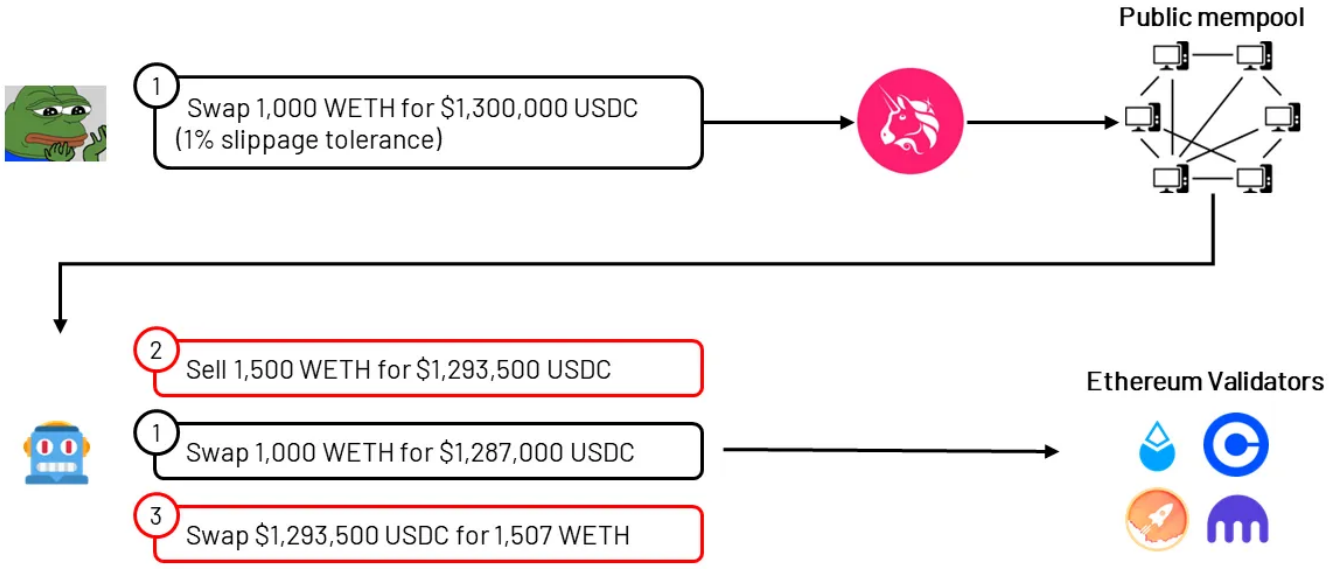
Other highly complex blockchains such as Solona also have high levels of MEV.

Font Running

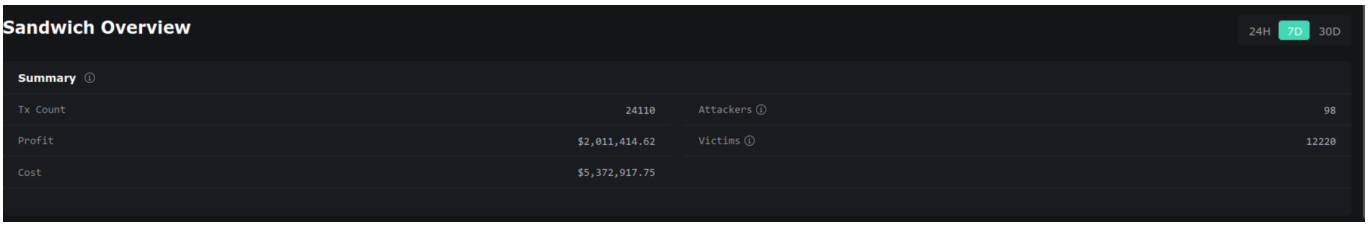
Generalised front running bots: Introductory Video from Scott Bigelow

https://www.youtube.com/watch?v=UZ-NNd6yjFM

- **Front Running**: Monitoring mempool for profitable transactions then submitting them but with higher gas fees.
- **Back Running**: monitoring of a mempool to execute a transaction immediately after a pending target transaction.
- **Sandwich Attack**: Combination of front & back running to sandwich a trade.



