Maximal Extractable Value (MEV) and Fair Ordering

Dan Boneh and Valeria Nikolaenko

Searchers

Ethereum gives rise to a new type of business: **searchers**

- **Arbitrage:** Uniswap DAI/USDC exchange rate is 1.001
  whereas at Sushiswap the rate is 1.002

  ⇒ a searcher posts Tx to equalize the markets and profits

- **Liquidation**: suppose there is a liquidation opportunity on Aave

  ⇒ a searcher posts a liquidation Tx and profits

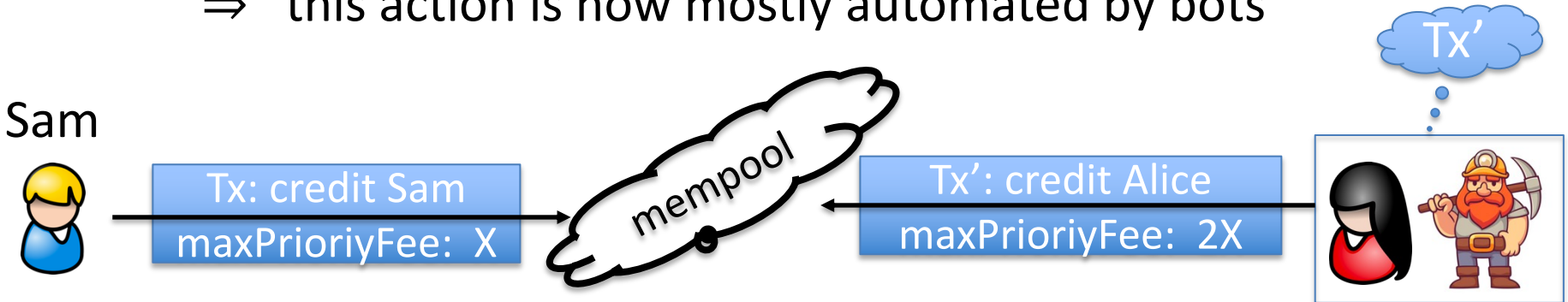
- Many other examples … often using a sequence of Tx (a bundle)

The MEV problem

What happens when a searcher posts a Tx to the mempool?

- **Validator:** create a new Tx' with itself as beneficiary, and place it before Sam's Tx in the proposed block
- **Another searcher:** create a new Tx' with itself as beneficiary, and posts it with a higher *maxPrioriyFee*

⇒ this action is now mostly automated by bots

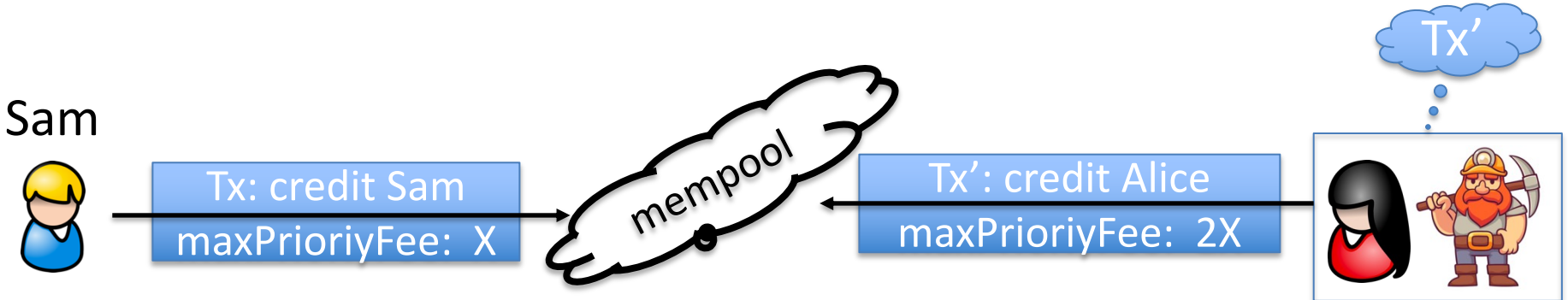


The result harms honest users

Price Gas Auctions (PGA): two or more searchers compete

- Repeatedly submit a Tx with higher and higher *maxPriorityFee* until a validator chooses one … happens within a few seconds

