CS251 Fall 2025

(cs251.stanford.edu)

Maximal Extractable Value (MEV)

Dan Boneh

HW#3 posted, guest next lecture (please come in person)

Where we are in the course

- How consensus protocols work
- **Bitcoin**: the UTXO model, and the Bitcoin scripting language
- **Ethereum** (the blockchain computer): the EVM and Solidity

Current topic: **decentralized finance**

on-chain: lending, exchanges, stablecoins, today: MEV

Next: privacy on the blockchain, scaling the blockchain, and interoperability across blockchains

Decentralized Finance (DeFi)

- **Permissionless**: any financial instrument can be implemented and deployed with a few lines of Solidity code

(a centralized system could refuse to deploy a competing service)

- **Transparent**: Dapp code and Dapp state are public

  $\Longrightarrow$ Anyone can inspect and verify

- **Composable**: Dapps can call one another

  ERC-20 standard enables interoperability (6 functions)

Why DeFi? Failures of the existing financial system

- **Cross border inefficiency**:

  send $10 to south america ⇒ 36% fees

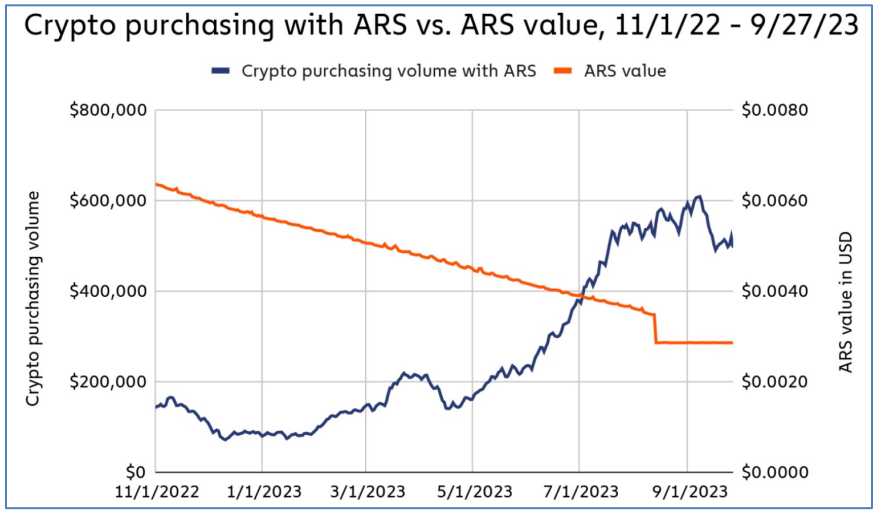
- **The high cost of being poor in america:**

  In 2019, **5.4 percent** of US households were unbanked

- **Economies with an unstable fiat currency**

- **Patchwork of technologies developed over time**

Why DeFi? Failures of the existing financial system



USDC/USDT daily purchasing volume in Argentina during inflation

> "As crypto adoption has grown, lots of people [in Argentina] will now get their paycheck and immediately put it into USDT or USDC."
>
> Alfonso Martel Seward, Lemon Cash

https://www.chainalysis.com/blog/latin-america-cryptocurrency-adoption/

Maximal Extractable Value (MEV)

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

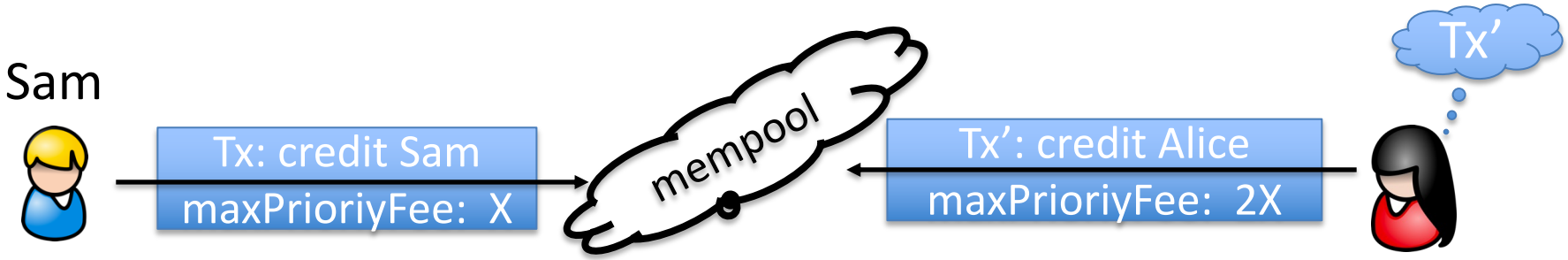
- Many other examples … often using a sequence of Tx (a bundle)

The MEV problem

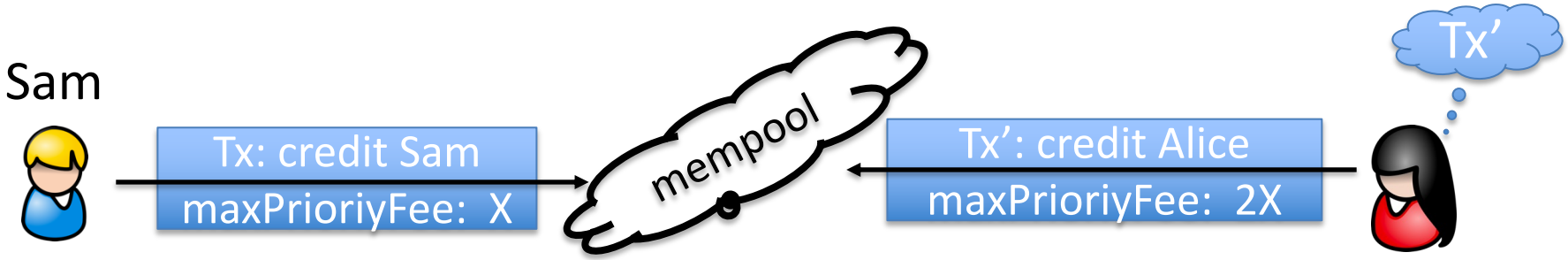
What happens when a searcher posts a Tx to the mempool?

