Bribers, Bribers on The Chain, Is Resisting All in Vain?

Trustless Consensus Manipulation Through Bribing Contracts

Bence Soóki-Tóth

Aarhus University

April 6, 2026

Joint work with

István András Seres

Kamilla Kara

Ábel Nagy

Balázs Pejó

Gergely Biczók

Eötvös Loránd University and

Budapest University of Technology and Economics

Traditional vs. Incentive Attacks

- Traditional Security Attacks:
  - **Goal:** Break the protocol or bypass its rules.
  - **Mechanism:** Exploiting software bugs (re-entrancy), cryptographic flaws, or network vulnerabilities (DDoS).
  - **Attacker Action:** Operates outside the intended behavior of the system.

Traditional vs. Incentive Attacks

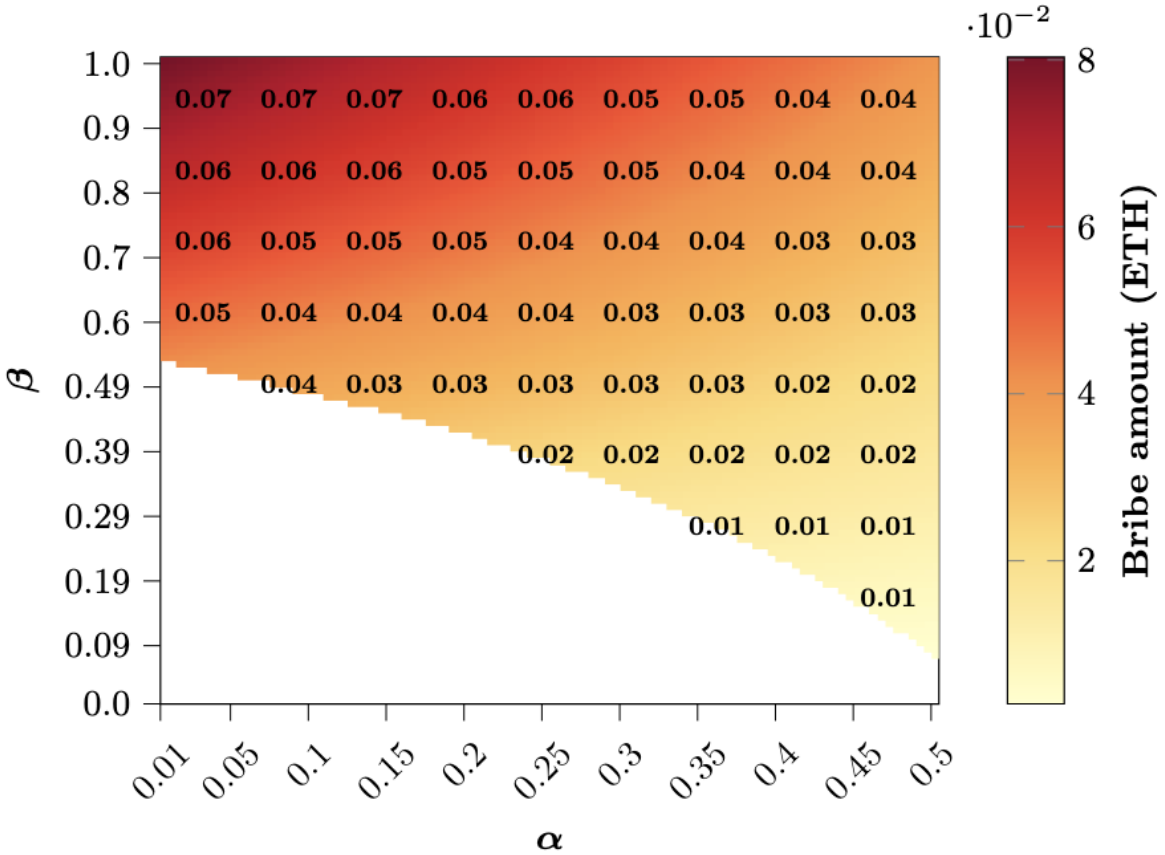
- Traditional Security Attacks:
  - **Goal:** Break the protocol or bypass its rules.
  - **Mechanism:** Exploiting software bugs (re-entrancy), cryptographic flaws, or network vulnerabilities (DDoS).
  - **Attacker Action:** Operates outside the intended behavior of the system.
- Incentive Manipulation (Bribing Attacks):
  - **Goal:** Use the protocol's rules to achieve an **adversarial outcome**.
  - **Mechanism:** Exploiting economic rationality through **trustless means** (*eg.*, smart contracts).
  - **Attacker Action:** Persuades participants to choose a specific (adversarial) path that is still **valid** within the protocol's rules but **harmful** to its goals.

Motivation: Economic security $\neq$ "value locked"

"PoS is secure because much **value is staked** and **misbehaviour is punished** by burning stake."

TL;DR

- Assuming validators are economically **rational** (profit maximizing)
- How could we **bribe** them to deviate from the protocol?
- How much should we **pay** for such deviations?
- Countermeasures (?!)