MEV analytics: https://eigenphi.io/mev/ethereum/sandwich



Arbitrage & Liquidations

- Arbitrage is the largest source of MEV revenues.
- Arbitrage does helps market efficiency in general.
- But it's incredibly competitive and this creates negative externality such as:
    - incentives to centralise,
    - collude
    - frontrun transactions.

Atomic arbitrage

- Reduced inventroy risk
- no timing risk
- Either you profit or transaction fails
- Flashloan can be used for collateral (less competitive due to gas requirements)

Cross domain MEV

- It is more difficult and risky mainly because you can't do atomic arbitrage.
- Bridges are slow so capital is needed on both chains.
- Transactions must be executed simultaneously on multiple chains.
- Create Incentives for validators to collude and try to monopolise order flow across multiple chains.
- Even if validators don't collude there are economies of scale to running multiple validators on multiple chains.
- Unchecked this could lead to a winner takes all dynamic and more centralisation.

If you want to go deep into the theory: Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value

Liquidations

- Lending protocols Aave, Compound, and Maker.
- Similar to arbitrage, liquidation events are highly competitive. During sharp market downturns, the competition to liquidate borrowers has led to enormous gas fees.

Priority Gas Auctions

- Without an effective fee market, competition for MEV can lead to Priority Gas Auctions.
- Searchers and bots compete against with increasingly higher gas fees

This leads to:

- Network congestion, which may cause other transactions to get stuck or dropped by the network

- High and volatile gas prices, crowding out other crypto users and disadvantaging less savvy participants
- Failed bids that unnecessarily consume block space (even failed transactions use up block space)

Pre Merge approach - Flashbots auction

Flashbots Auction provides a private communication channel between Ethereum users and miners for efficiently communicating preferred transaction order within a block.

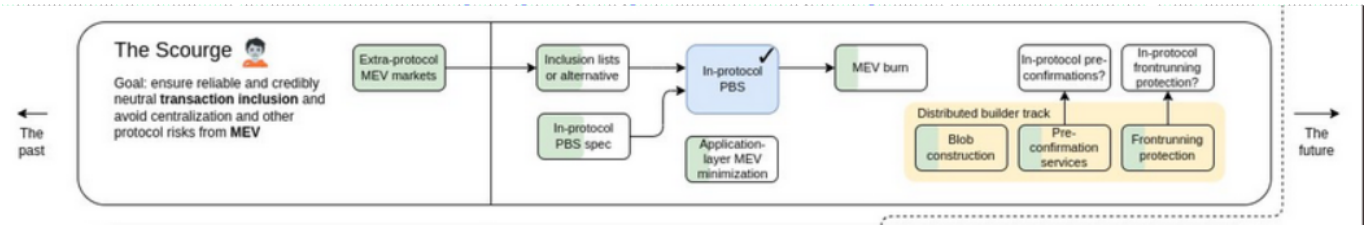
Flashbots Auction consists of mev-geth, a patch on top of the go-ethereum client, along with the mev-relay, a transaction bundle relayer.

Flashbots mitigated the negative effects of PGAs and front running but it created other problems.

Endgame

Endgame Vitalik: https://vitalik.ca/general/2021/12/06/endgame.html

There's a high chance that block production will end up centralized: either the network effects within rollups or the network effects of cross-domain MEV push us in that direction in their own different ways. But what we can do is use protocol-level techniques such as committee validation, data availability sampling and bypass channels to "regulate" this market, ensuring that the winners cannot abuse their power.



Ethereum roadmap: https://twitter.com/VitalikButerin/status/1588669782471368704/photo/1

Proposer Builder Separation (PBS)

(Status: In development not implemented/Not a solved problem)

TL;DR: Split the roles of validators/builders so that the centralisation forces of MEV are contained in builder layer.

- Builders are highly specialized actors that construct candidate blocks and make bids to get them included
- Proposers are validators that naively accept the highest bid
- Goal: builders absorb economies of scale, proposers stay decentralized

But how How to stop validators from stealing the MEV from searchers?