- **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 copy-paste bots



The MEV problem

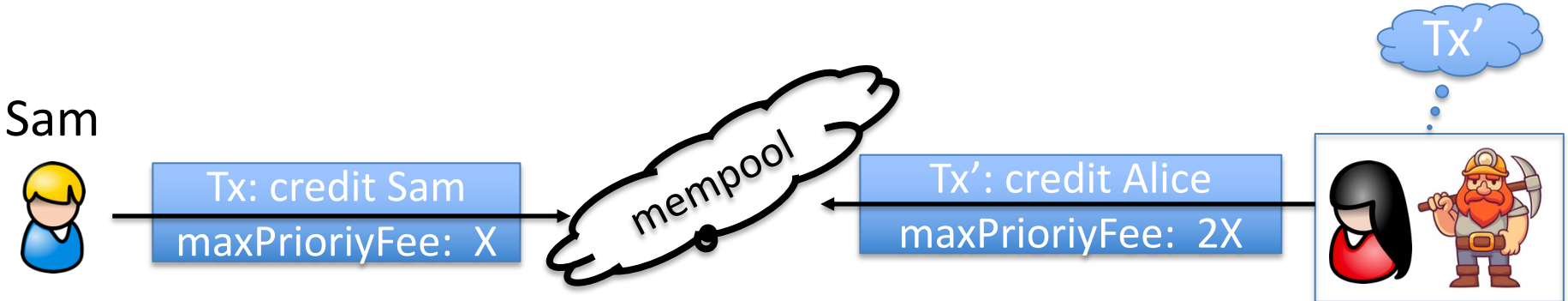


The result harms honest users

Price Gas Auctions (PGA): many searchers compete

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

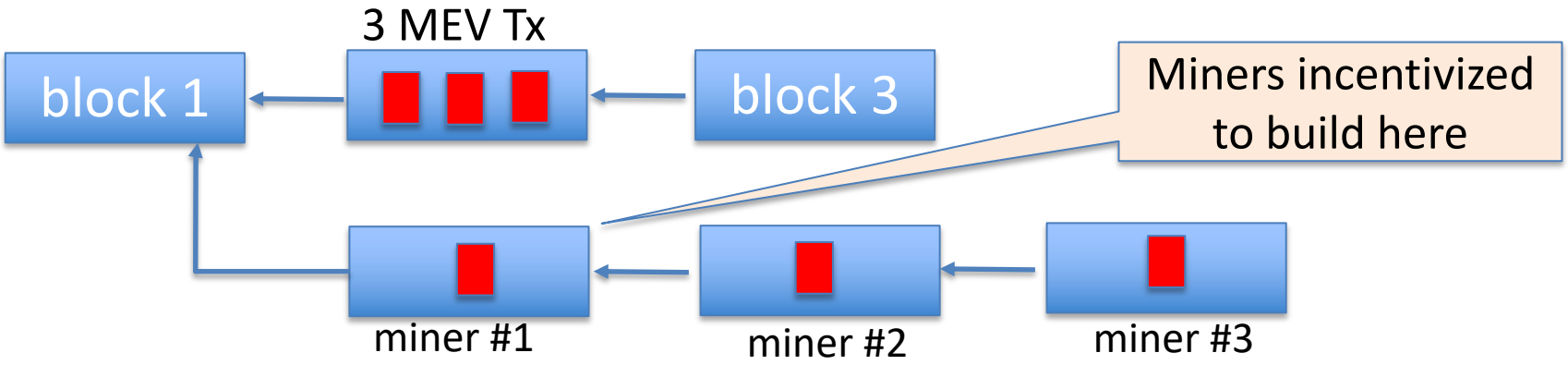
⇒ causes congestion (lots of Tx in mempool) and high gas fees



The result harms consensus

Undercutting attack on <u>longest-chain</u> consensus (not Ethereum)**:**

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

What to do??

Two options

Option 1:

- Accept MEV is unavoidable; minimize its harm to the ecosystem

  ⇒ Proposer-Builder Separation (PBS)

Option 2: Try to prevent some MEV:

- Hide Tx data while Tx is in the mempool, or
- Remove the block proposer's choice in ordering Tx in a block

Option 1: Proposer Builder Separation (PBS)

Goals:

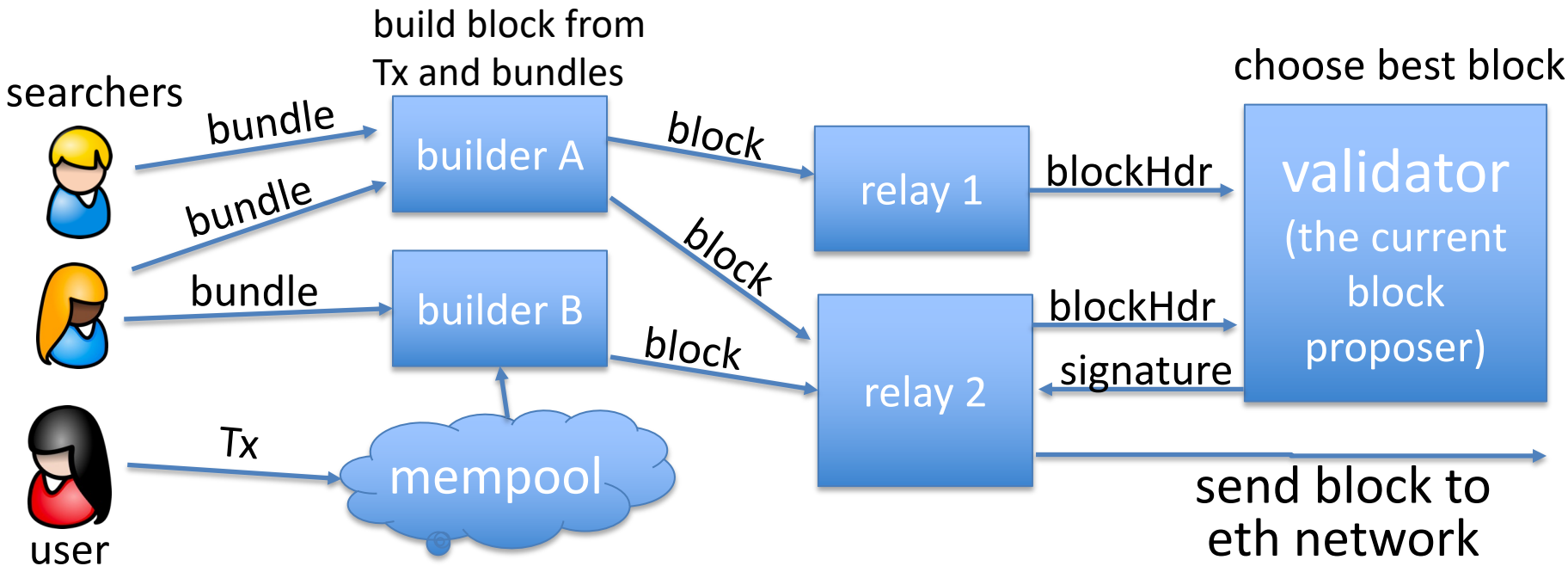
- Eliminate price gas auctions in the public mempool
  - Instead, create an off-chain market for searchers to compete on the position of their bundles in a block
- Prevent validator concentration: make it possible for every validator to earn MEV payments from searchers