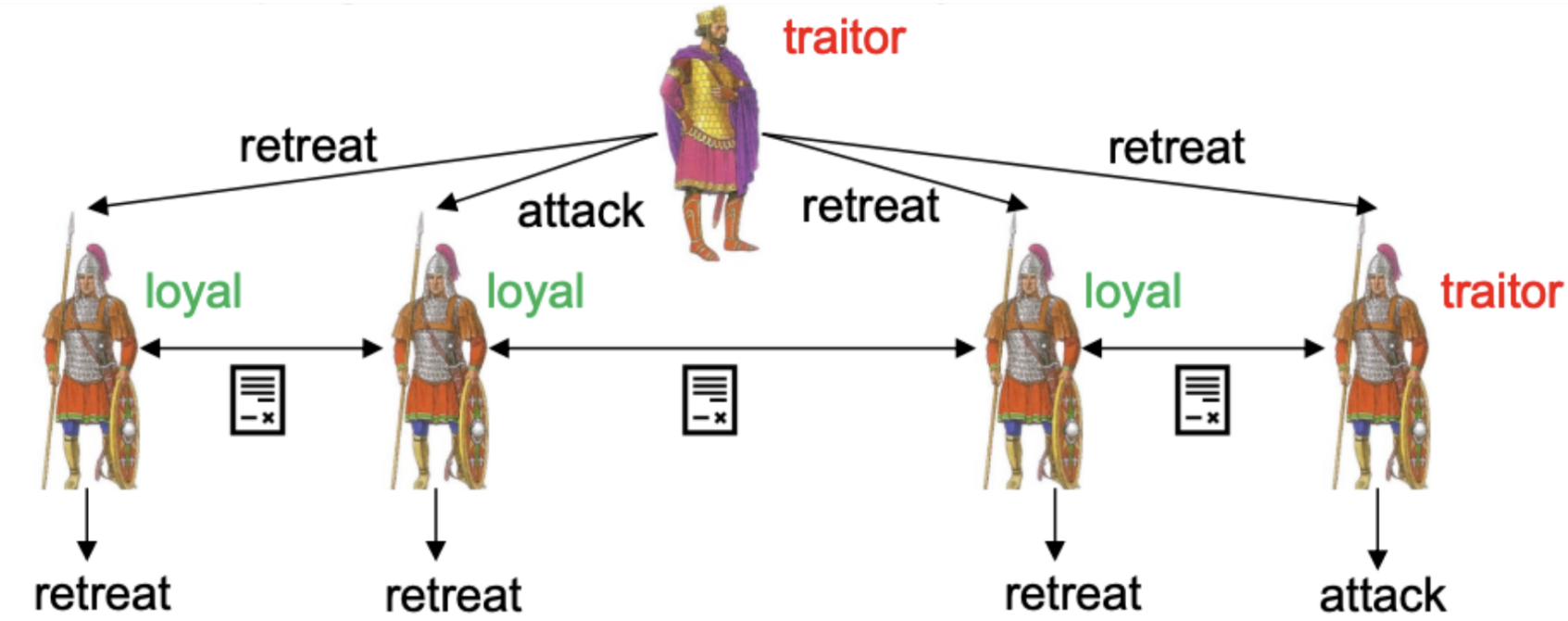


Outline

- Preliminaries on Ethereum and consensus
- System Model
- Three Bribery Attacks *(Bribers, Bribers on The Chain!)*
  - PayToAttest
  - PayToExit
  - PayToBias
- Incentive Analysis *(Is Resisting All in Vain?)*
  - Incentive analysis of the proposed bribery markets
- Conclusion, Mitigations, Open problems

Preliminaries

Consensus 101: Byzantine generals problem



Consensus 101: Goals of consensus

- **Agreement:** no two loyal generals take different actions.
- **Validity:** if the commander is loyal, then all loyal generals take the same action suggested by the commander.
- **Termination:** all loyal generals must eventually take some action.

Who can participate?

- **Permissioned:** There are fixed set of nodes. (classical consensus)
- **Permissionless:** anyone is free to join the protocol any time.

Who can participate?

- **Permissioned:** There are fixed set of nodes. (classical consensus)
- **Permissionless:** anyone is free to join the protocol any time.

Sybil attacks!

solution: proof-of-stake, proof-of-work, proof-of-space, proof-of...

Real World Consensus (RWC)

- Byzantine generals problem [Lamport, 1982]
  - Fixed number of participants
- Bitcoin Proof-of-work (PoW) [Nakamoto, 2008]
  - Sybil-resistance
  - Dynamic availability: join or leave permissionlessly
  - PoW: H(transactions,nonce) $< 2^{\frac{256}{D}}$, where D is the difficulty
- Ethereum: Proof-of-stake [Buterin et al., 2022]
  - Validators (BLS-)sign block headers
  - Weighted by their stake

BLS signatures from pairings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ [BLS01'ASIACRYPT]

Key generation

$$\mathsf{KeyGen}(1^\lambda) : sk \xleftarrow{\$} \mathbb{F}_p;\ pk = g_1^{sk} \in \mathbb{G}_1$$

Signing

$$\mathsf{Sign}(sk, m) : \sigma := H(m)^{sk} \in \mathbb{G}_2,\ \text{where } H : \{0,1\}^* \to \mathbb{G}_2$$

Verification

$$\mathsf{Verify}(pk, m, \sigma) = 1 \iff e(g_1, \sigma) = e(pk, H(m))$$

Hash-to-curve H is modeled as a random oracle.

https://www.iacr.org/archive/asiacrypt2001/22480516.pdf

BLS Aggregate Signatures [BGLS03'EUROCRYPT]

Same Message Aggregation ($\mathcal{O}(1)$)

$$e\left(\prod_{i=1}^{n}\sigma_i, g_2\right) = e\left(\prod_{i=1}^{n}\mathrm{pk}_i, H(m)\right)$$