Ideas:

- Encrypt transactions: commit reveal. Trusted hardware etc.
- Use ZK-snarks
- Cr lists. Proposer submits a list of must include transactions to the builder.

Explainer video: https://www.youtube.com/watch?v=fAgrIdyWIqc

Sources: Hitchhikers guide to Ethereum Proposer Builder Separation

MEV-Boost (Flashbots) an interim solution

Mev-Boost is an offchain implementation of PBS

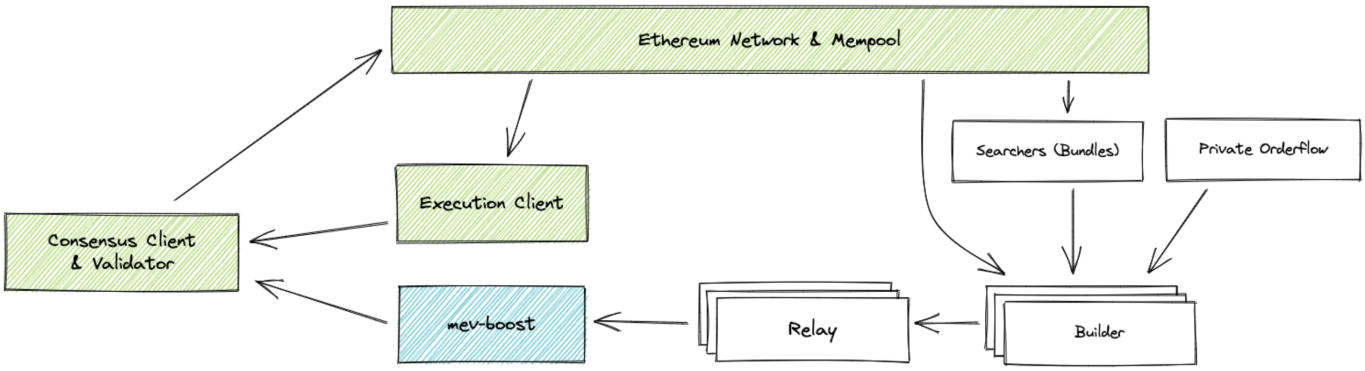
It is an interim solution until PBS can implemented at protocol level. It's optimised for liveness and resilience.

How it works:

- **Searchers** take transactions from the public mempool, potentially adding their own, and arrange them into bundles.
- **Block builders** are services/providers that will aggregate various bundles and transactions into block templates.
- The **relay** receives these block templates (also referred to as execution payloads), and will verify their validity.
- The **MEV-boost** component is middleware which handles communication with the relays, the profit-switching logic (for selecting the most valuable payload), and a fallback mechanism in the case of some system failure.

Using a middleware instead of direct modification of the consensus clients allows for maintaining each component independently and provide cross client compatibility with minimal changes to the engine api.

Source: https://ethresear.ch/t/mev-boost-merge-ready-flashbots-architecture/11177