Current PBS implementation: **MEV-boost** (a sidecar for beacon nodes)

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

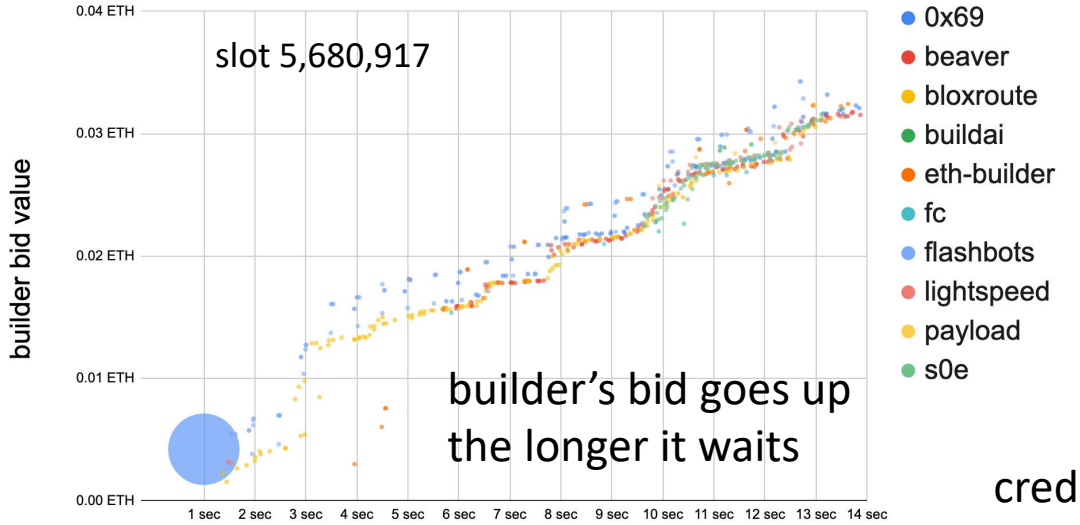
⇒ Now, proposer cannot steal MEV (would be exposed to slashing)

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: (runs two identical relays) same as Flashbots relay

UltraSound (the largest relay): not for profit, non censoring

...