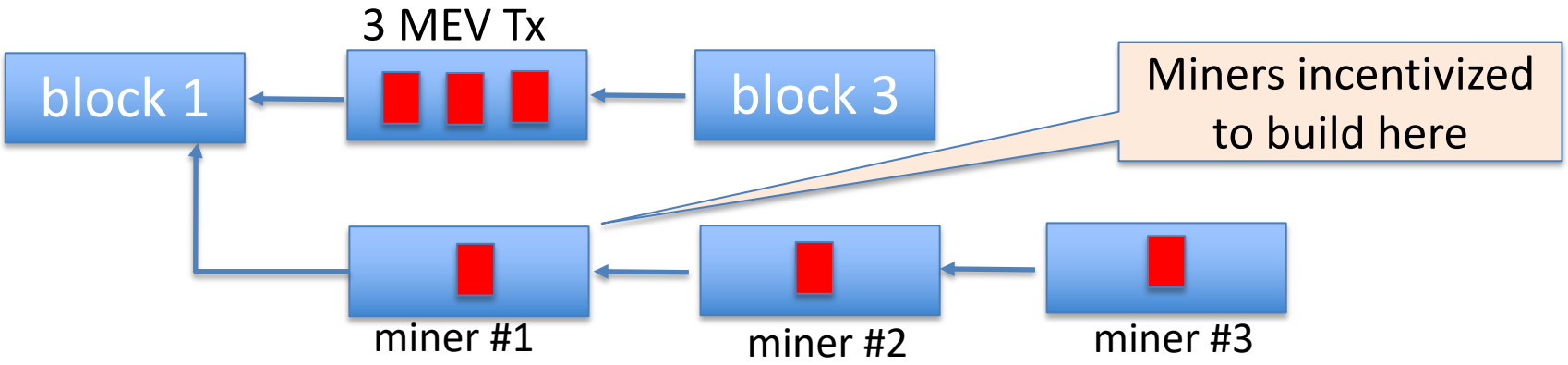
⇒ causes congestion (lots of Tx in mempool) and high gas fees



The result harms consensus

Undercutting attack on <u>longest-chain</u> consensus:

Rational miner: can cause a re-org by taking one MEV Tx for itself and leave two for other miners



The problem: MEV Tx generate extra revenue for miners, higher than block rewards

The result causes centralization

Validators can steal MEV Tx from searchers ⇒ **Private mempools**

Searchers only send Tx to a validator they trust

(have a business relation with)