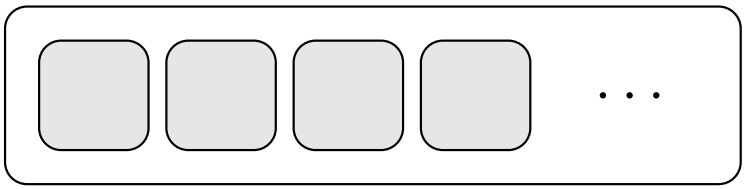
Different Message Aggregation ($\mathcal{O}(n)$)

$$e\left(\prod_{i=1}^{n}\sigma_i, g_2\right) = \prod_{i=1}^{n} e(\mathrm{pk}_i, H(m_i))$$

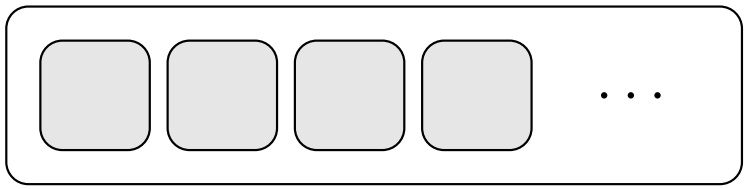
Aggregate verification via bilinear pairings

https://eprint.iacr.org/2002/175

Epochs and Slots



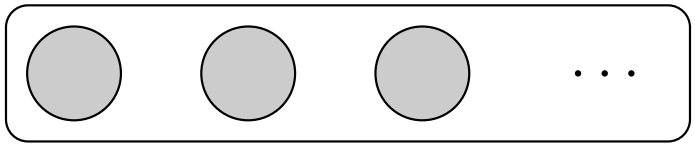
Epoch i



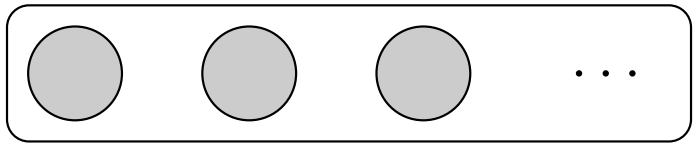
Epoch $i + 1$

- Time is divided into **slots** (12 seconds).
- A single validator proposes a block in each slot.
- **32 slots form an epoch**.
- Validator duties are scheduled per epoch.

Validator Committees



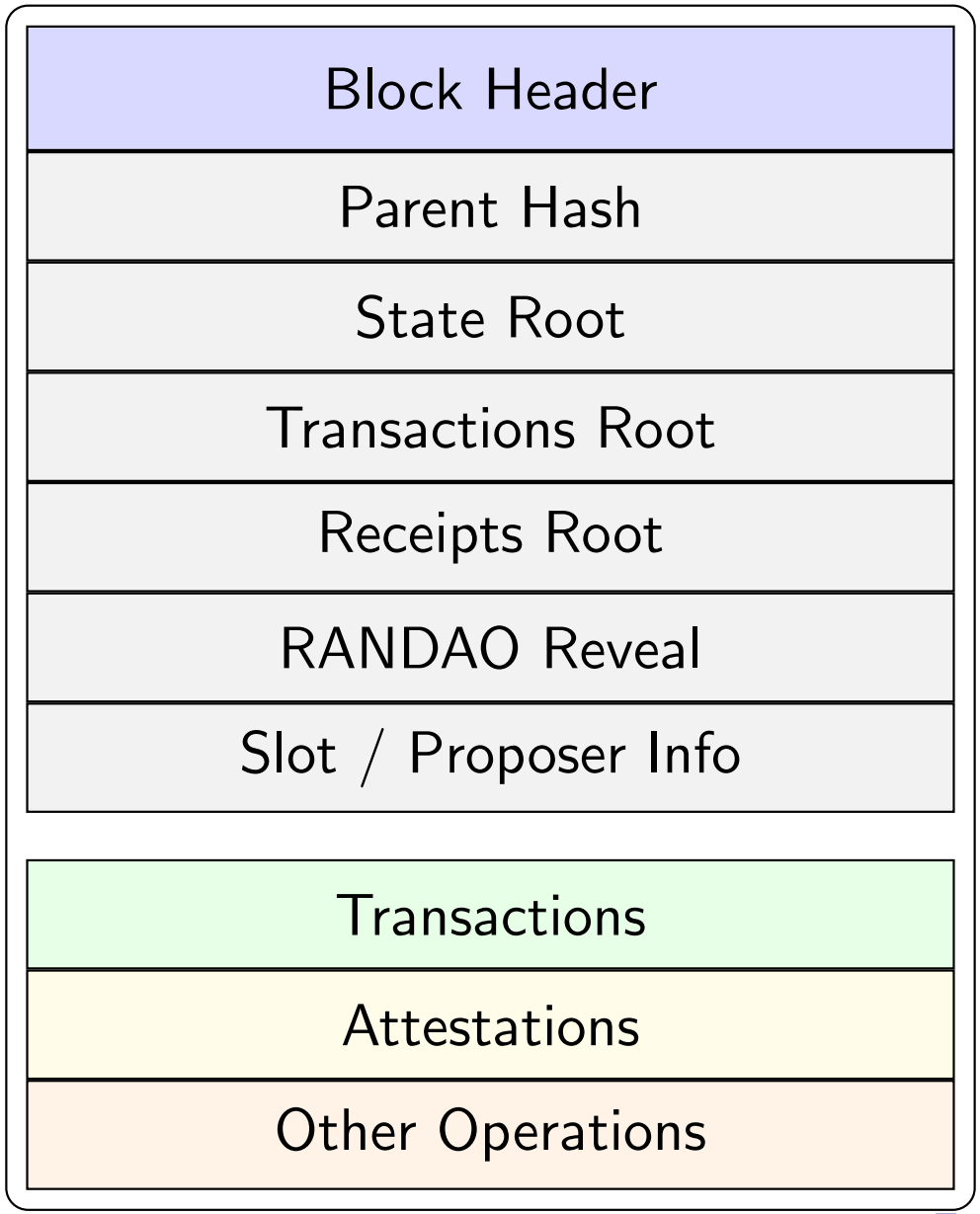
Committee A



Committee B

- Validators are randomly split into **committees**.
- Each committee is assigned to a **specific slot**.
- Committee members verify blocks and submit votes.
- This provides scalability and distributes verification work.