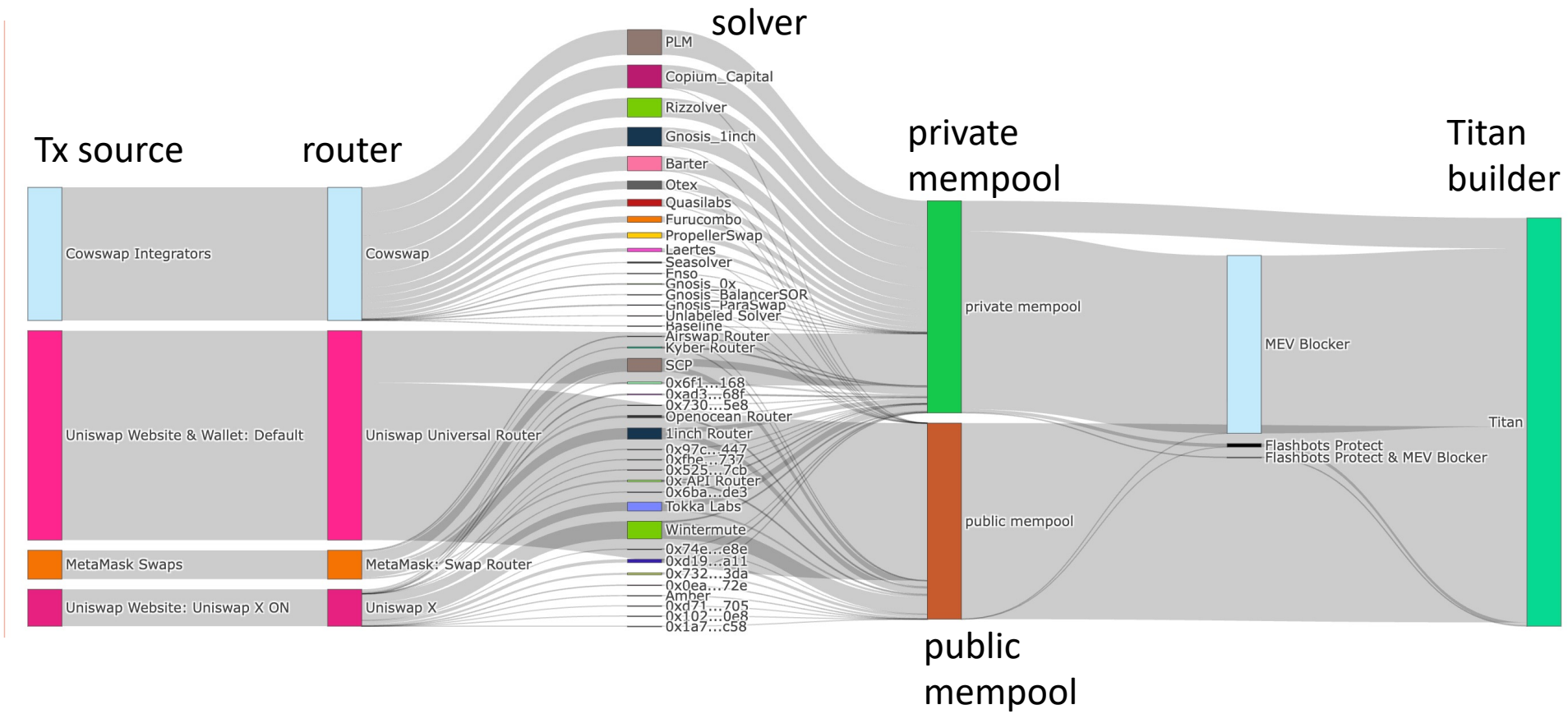
An example: flashbots relay

Recently Delivered Payloads

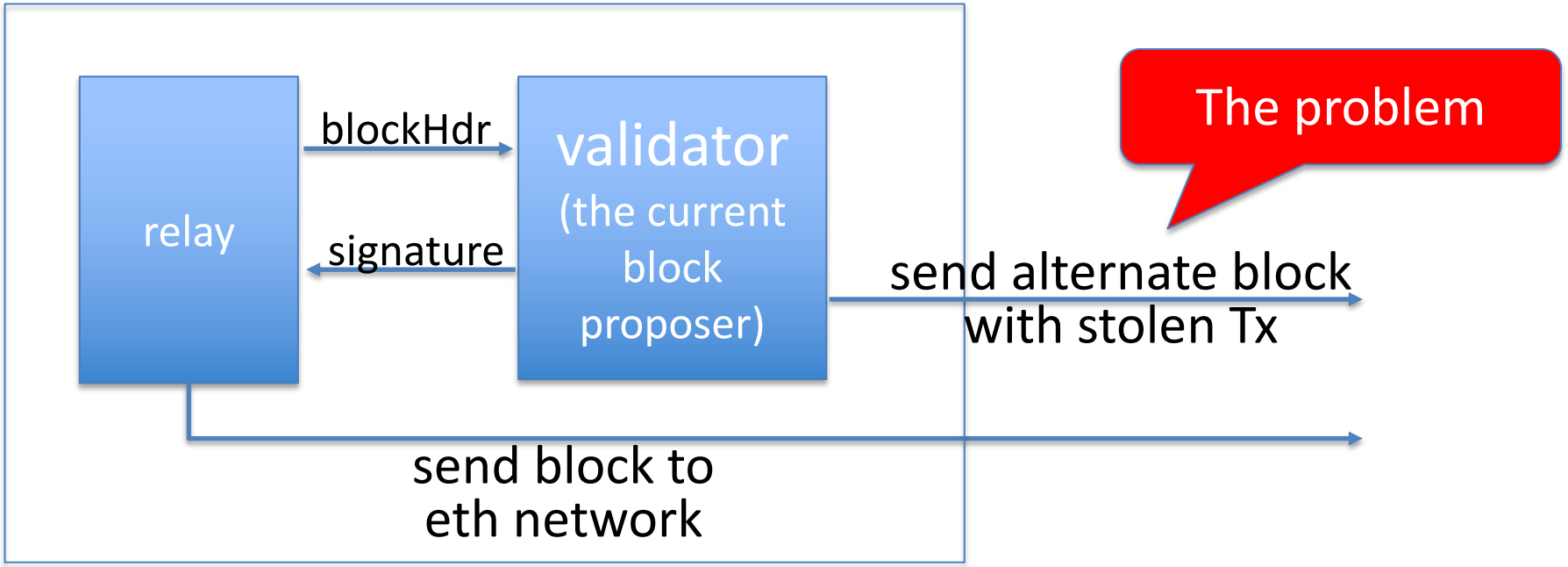
fee to validator

Epoch	Slot	Block number	Value (ETH)	Num tx
402,851	12,891,242	23,664,717	0.006264077745	227
402,851	12,891,237	23,664,712	0.006611980236	119
402,851	12,891,232	23,664,707	0.01331538742	195
402,850	12,891,209	23,664,684	0.01029033649	210
402,849	12,891,194	23,664,669	0.01113683151	202
402,849	12,891,191	23,664,666	0.009631052538	185

Transaction flow: an example



The race problem (exploited in Apr. 2023)



Block proposer will be slashed (why?) ⇒ Lose 1 ETH

… but can gain much more in stolen MEV.

Are we done? Not quite …

Builder concentration: three builders build 88% of all blocks !!

- Centralization in the builder market
- Enables censorship by builders

(Titan Builder, BuilderNet, Quasar)

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

⇒ Most MEV profits flow to block proposers

a potential solution: Multiple Concurrent Proposers (MCP)

MEV-boost is not designed for cross-chain MEV

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

The next step: BuilderNet

Goal: decentralized block building

Why? Better block building when builders share order flow

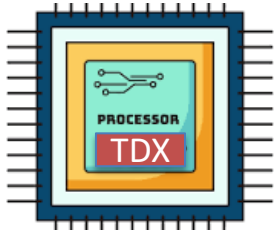
… but transactions should remain private until sent to relay

… even builders should not see the transactions.

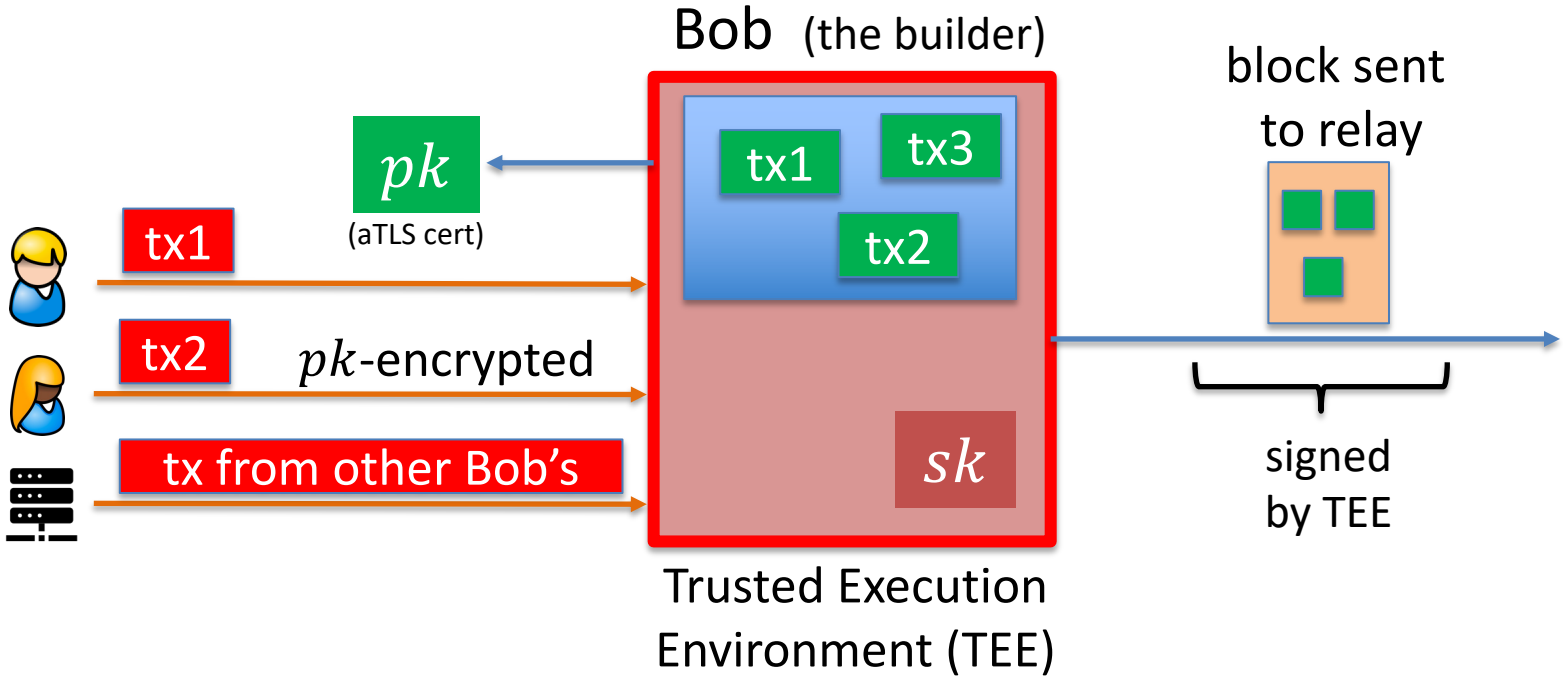
How? Trusted Execution Environment (TEE), such as Intel TDX