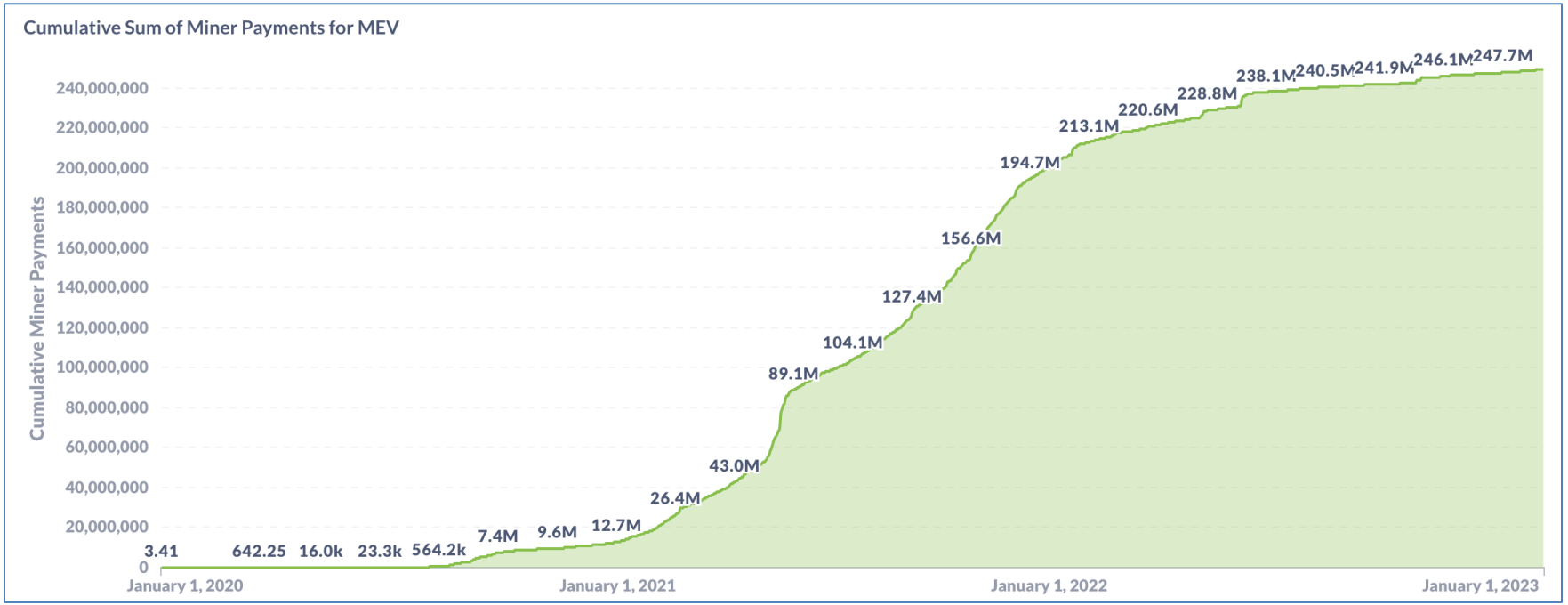
These validators do not propagate Tx to the network,

but put them in blocks themselves

In the long run: a few validators will handle the bulk of all Tx

How big are MEV rewards?

MEV payments to validators:



source: explore.flashbots.net

What to do??

Proposer Builder Separation (PBS)

Goals:

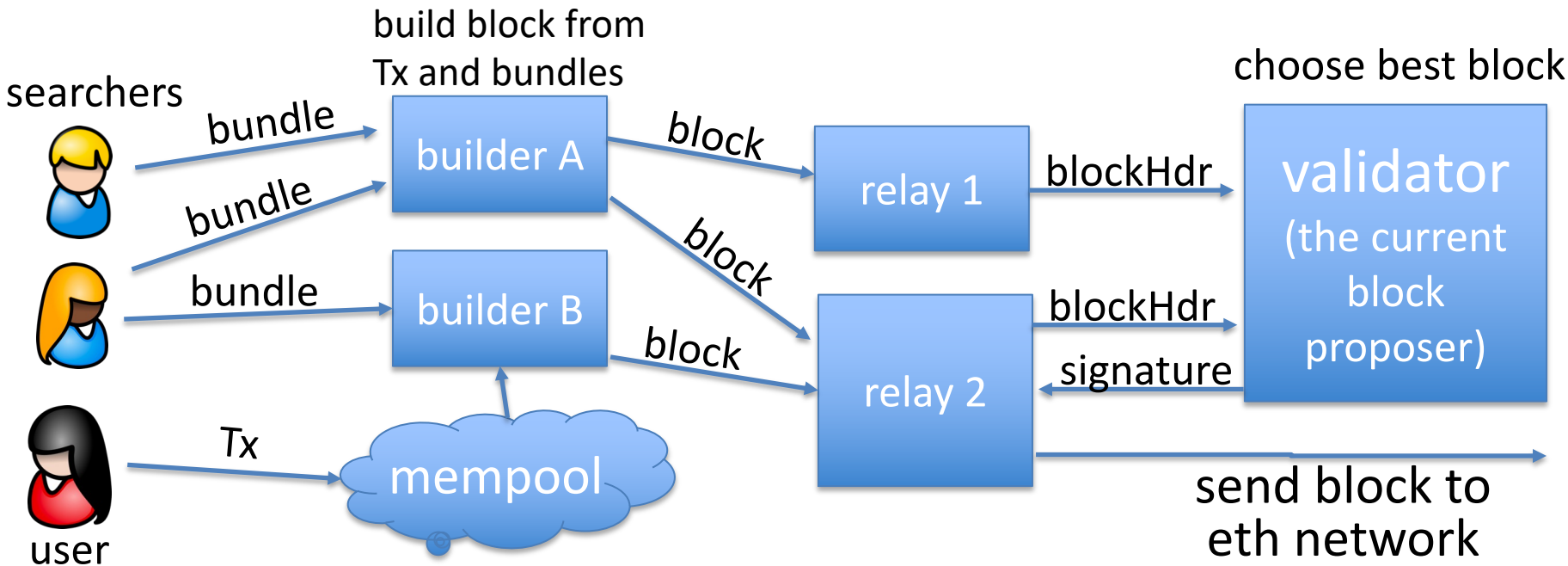
- Eliminate price gas auctions in the public mempool
  - Instead, create an off-chain market for searchers to compete on the position of their bundles in a block
- Prevent validator concentration: make it possible for every validator to earn MEV payments from searchers