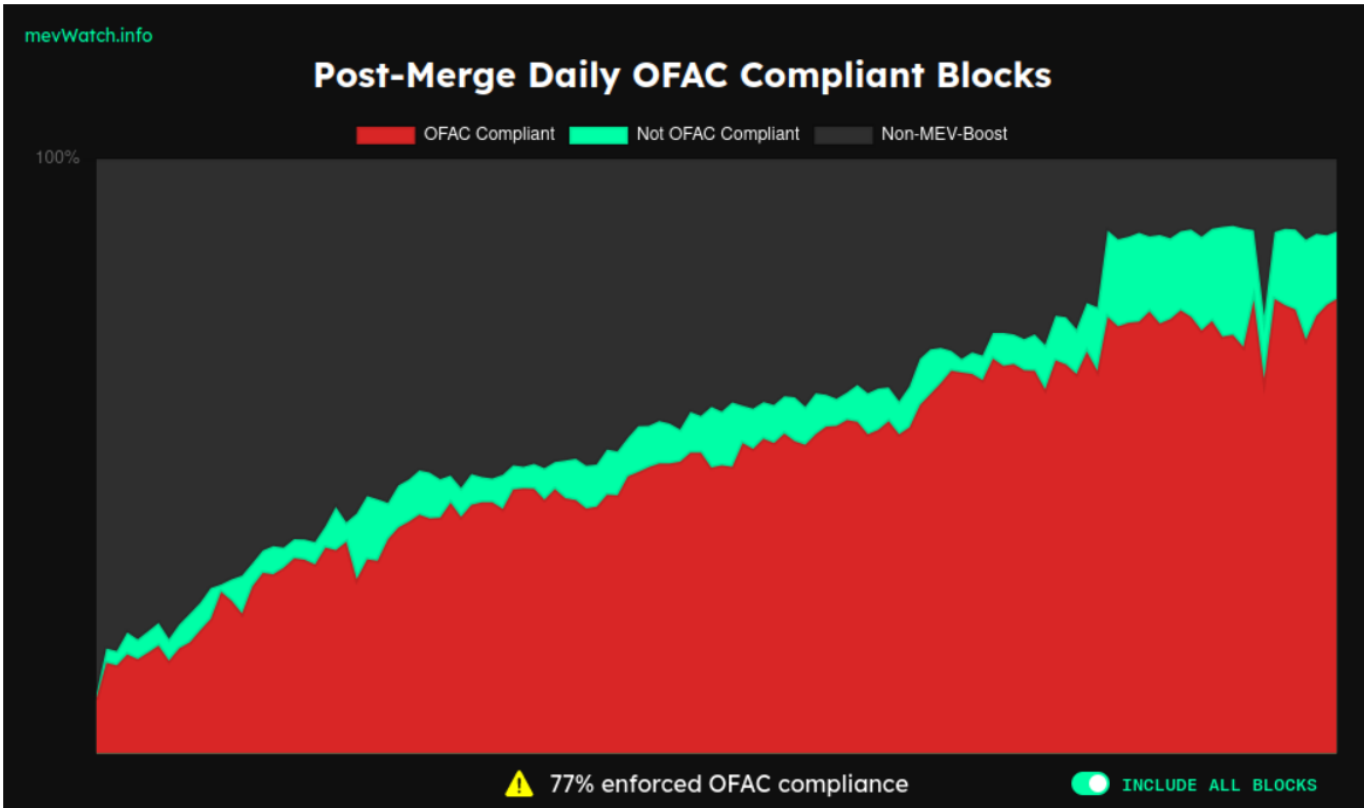


Beginner's Guide to mev-boost: https://writings.flashbots.net/writings/beginners-guide-mevboost/

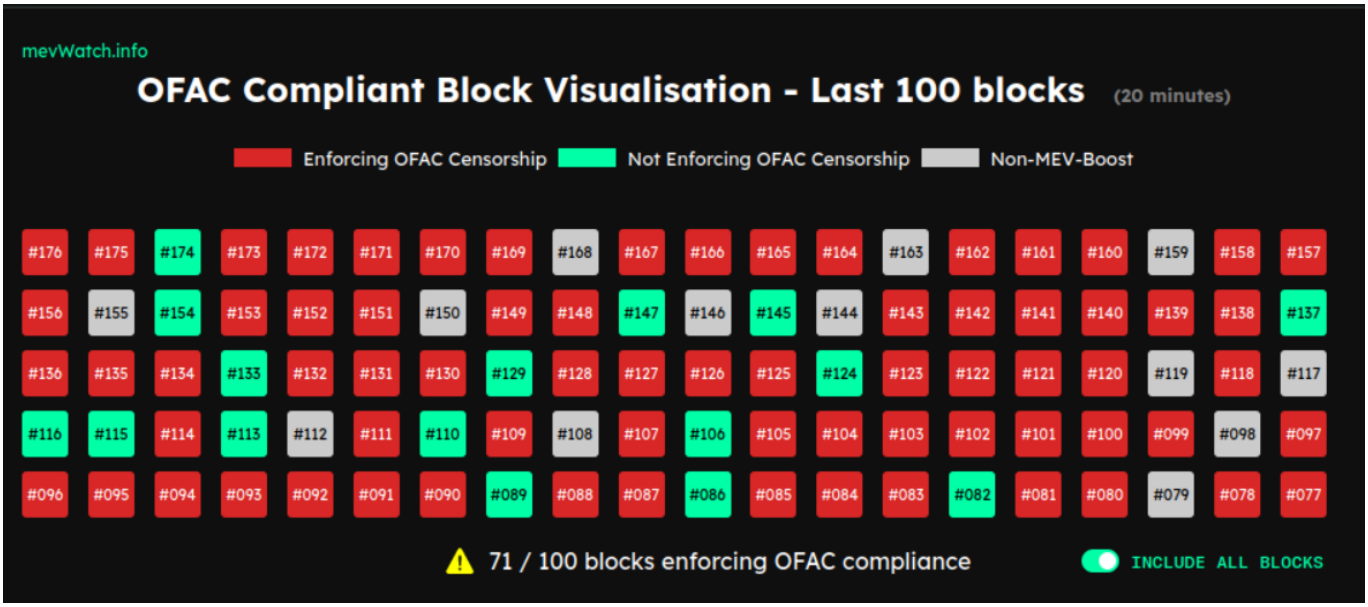
MEV-Boost, How Does it Work & What's Missing: https://www.youtube.com/watch?v=PS9SlHGJlrU