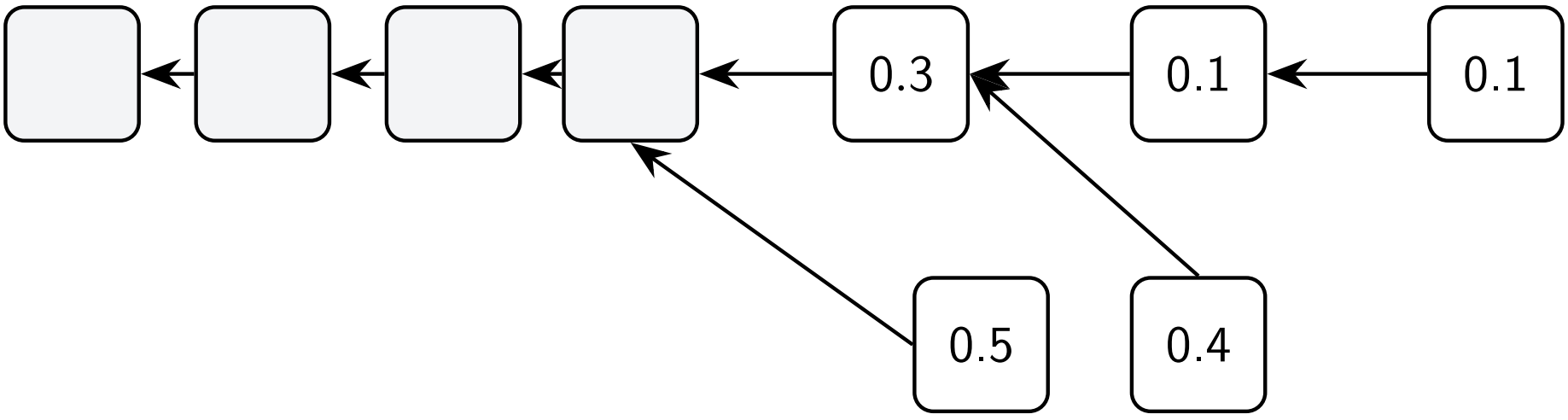
The Anatomy of a Block



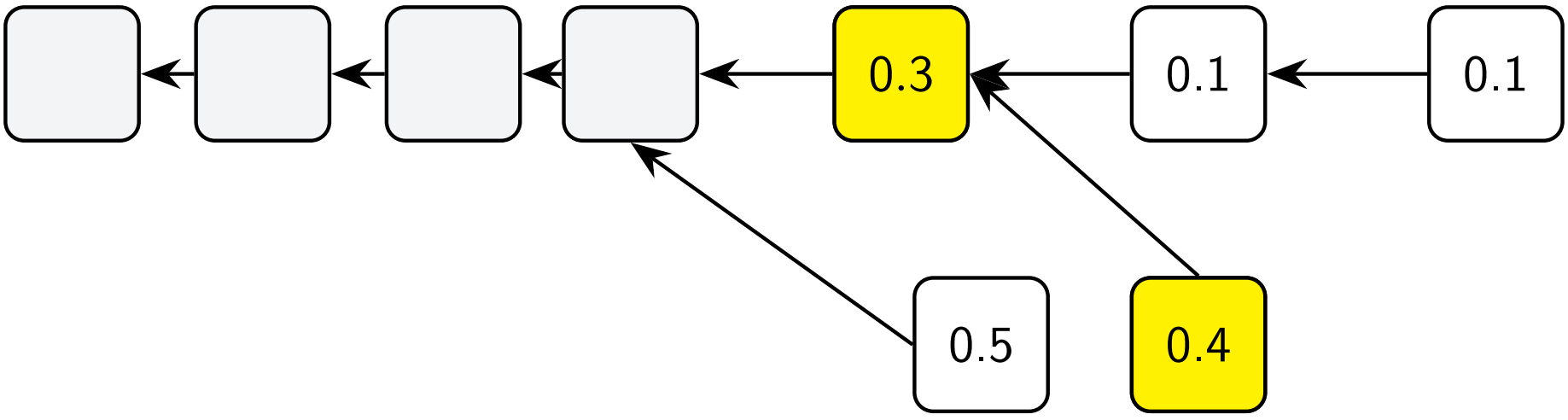
Ethereum Proof-of-Stake (PoS)

- **Permissionless protocol**
  - Anyone can *join or leave* the protocol at any time
- **Validators have two main responsibilities**
  - **Attest (vote)** on blocks
  - **Propose blocks**
- **Misbehavior is penalized (slashing)**
  - Voting for **conflicting** blocks
  - Proposing **multiple** blocks at the **same height**
- **Fork choice rule**
  - The block tree with the **most validator votes** becomes the **canonical** chain