- Code and data inside enclave are not visible outside
- Not even to the OS or a malicious admin

Is TDX a strongly secure TEE? Well, not quite …



Block Building inside of a TEE (way simplified)



Main point: building strategy is public and attested (enforced by TEE)
Tx remain hidden in TEE until sent to relay

Reducing MEV Using an Encrypted Mempool

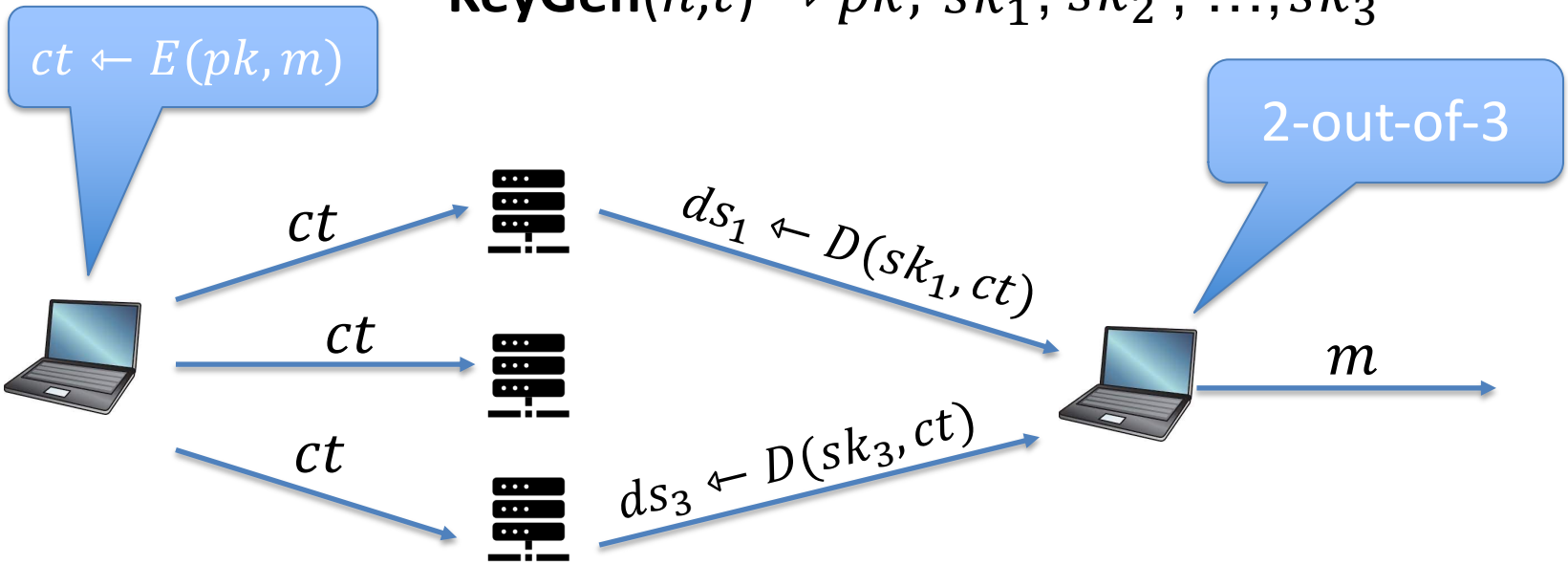
Goal: Tx is only decrypted after it is confirmed on chain

… by then it is too late to front run it

How? Threshold decryption

What is threshold decryption?