Censorship Risks

Only a handful of entities run relays and builders, and some of those entities have to abide by local regulators. Notably, several relays and builders, including Flashbots' own ones, filter out OFAC-sanctioned addresses. They will not include any transactions that interact with any smart contract or addresses specified by the US Treasury Department.



- Important to note that we only need a small number of non-censoring blocks to prevent censorship



Most agree that best mitigation for now is having relay diversification & long term to implement PBS.

Relay Operator	Filtering/Censorship	Market Share (Blocks Relayed)	Average Block Value (ETH)	Other Notes
Flashbots	Filters out OFAC sanctioned addresses	80.7%	0.144	• Will continue to be OFAC compliant in the future • 261k active validators • 31 unique builders
bloXroute Max Profit	None	7.6%	0.115	• Relay that propagates all available transactions/bundles with no filtering • 4 unique builders
bloXroute Ethical	Filters out generalized frontrunning and sandwich transactions	2.8%	0.082	• 3 unique builders
bloXroute Regulated	Filters out OFAC sanctioned addresses	1.8%	0.127	• 4 unique builders
Blocknative	Support OFAC compliance	2.7%	0.090	• 2 unique builders
Eden Network	Support OFAC compliance and other application regulations	2.2%	0.140	• 87k active validators
Manifold	None	2.1%	0.161	• 5 unique builders

Sources: mevboost.org, beaconcha.in, project websites as of 22 October 2022 AMBER

How to get started

Use flashbots with meta mask to protect against front running and sandwich attacks

- https://docs.flashbots.net/flashbots-protect/rpc/quick-start/
- https://docs.bloxroute.com/introduction/fast-protect

Listening to mempool

Using ether js we can listen to mempool

```
var ethers = require("ethers");
var url = "ADD_YOUR_ETHEREUM_NODE_WSS_URL";

var init = function () {
  var customWsProvider = new ethers.providers.WebSocketProvider(url);

  customWsProvider.on("pending", (tx) => {
    customWsProvider.getTransaction(tx).then(function (transaction) {
      console.log(transaction);
    });
  });

};

init();
```

Listen to contract events

```
const ethers = require("ethers");
const url = 'wss://eth-mainnet.g.alchemy.com/v2/...Key here...'
const abi = '[{....}]'

const customWsProvider = new ethers.providers.WebSocketProvider(url);
//Curve Aave stable Swap
//https://etherscan.io/address/0xDeBF20617708857ebe4F679508E7b7863a8A8EeE#code
const contract = new
ethers.Contract('0xDeBF20617708857ebe4F679508E7b7863a8A8EeE', abi,
customWsProvider)


contract.on("AddLiquidity", (provider, token_amounts, fees) => {
    console.log(provider, token_amounts, fees);
});
```