Current PBS implementation: **MEV-boost**

The participants in PBS (as in MEV-boost)

Users have Tx and searchers have bundles (sequence of Tx)

- searcher wants its bundle posted in a block unmodified



MEV-boost

Builder: collects bundles and Tx, builds a block (≈300 bundles/block)

- includes a MEV offer to validator (feeRecipient)

Relay: collects blocks, chooses block with max MEV offer

- sends block header (and MEV offer) to block proposer
- Can't expose Tx in block to proposer (proposer could steal Tx)

Proposer: chooses best offer and signs header with its staking key

⇒ Then Relay sends block to network, making it public

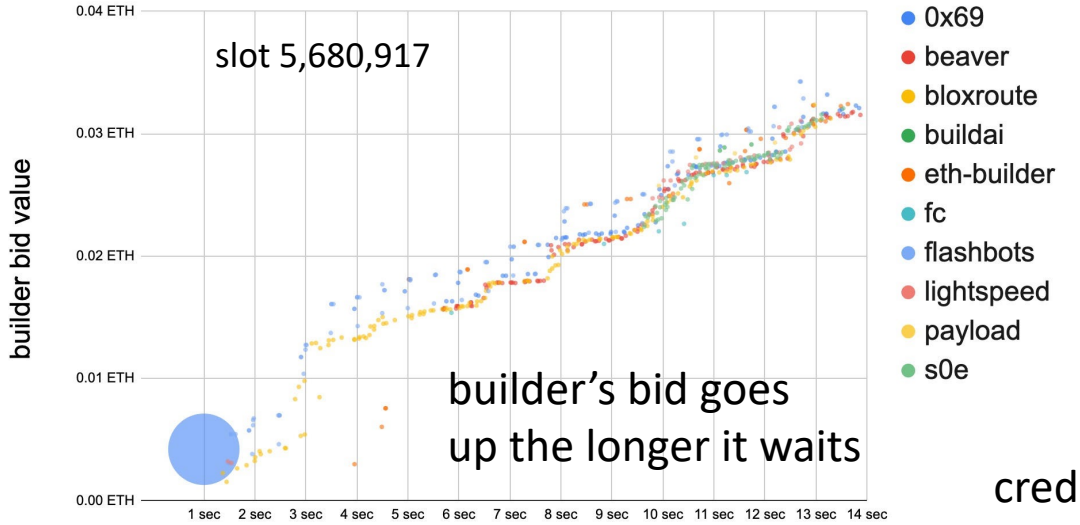
⇒ Now, proposer cannot steal MEV (would be exposed to slashing)

https://writings.flashbots.net/searching-post-merge

Many block options per slot

A relay might receive 500 blocks per slot from builders

- Each builder might send 20 blocks to relay for one slot
- Why? The longer builder waits the more MEV opportunities …



credit: Justin Drake and Shea Ketsdever

Operating relays

Flashbots: Filters out OFAC sanctioned addresses,
aims to maximize validator payout
(so that many validators will work with it)

BloXroute: no censorship
aims to maximize validator payout

UltraSound: not for profit, non censoring

...