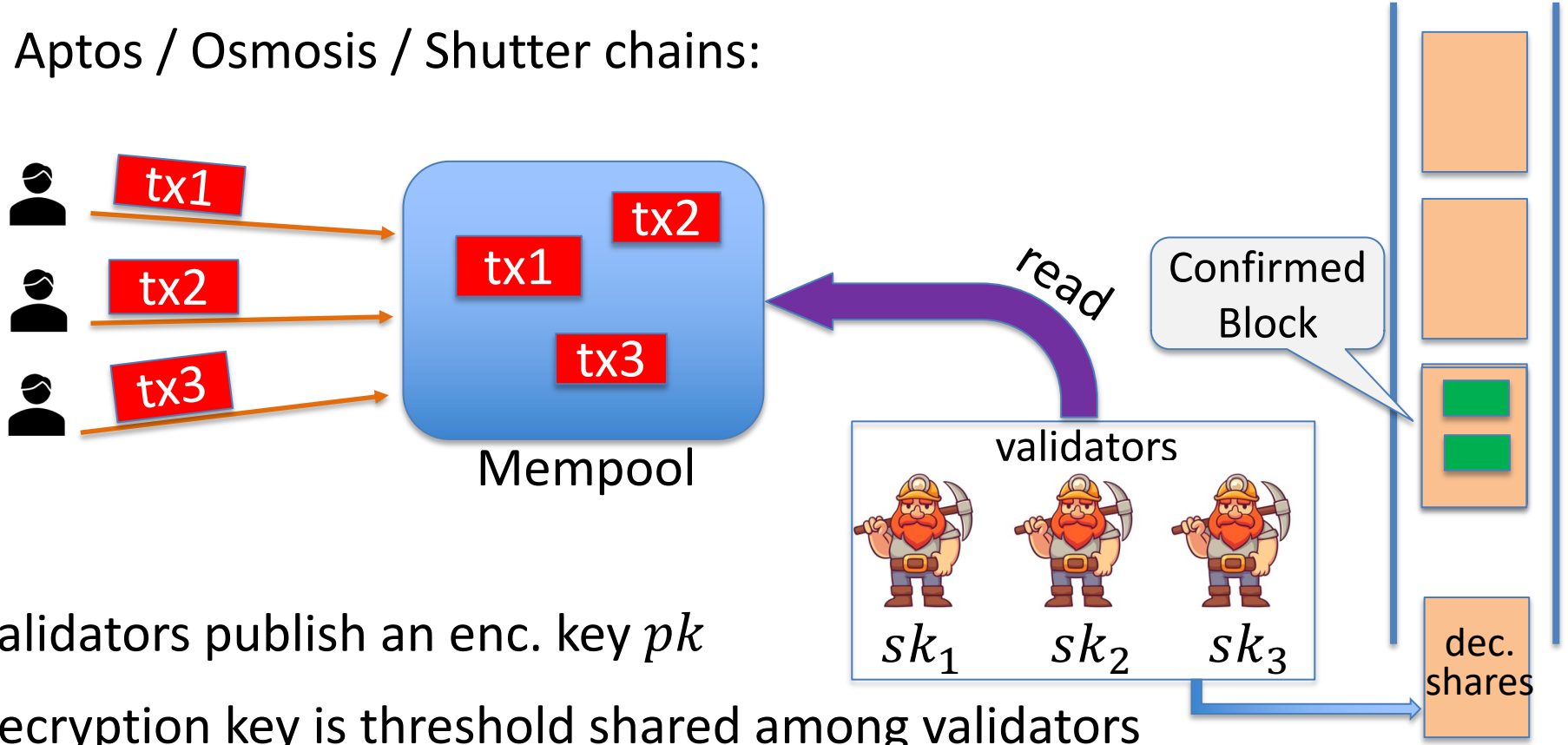
KeyGen$(n,t) \rightarrow pk,\ sk_1, sk_2, \ldots, sk_3$



Security: one sk_i "reveals nothing"

A shielded mempool via threshold decryption

Aptos / Osmosis / Shutter chains:



Validators publish an enc. key pk

decryption key is threshold shared among validators

A shielded mempool via threshold decryption

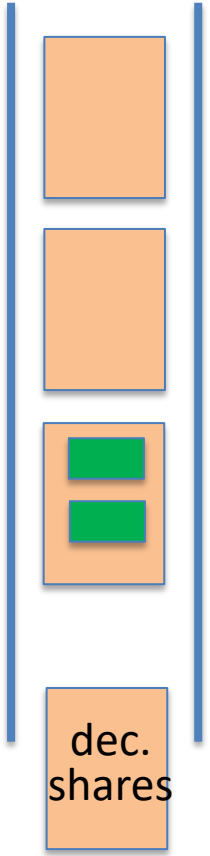
Aptos / Osmosis / Shutter chains:

- Sequencing Tx in block is done on encrypted Tx
- Block is executed only once it is decrypted

Typically:

decryption threshold == confirmation threshold

consensus liveness ⇒ enough decryption shares



Option 2: Fair ordering

A different idea: force fair ordering of transactions in block

(not currently used)

How to force fair ordering?

1. **Randomize transactions before executing**
   Downside: spamming with identical extracting transaction
2. **Time-Based Order-Fairness**
3. **Blind Order-Fairness (e.g., via encrypted mempools)**

More ideas? Your idea here …

Aequitas: Time-Based Order-Fairness

Basic idea: if most validators received tx1 before tx2, then tx1 should precede tx2 in the final ordering.

The problem of **Condorcet cycles**:

- validator #1: [tx1, tx2, tx3]
- validator #2: [tx2, tx3, tx1]
- validator #3: [tx3, tx1, tx2]

Two received (tx1 before tx2) AND two received (tx2 before tx3) AND two received (tx3 before tx1)

⇒ No ordering !!

A possible solution: reject entire cycle if Tx in cycle conflict.

[Kelkar-Zhang-Goldfeder-Juels 2020]

Aequitas: Time-Based Order-Fairness

Block-Fair-Ordering protocol:

1. Validators broadcast their order preferences.
2. Build a graph of transactions:
   a. Vertices = transactions present in a large number of orderings,
   b. Edge(tx1 ⇀ tx2) if tx1 comes before tx2 in most orderings.
3. Collapse strongly connected components to a single vertex.
4. Topologically sort vertices.
5. Finalize an ordering that respects the sort.

[Kelkar-Zhang-Goldfeder-Juels 2020]

More Time-Based Order-Fairness Protocols

Problems:

- Advantages searchers with better connectivity
- High communication: $\boldsymbol{O(n^3)}$.

Themis: same goals as Aequitas, but only $\boldsymbol{O(n^2)}$ communication.

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

More ideas needed!

An active area of research

Bonus topic: DAOs

Decentralized Organizations

Decentralized Orgs (DAO)

What is a DAO?

- A Dapp deployed on-chain at a specific address

- Anyone (globally) can send funds to DAO treasury
- Anyone can submit a proposal to DAO
  - ⟹ participants vote
  - ⟹ approved ⇾ proposal executes

snapshot.org

Examples of DAOs

- **Collector DAOs**: PleasrDAO, flamingoDAO, ConstitutionDAO, …

  (see art collection at https://gallery.so/pleasrdao)

PleasrDAO: 103 members.

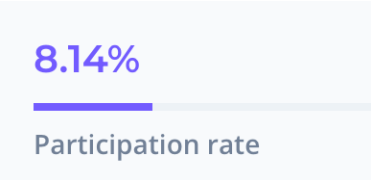
Manages a treasury, has full time employees.

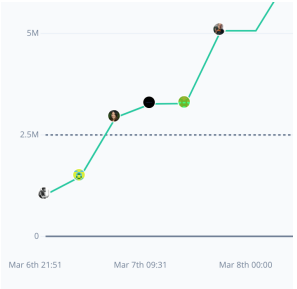
Deliberations over what to acquire over telegram.