Send a single transaction via Flashbots

```
const signer = Wallet.createRandom()
const provider = new providers.JsonRpcProvider("http://localhost:8545")
const flashbotsProvider = await FlashbotsBundleProvider.create(provider,
signer)

const transaction = {
    from: signer.address,
    to: signer.address,
    value: "0x42",
    gasPrice: BigNumber.from(99).mul(1e9),
    gasLimit: BigNumber.from(21000),
}

const res = await flashbotsProvider.sendPrivateTransaction({
    transaction,
    signer,
}, {
    maxBlockNumber: (await provider.getBlockNumber()) + 5, // only allow tx
to be included for the next 5 blocks
});

const waitRes = await res.wait();
if (waitRes === FlashbotsTransactionResolution.TransactionIncluded) {
    console.log("Private transaction successfully included on-chain.")
} else if (waitRes === FlashbotsTransactionResolution.TransactionDropped) {
```

```
    console.log("Private transaction was not included in a block and has
been removed from the system.")
}
```

Flashbots data API

- Historical blocks api: https://blocks.flashbots.net/

Homework

1. See if you can listen to the mempool using ether.js (or similar web3.py etc)
2. Can you find a way to filter your mempool listener and get only uniswap transactions?
3. How might a you mitigate MEV & front running if you were building your own Dapp?
4. Have a look at the example sandwich bot and see how it works Repo.

Example Sandwich Bot Contract

(We do not encourage people to run these types of bots)

See Repo

Overview

From the README "In every Uniswap V2 trade, the user (victim) will specify a minimum amount of output tokens they're willing to receive.

The job of the sandwich bot is to calculate how much of the output tokens they should buy (to push the price of the token up) to match the victim's minimum out requirement. This minimum out requirement on most cases will be 2%, but on extreme cases it can be as high as 20% on volatile pairs (such as the SHIBA-WETH pair during the craze).

Once the sandwich bot has calculated the optimal number of tokens to buy, it'll wait for the victim to buy their tokens, and immediately sell to gain a profit."

The bot

1. Gets transactions from the mempool that are for the Uniswap router
2. Checks they don't have a receipt.
3. Parse the Uniswap data for 'swapExactETHForToken' transactions
4. Calculate a profitable sandwich
5. Work out what fees are likely to be involved
6. Submit the slices of the sandwich

Interestingly, the front and back slices are submitted via the Flashbots network.

Resources

- MEV-sbc Workshop: https://flashbots.notion.site/MEV-sbc-Workshop-6dcb3501d7e647b98b703b6312c3e95b
- Overview of MEV: https://amberlabs.substack.com/p/extractable-value
- General overview of Ethereum + PBS: https://members.delphidigital.io/reports/the-hitchhikers-guide-to-ethereum/

Other Approaches

Protocol Approaches

Some protocols have taken steps to prevent MEV such as using Verifiable Delay Functions in order to prevent gaming of transaction ordering, as Solana has done on its base layer to ensure transactions are ordered by time of arrival, or simply delegate ordering to something like Chainlink's Fair Sequencing Service (FSS)

Fair Sequencing Service

See Blog "In a nutshell, the idea behind FSS is to *have an oracle network order the transactions sent to a particular contract SC*, including both user transactions and oracle reports. Oracle nodes ingest transactions and then reach consensus on their ordering, rather than allowing a single leader to dictate it. Oracle nodes then forward the transactions to the contract *SC*. They sequence these transactions by attaching nonce or sequence numbers to them or sending them in batches."