An example: flashbots relay

Recently Delivered Payloads

fee to validator

Epoch	Slot	Block number	Value (ETH)	Num tx
165,046	5,281,503	16,115,184	0.0759673152	186
165,046	5,281,501	16,115,182	0.05098935853	142
165,046	5,281,499	16,115,180	0.1902791095	167
165,046	5,281,498	16,115,179	0.103438972	295
165,046	5,281,496	16,115,177	0.07159735143	199
165,046	5,281,495	16,115,176	0.04034671944	125

An example: flashbots relay

Epoch:	165,046
Slot:	**5,281,503**
Block Number:	**16115184**
MEV Reward Recipient:	0xebec795c9c8bbd61ffc14a6662944748f299cacf
MEV Block Reward:	0.07596 Ether

address of validator who proposed the block

Are we done? Not quite …

Builder concentration: three builders build 75% of all blocks !!

- Clear centralization in the builder market
- Enables censorship by builders

(builder0x69,beaverbuild,Flashbots)

Proposers hold all the power (First price auction among builders)

⇒ Most MEV profits flow to proposers

MEV-boost is not designed for cross-chain MEV

- For cross-chain arbitrage, no atomicity guarantee for bundle

The next step: SUAVE

Goals:

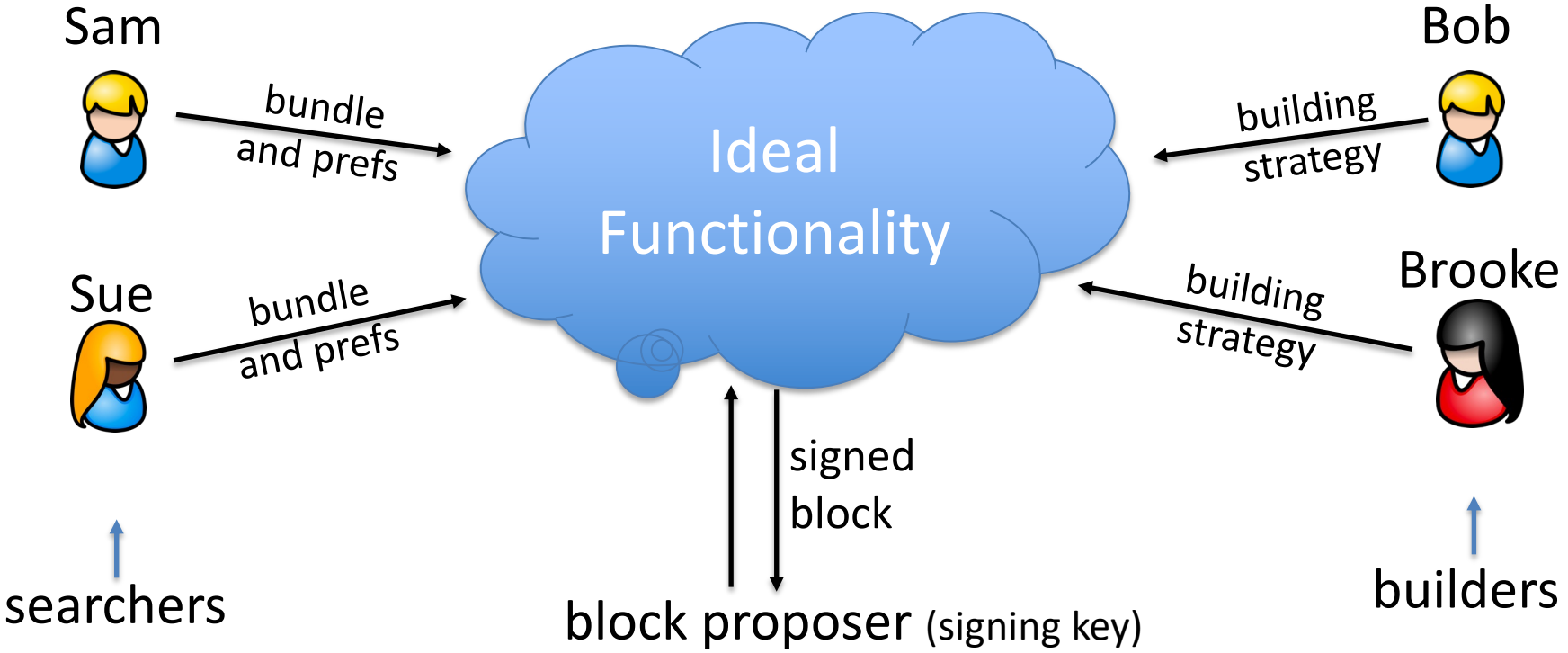
- Tx should be private (encrypted) until signed by block proposer