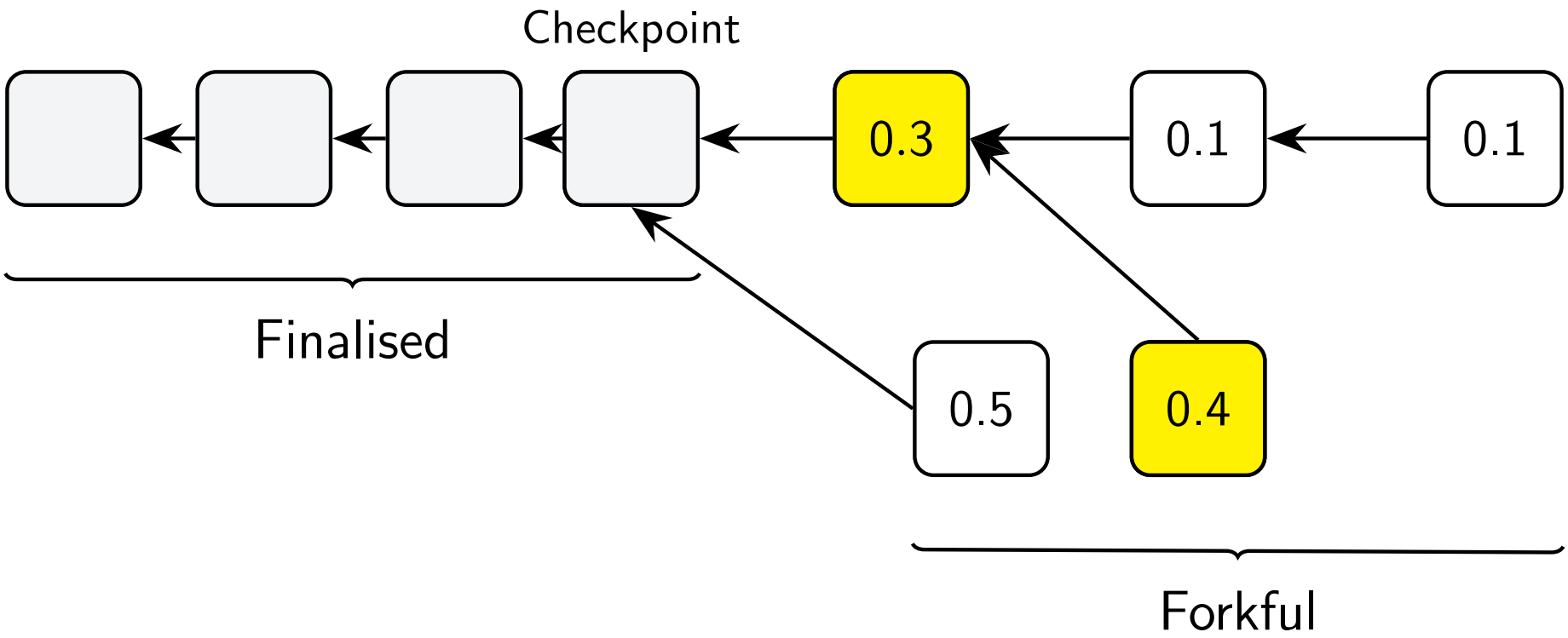
Fork choice rule and finality [Buterin et al., 2020,SA15'FC]



Fork choice rule and finality [Buterin et al., 2020,SA15'FC]



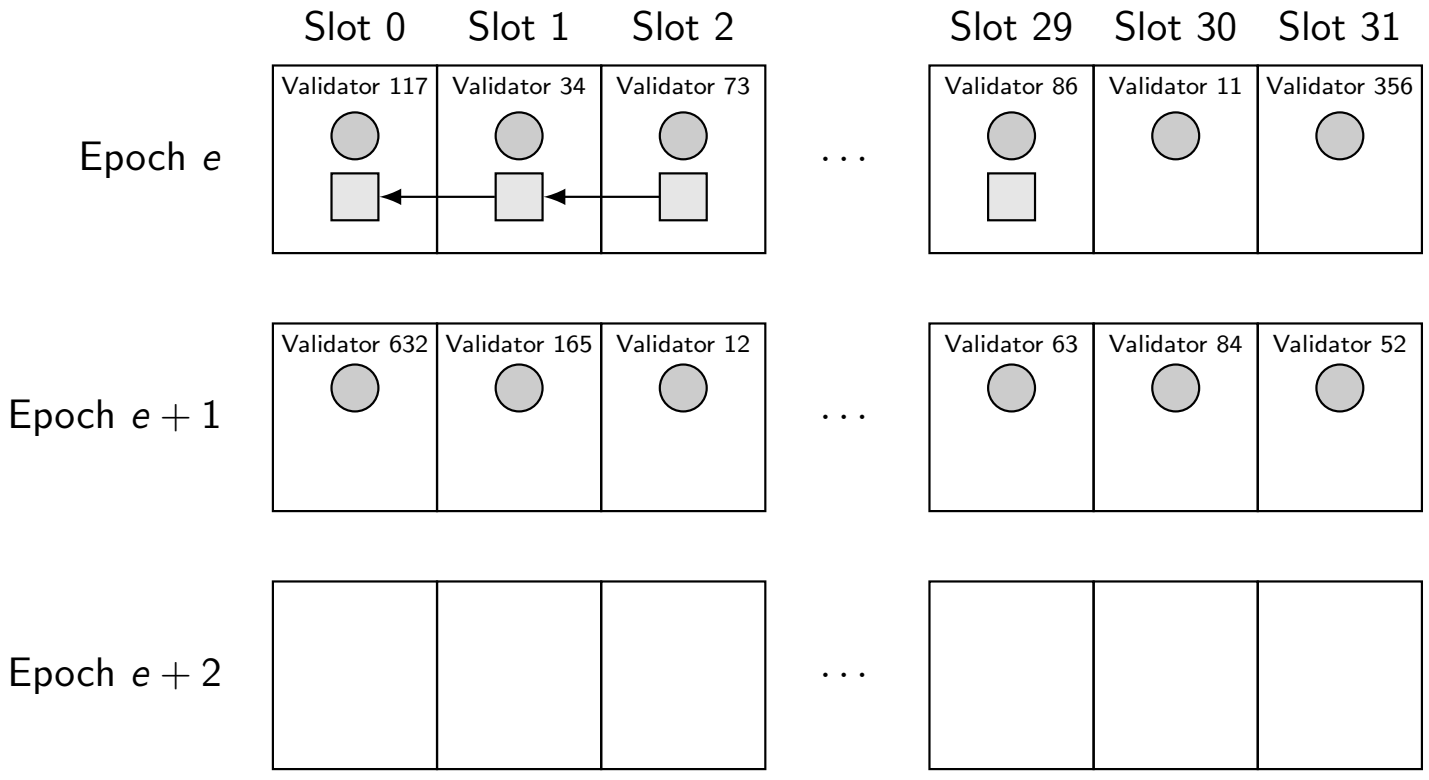
Fork choice rule and finality [Buterin et al., 2020,SA15'FC]