You can read more about Fair Sequencing Services in the Chainlink 2.0 White paper (Section5): https://research.chain.link/whitepaper-v2.pdf

5.2 FSS Details

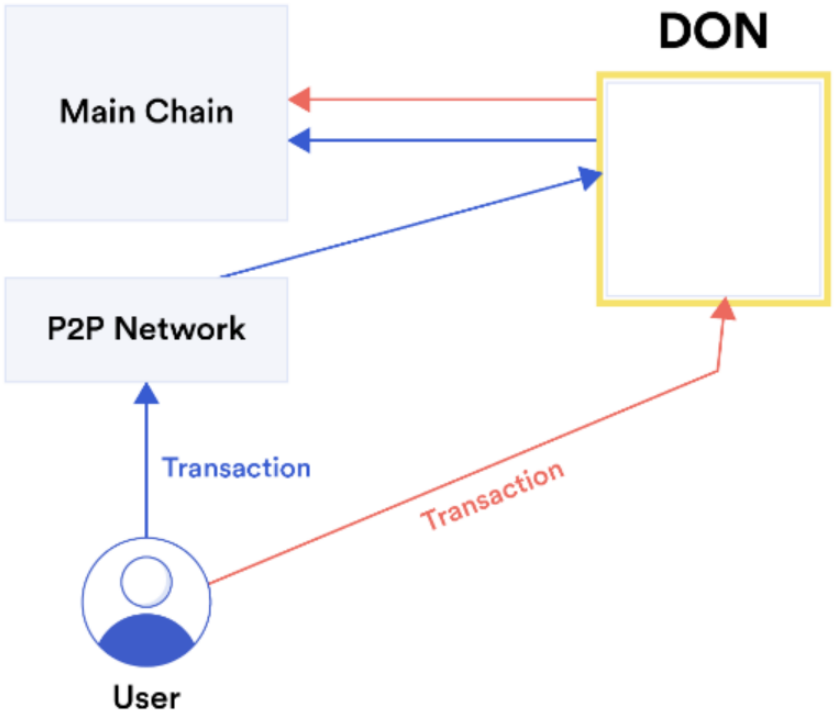


Figure 12: Order-fair mempool with two different transaction paths: direct and mempool-based.

1. Direct: The direct approach is conceptually simplest, but requires changes to user clients so that transactions are sent directly to the Decentralized Oracle 48 Network nodes, rather than to the nodes of the main chain. The DON collects user transactions destined to a specific smart contract SC and orders them based on some ordering policy. The DON then sends the ordered transactions to the smart contract on the main chain. Some ordering mechanisms also require the direct approach because the user that creates a transaction must cryptographically protect it before sending it to FSS.

2. Mempool-based: To facilitate the integration of FSS with legacy clients, the DON can use Mempool Services (MS) to monitor the main chain's mempool and collect transactions.

MEV on L2s

MEV lanscape is complex and rapidly evolving. Most L2 are actively working on upgrades to mitigate bad MEV and/or ensure MEV remains decentralised.

L2s and their approaches:

L2	Front Running	MEV mitigation	Status	Overview	Description
Arbitrum	No	Fair Sequencing Service	Not implemented but currently sequencing is centralised to prevent MEV	Oracles ensure TX order is fair	Decentralised Oracle Network used to ensure TX order is first in first out.
Optimism	Some	MEV Auction (MEVA) is in design but not implemented sequencing is centralised to prevent MEV	Not implemented	Elections contract for sequencers	Sequencers are elected by a smart contract managed auction run by the block producers called the MEVA contract. This auction assigns the right to sequence the last N transactions.
Polygon	Yes	A flashbots implementation has been deployed on Polygon	Live but not battle tested	Link	

Arbitrum

See Hashflow x Arbitrum AMA The MEV lanscape on Arbitrum is very different to Ethereum/L1s. In general the Arbitrum system tries to mitigate MEV.

- Arbitrum has no mempool.
- Users send Tx directly to sequencers, Bots and the public can only see the transaction once it has been confirmed.
- In theory the sequencer could re-order transactions and extract value through MEV however Arbitrum currently runs the sequencer to prevent this.
- Arbitrum has plans to move towards a fair sequencing system which will decentralise the sequencing through a fair ordering consensus mechanism.