  ... but should be available to all block builders to build blocks

Seems contradictory! crypto to the rescue:

⇒ requires a massive MPC or secure HW enclaves

The SUAVE Multiparty Computation



Fair Ordering of Transactions

MEV - accept or try to stop?

Two schools of thought:

1. **Accept MEV** as inevitable and develop processes to democratize MEV extraction:
   a. efficient (no gas wars),
   b. decentralized (no censorship) and
   c. transparent (keep clients' trust)

2. **Stop MEV** with techniques that make order manipulation impossible or very costly

How can we stop MEV?

1. **Applications-specific MEV prevention**
   (e.g. automatically collect arbitrage, A2MM)
   Downside: application-specific
2. **Trusted execution environments (TEEs)** to order transactions
   Downside: hardware assumption
3. **Randomize transactions before executing**
   Downside: spamming with identical extracting transaction
4. **Time-Based Order-Fairness**
5. **Blind Order-Fairness**

Causal ordering: a transaction tx2 derived from tx1 should not be sequenced before tx1.

Time-Based Order-Fairness: Aequitas

Intuitively: if most (γ) miners received tx1 before tx2, then tx1 should precede tx2 in the final ordering.

Challenge of Condorcet cycles:
* miner #1: [tx1, tx2, tx3]
* miner #2: [tx2, tx3, tx1]
* miner #3: [tx3, tx1, tx2]

A majority received (tx1 before tx2) AND (tx2 before tx3) AND (tx3 before tx1)!

Solution: place cycles that can't be resolved in the same block.

Block-Order-Fairness: If tx1 was received before tx2 by most miners, then tx1 will be placed in the same block with tx2 or in some preceding block.

[Kelkar-Zhang-Goldfeder-Juels-2020]

Time-Based Order-Fairness: Aequitas

Block-Fair-Ordering protocol idea:

1. Miners broadcast their order preferences.
2. Miners agree on the subset of miners whose orderings to consider.
3. Build a graph of transactions:
   a. Vertices = transactions present in a large number of orderings,
   b. Edge(tx1 → tx2) = tx1 stands before tx2 in most orderings.
4. Collapse strongly connected (CSC) components.
5. Topologically sort.
6. Final ordering respects the sort.

[Kelkar-Zhang-Goldfeder-Juels-2020]

More Time-Based Order-Fairness Protocols

- Large communication: **O(n^2)**
- Does not mitigate **attackers with better connectivity**
- **Weaker definitions**

Protocol	Transaction Ordering	Comm. Complexity Optimistic	Comm. Complexity Worst	Corruption	Liveness	Censorship Resistance	Synchronized Clocks?
Aequitas [19]	γ-batch-order-fairness (Definition III.1)	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$n > \frac{4f}{2\gamma-1}$ (5)	Weak	Yes	No
Wendy [21]	Timed-Relative-Fairness(1) (Section VI-A)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n \geq 3f+1$	Standard	Yes	Yes(2)
Pompē [38]	Ordering Linearizability(1) (Section VI-A)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n \geq 3f+1$	Standard	No	Yes
Quick-Fairness [9]	κ-differential order-fairness(3)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n \geq 3f+\kappa+1$	Only when all nodes are honest		No
Themis (This Work)	γ-batch-order-fairness(4)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n > \frac{4f}{2\gamma-1}$ (5)	Standard	Yes	No
SNARK-Themis (This Work)		$\mathcal{O}(n)$	$\mathcal{O}(nf)$				