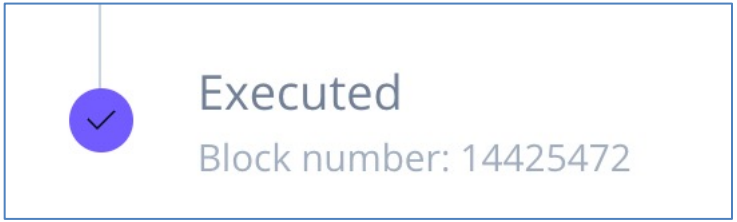
Examples of DAOs

- Collector DAOs: PleasrDAO, flamingoDAO, ConstitutionDAO, …
- **Grants DAOs**: gitcoin (83K members), …

Proposal ID 21: This proposal looks to ratify the allocation of 30,000 GTC from the Community Treasury to the MMM workstream.







(tally.com)

Examples of DAOs

- Collector DAOs: PleasrDAO, flamingoDAO, ConstitutionDAO, …
- Grants DAOs: gitcoin, …
- **Protocol DAOs**: manages operation of a specific protocol
  Uniswap DAO (74K), Compound DAO (8K), …
- **Social DAOs**: FWB, …
- **Investment DAOs**: many

Many DAO governance experiments

Who can vote? How to vote? What voting mechanism?

DAOs: a platform for experimenting with governance mechanisms

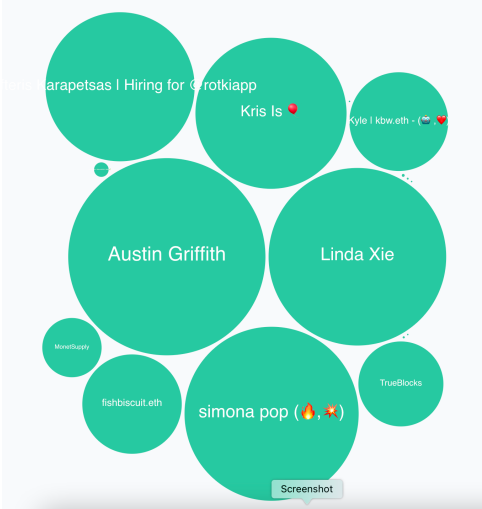
Governance method

One token one vote: (most common)

- Members receive tokens based on their contribution
- Everyone can vote

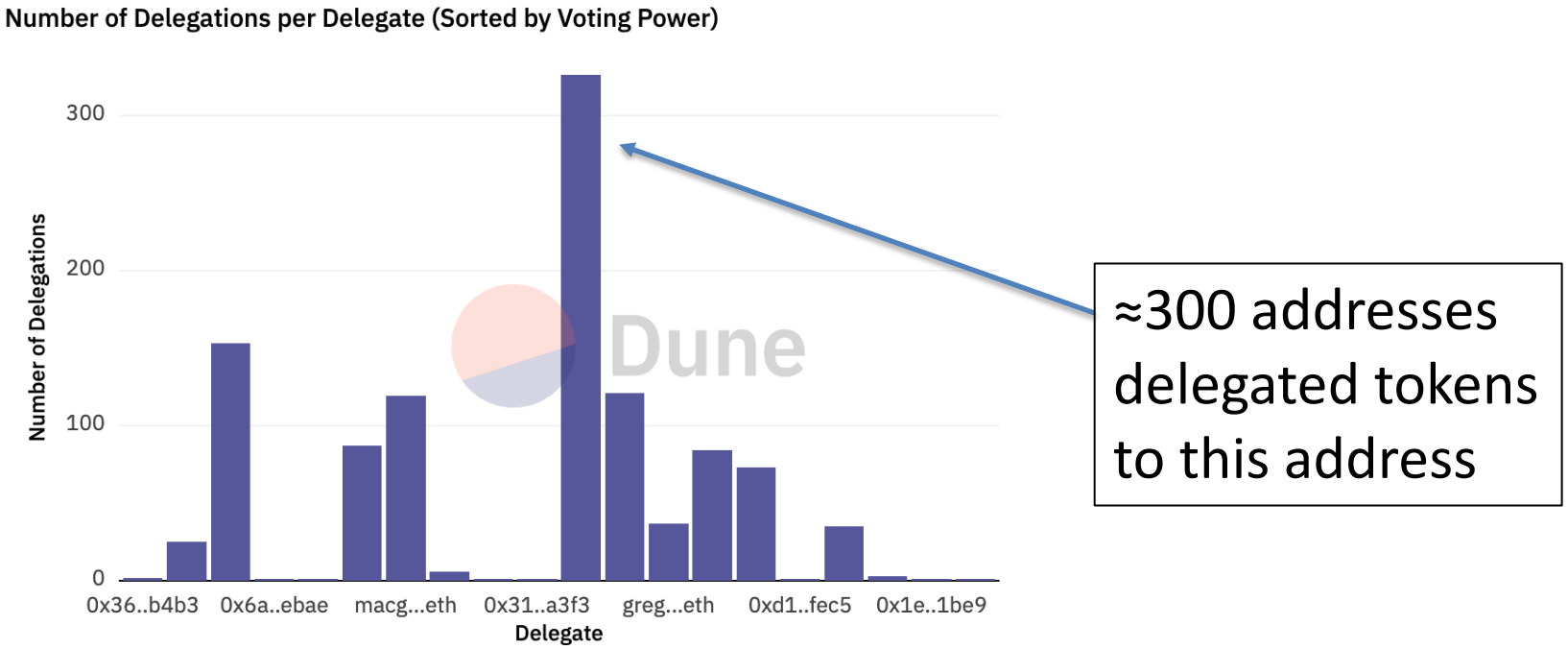
In reality: low participation rate (8%)

⇒ delegation



proposal 21 (tally.com)