One interesting challenge is that consensus alone doesn't guarantee a 'first come first served' order for TXs. Sequencers could conceivable reach consensus on the most profitable order not the fairest order. Therefore Arbitrum has work extensively with Chainlink to research Fair Sequencing Services. Additionally research has been done into fair ordering consensus mechanisms such as Aequitas

3 Modes of Arbitrum

1. Single Sequencer: L2 MEV-Potential (Mainnet Beta)

For Arbitrum's initial, flagship Mainnet beta release, the Sequencer will be controlled by a single entity. This entity has transaction ordering rights within the narrow / 15 minute window; users are trusting the Sequencer not to frontrun them.

2. Distributed Sequencer With Fair Ordering: L2-MEV-minimized (Mainnet Final Form)

The Arbitrum flagship chain will eventually have a distributed set of independent parties controlling the Sequencer. They will collectively propose state updates via the first BFT algorithm that enforces fair ordering within consensus (Aequitas). Here, L2 MEV is only possible if >1/3 of the sequencing-parties maliciously collude, hence "MEV-minimized."

3. No Sequencer: No L2 MEV

A chain can be created in which no permissioned entities have Sequencing rights. Ordering is determined entirely by the "Inbox" contract on L1. No party involved in L2, including Arbitrum validators, has any say in transaction ordering, and thus no L2 MEV enters the picture.

source

Optimism

Optimism is fairly similar to Arbitrum in that it relies on a centralised sequencer to prevent transaction reordering but is researching and building a decentralised sequencing system.

The Optimism thesis is:

- Prevent what MEV we can
- Democratise & extract the rest
- Redirect this as protocol revenue (retroactive public goods funding)

How Optimism works now:

- Optimism block production is primarily managed by a single party, called the "sequencer,"
- There is no mempool
- Transactions are immediately accepted or rejected in the order they were received

Is transaction front running possible on Optimism?

> Right now front running is very difficult. You can't just do it by offering a higher transaction fee, because Optimism transactions are priced by the sequencer. In theory, you could do it by breaking into one of the routers between the sending system and the Optimism sequencer and dropping packets, but if you can do that you have better things to do than front running transactions.

> Once we decentralize the sequencer, whoever runs the sequencer would be able to determine the order of transactions, so some front running might be possible.

Instead of a Fair Sequencing Consensus mechanism Optimism plans to auction the right to sequence transations thereby redirecting the MEV value back into the protocol & toward retroactive public goods.

MEV Auctions on Optimism

Implementing the Auction

The auction is able to extract MEV from miners by separating two functions which are often conflated:

1. Transaction inclusion; and
2. Transaction ordering.

In order to implement our MEVA we can define a role for each function.

1. Block producers which determine transaction inclusion, and
2. Sequencers which determine transaction ordering.
3. Block Producers Block proposers are most analogous to traditional blockchain miners. It is critical that they preserve the censorship resistance that we see in blockchains today. However, instead of proposing blocks with an ordering, they simply propose a set of transactions to eventually be included before N blocks.

4. Sequencers Sequencers are elected by a smart contract managed auction run by the block producers called the MEVA contract. This auction assigns the right to sequence the last N transactions. If, within a timeout the sequencer has not submitted an ordering which is included by block proposers, a new sequencer is elected.

Sequencers and Instant Transaction Inclusion

In addition to extracting MEV, the MEVA provides the current sequencer the ability to provide instant cryptoeconomic guarantees on transaction inclusion. They do this by signing off on an ordering immediately after receiving a transaction from a user – even before it is sent to a block producer. If the sequencer equivocates and does not include the transaction at the index which they promised, the user may submit a fraud proof to the MEVA contract to slash the sequencer. As long as the sequencer stands to lose more than it can gain from an equivocation, we can expect the sequencer to provide realtime feed of blockchain state which can be monitored, providing, for instance, realtime price updates on Uniswap.
Source

Starknet MEV

Sequencers can reorder transactions but currently Sequencing is not public/decentralized.

Background

- Pre defi summer, consensus was simple: Miners competed for block rewards & transactions fees.
- Over time complexity increased with
- 2020/2021 - Dark forest idea
- Flashbots -
- Merge
- Mev Boost
- Post merge PBS.

Implications

- MEV is an important topic to understand for the long term success of ethereum existential
- For dapp builders it's important to understand so you can design apps around issues

---