From "Themis: Fast, Strong Order-Fairness in Byzantine Consensus" by Kelkar-Deb-Long-Juels-Kannan 2021

Blind Order-Fairness

Three phases:

- **Commit transactions**: users commit to their transactions.
- **Order commitments**: validators order commitments into a block.
- **Reveal transactions**: commitments are revealed (by users themselves, or by validators, or “automatically”).

Blind Order-Fairness

Three phases:

- **Commit transactions (by users)**
- **Order commitments (by validators)**
- **Reveal transactions (by ?)**

Solution #1 (warm-up) - collateral based commitments

- **Commit (tx)**:
  - Lock collateral
  - Output ct = Commit(tx)
- **Reveal (ct) (by users)**:
  - User reveals tx = Open(ct), otherwise loses collateral

Blind Order-Fairness

Three phases:

- **Commit transactions (by users)**
- **Order commitments (by validators)**
- **Reveal transactions (by ?)**

Solution #2 - threshold cryptography:

- Setup: validators generate pk, threshold share a secret key sk
- **Commit (tx)**:
  - Output ct = Encrypt(pk, tx)
- **Reveal (ct) (by validators)**:
  - Validators run MPC: tx = Decrypt(sk, ct)

Reiter-Birman-1994

Cachin-Kursawe-Petzold-Shoup-2001

Blind Order-Fairness

Three phases:

- **Commit transactions (by users)**
- **Order commitments (by validators)**
- **Reveal transactions (by ?)**

Solution #3 - secret dissemination

- **Commit (tx)**:
  - Generate symmetric secret key k
  - Using IDA share k to validators
  - Output ct = Encrypt(k, tx)
- **Reveal (ct) (by validators)**:
  - Validators run MPC: tx = Decrypt(k, ct)

Malkhi-Szalachowski-2022

Blind Order-Fairness

Three phases:

- **Commit transactions (by users)**
- **Order commitments (by validators)**
- **Reveal transactions (by ?)**

Solution #4 - time-lock-puzzle commitments

Example: trapdoor Verifiable Delay Functions - tVDF

- **Commit (tx)**:
  - Generate tVDF parameters: (pp, msk)
  - VDF.Eval_quickly(msk, Δ, x) → k // takes constant time
  - Output ct = [pp, Encrypt(k, tx)]
- **Reveal (ct) (by anybody)**:
  - Computes (anybody) VDF.Eval_slowly(pp, x) → k // takes time Δ
  - Output tx = Decrypt(k, ct)

How can we stop MEV?

1. **Applications-specific MEV prevention**
   (e.g. automatically collect arbitrage, A2MM)
   Downside: application-specific
2. **Trusted execution environments (TEEs)** to order transactions
   Downside: hardware assumption
3. **Randomize transactions list for execution with a randomness beacon**
   Downside: spamming with identical extracting transaction
4. **Time-Based Order-Fairness**
   Downside: yet practically inefficient, not preventing well connected extractor
5. **Blind Order-Fairness**
   Downside: threshold cryptography or VDFs, does not prevent front-running
6. **More ideas?**

How can we stop MEV?

1. **Applications-specific MEV prevention**
   (e.g. automatically collect arbitrage, A2MM)
   Downside: application-specific
2. **Trusted execution environments (TEEs)** to order transactions
   Downside: hardware assumption
3. **Randomize transactions list for execution with a randomness beacon**
   Downside: spamming with identical extracting transaction
4. **Time-Based Order-Fairness**
   Downside: yet practically inef g well connected extractor
5. **Blind Order-Fairness**
   Downside: threshold cryptog ot prevent front-running
6. **More ideas?**

THE END