Delegation example: element



Governor Bravo

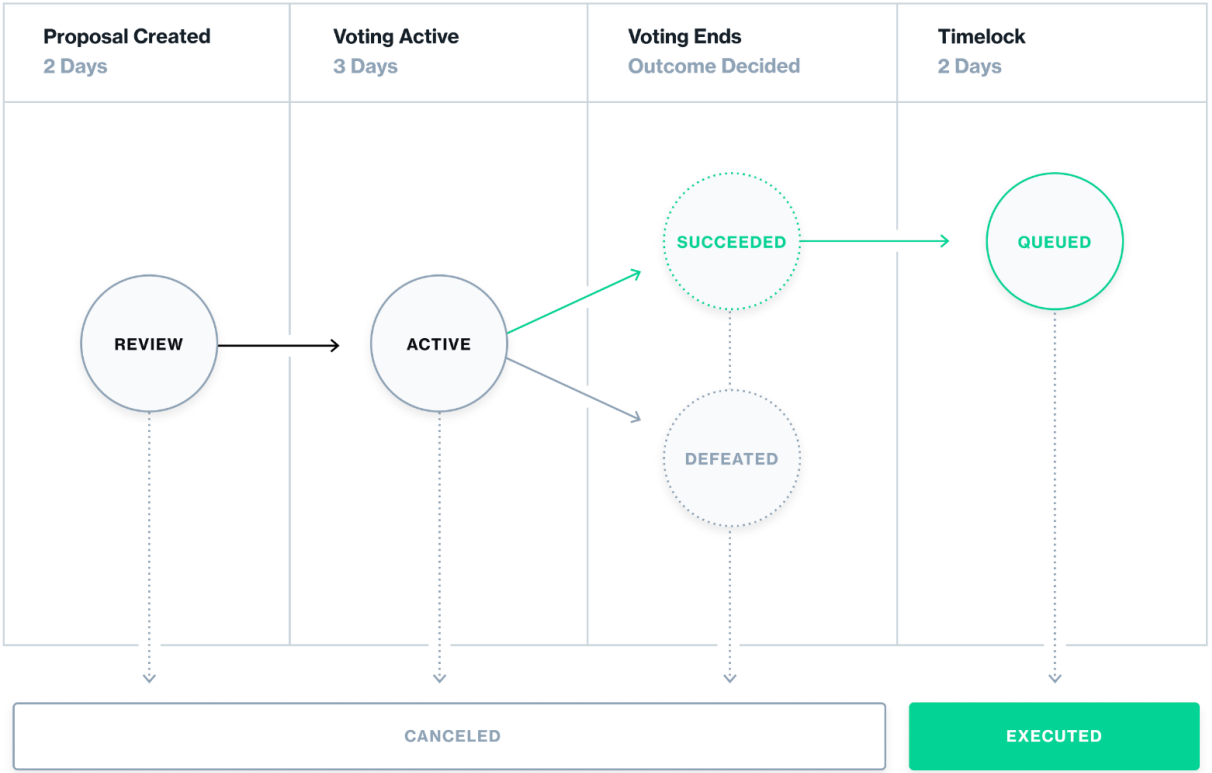
Governor alpha (Jan. 2020) updated to **Bravo** (March 2021):

- A collection of smart contracts used for governance
- Forked and used by many other protocols

Other alternatives:

- OpenZeppelin Governor protocol

The life cycle of a proposal



Governor Bravo

Implements an ERC-20 governance token. Let's call it GOV.

- An address that owns GOV tokens can vote on proposals

The proposal I am voting for
(with all my allotted voting tokens)

0: against
1: in favor
2: abstain

```
function castVote(uint proposalID, uint8 support)
```

(cannot vote partially)

Governor Bravo

Implements an ERC-20 governance token. Let's call it GOV.

… or delegate its GOV tokens to another address

function **delegate**(address delegatee)

delegate all my voting tokens

can re-delegate at any time by calling function again

An address can only delegate all its tokens to one address at a time

… but I can hold my GOV tokens at multiple addresses

Creating a proposal

```
function propose(
        address[] targets,
        uint[]    values,
        string[]  signatures,
        bytes[]   calldatas,

        string description)      //  proposal description
                                        (human readable)
    returns (uint)
```

The function to call on governor contract

If proposal passes,
how to execute it:
what functions to call (actions)
and with what arguments

The ID of newly created proposal

Global contract parameters

- **quorumVotes**: the minimum # of votes for a proposal to pass
- **proposalThreshold**: min # votes needed to create a proposal
- **votingDelay**: # blocks to wait until voting can begin after a proposal is created (e.g., two days)
- **votingPeriod**: # blocks when voting is open (e.g., three days)

All these parameters can be changed by governance in Bravo

Proposal execution

Once proposal gets enough votes and delays expire:

- anyone can call: `function execute(uint proposalID)`

⇒ prescribed functions in proposal get called

The limits of democracy

Anytime before execution

- “guardian” can call **cancel()** to cancel a proposal

Who is this guardian??

- set at Governor contract creation time
- can abdicate at any time by calling **_abdicate**()

DAO privacy questions

- **Private DAO participation**: keep membership list private
- **Private voting**: keep who voted how on each proposal private
- **Privately delegate** voting rights
- **Private treasury**

… while complying with all relevant laws.

Some DAOs made mistakes and were attacked

- Attack on Beanstalk governance: $182M
- Attack on Yam Finance governance: thwarted
- Attack on Build Finance governance $1.5M
- Attack on Mango governance: $115M
- Steem governance drama

What happened? What can we learn?

(1) Beanstalk (Apr. 2022)

An Ethereum-based stablecoin **Bean**. Governance token **Stalk.**

The problem:

- an attacker can buy Stalk tokens,
- submit a proposal,
- vote on proposal, and
- have proposal execute

all in a single transaction

Beanstalk: the attack

Recall: **flashloan** (has many applications)

- a loan that is taken out and repaid back in a single transaction
- No risk to lender ⇒ unbounded amount can be borrowed

The attack:

- Attacker took a huge flashloan from Aave, bought lots of Stalk,
- Passed a proposal to pay $80M to the attacker from the treasury,
- Sold the Stalk, and repaid the flashloan,
- Sent the proceeds to Tornado cash (donated $250K to Ukraine)

The lesson

The protocol relaunched four months later.

The lesson: governance requires delays

- Require token holders to hold token for a long period of time
- Build a delay between proposal genesis, voting, and execution

(2) Yam Finance (July 2022)

- A DeFi project running on Ethereum. Governed token **YAM**.

What happened? (within a single day)

- Attacker bought 224739 YAM with borrowed funds
- Attacker submitted a governance proposal
  granting it control of project reserves ($3M USD)
- Voted on proposal with borrowed 224739 YAM, hitting quorum
- Borrowed YAM exchanged for Eth and paid back

What happened next?

Yam Finance team: noticed the proposal shortly after it hit quorum

⇒ retained **cancel** power, and canceled the proposal.

Lesson:

- Illustrates the positive power of a veto
- The problem: power can never be taken away
  (unless voluntarily given away)

END OF LECTURE

Next lecture: The regulatory landscape