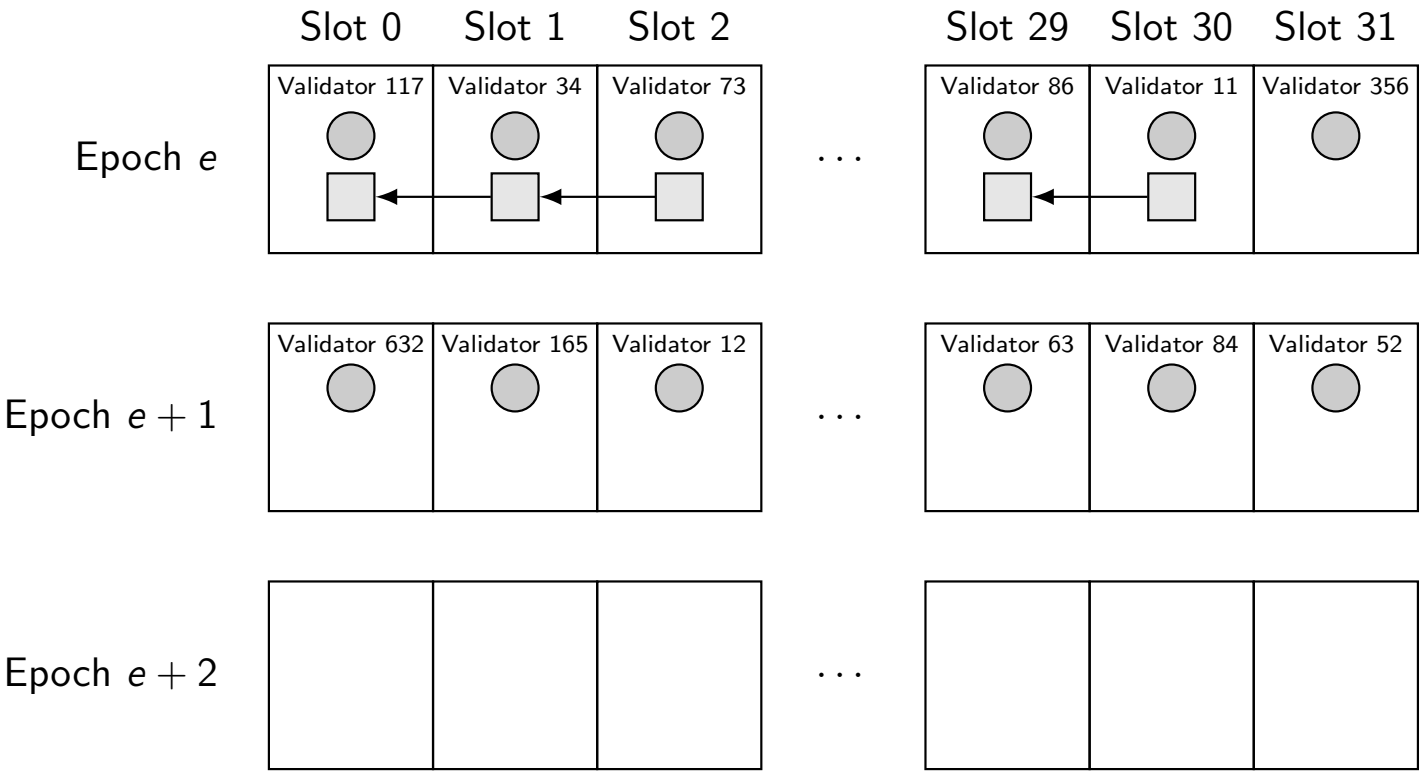
Goals of Ethereum PoS Consensus

- **Safety:** The protocol does not produce conflicting states.
- **Liveness:** The protocol consistently produces new blocks and finalizes transactions
- **Fairness:** A validator with $x\%$ stake earns $x\%$ of block proposals and rewards (in expectation)

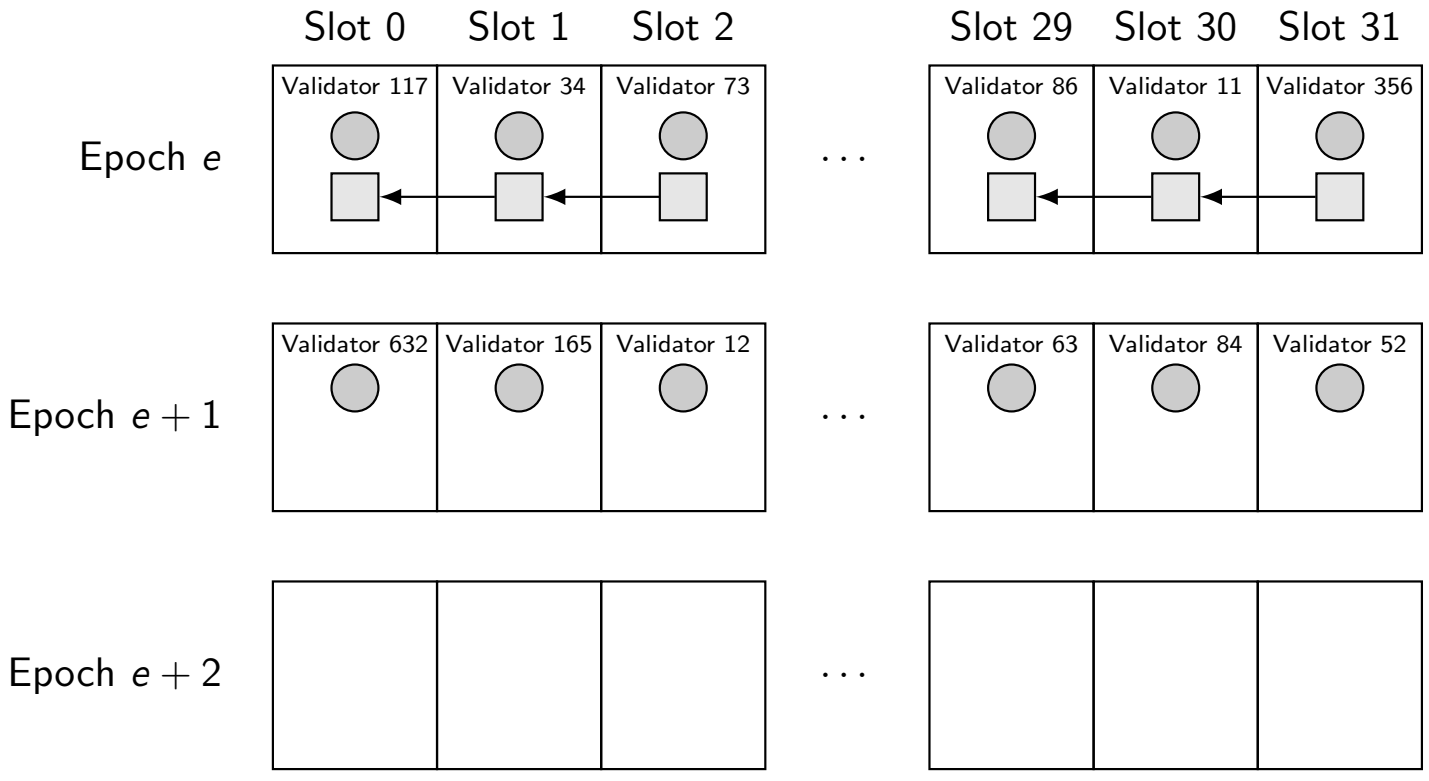
Full Epoch Process



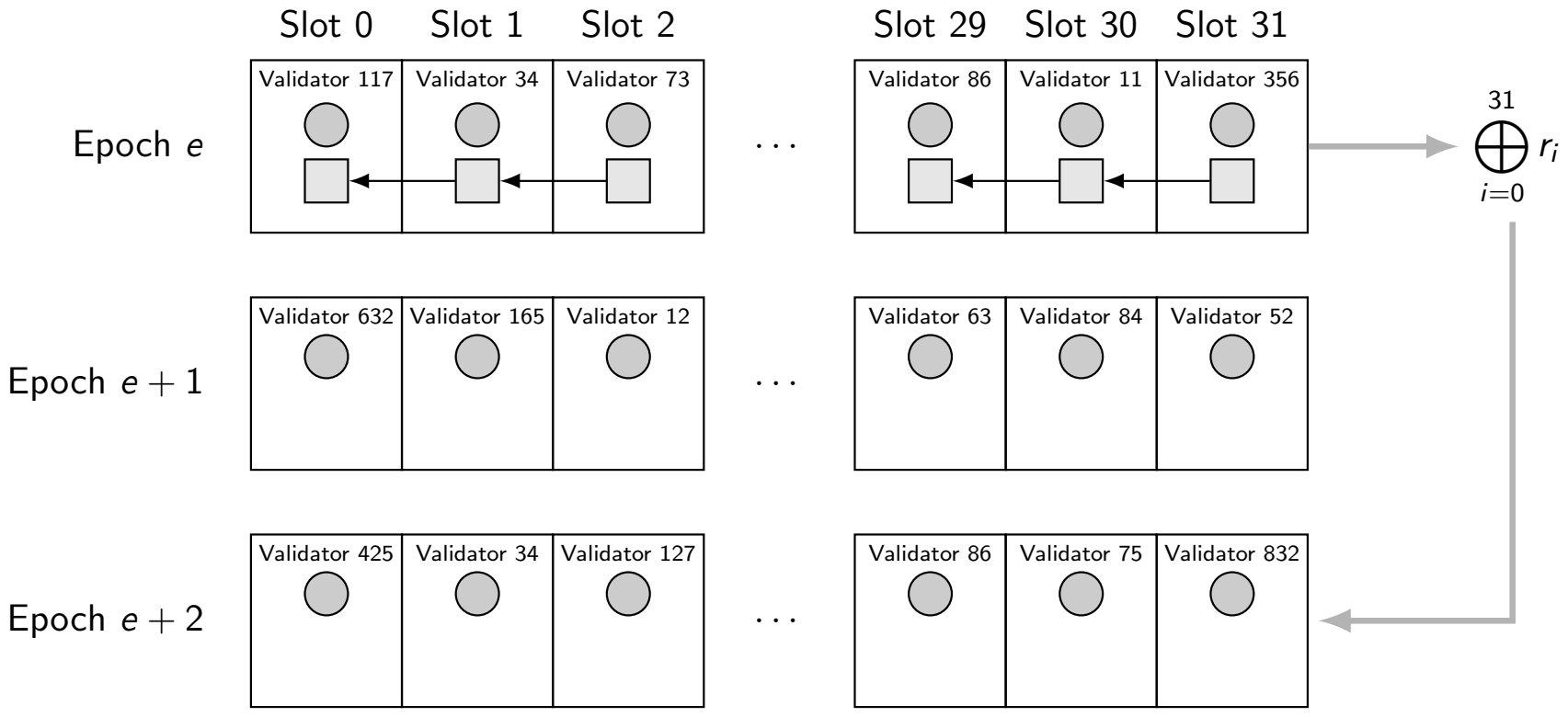
Full Epoch Process



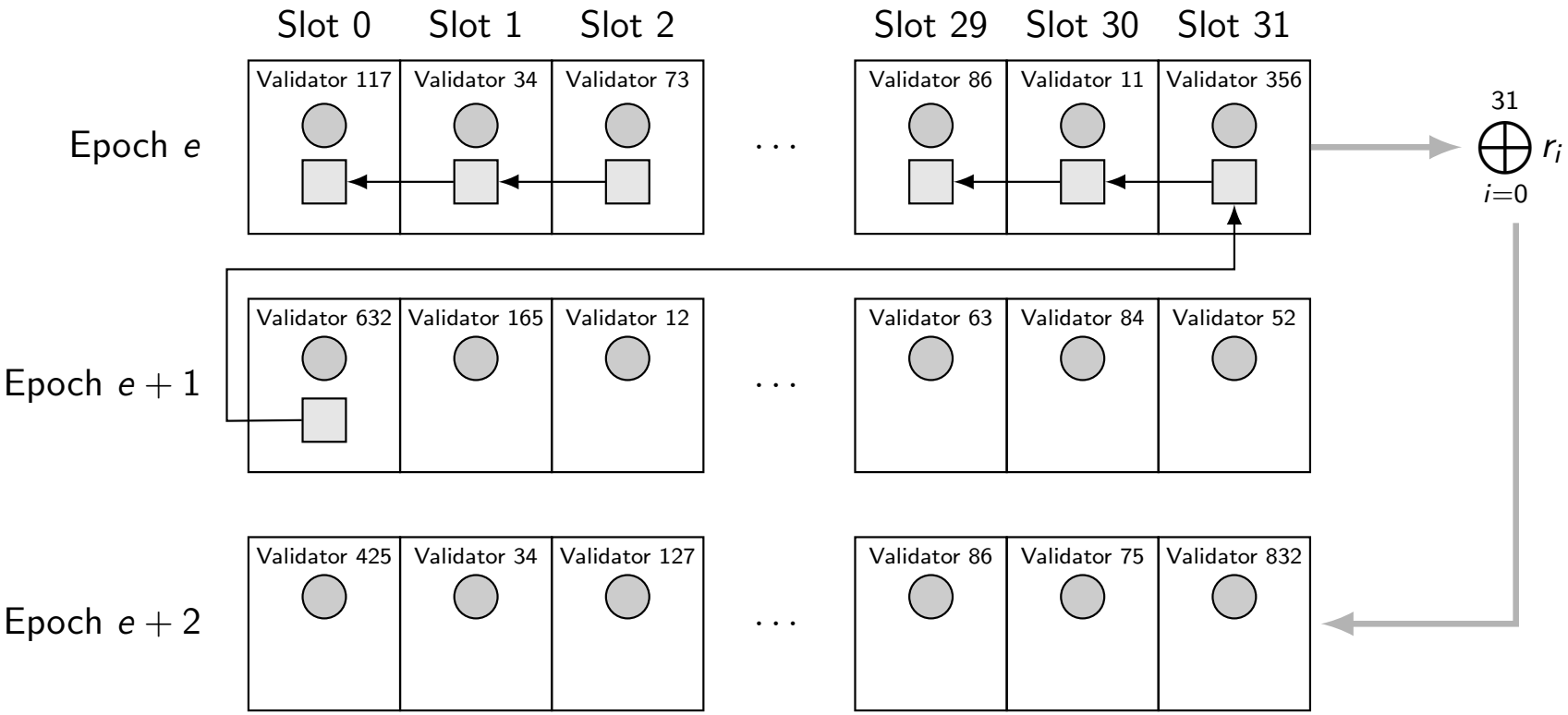
Full Epoch Process



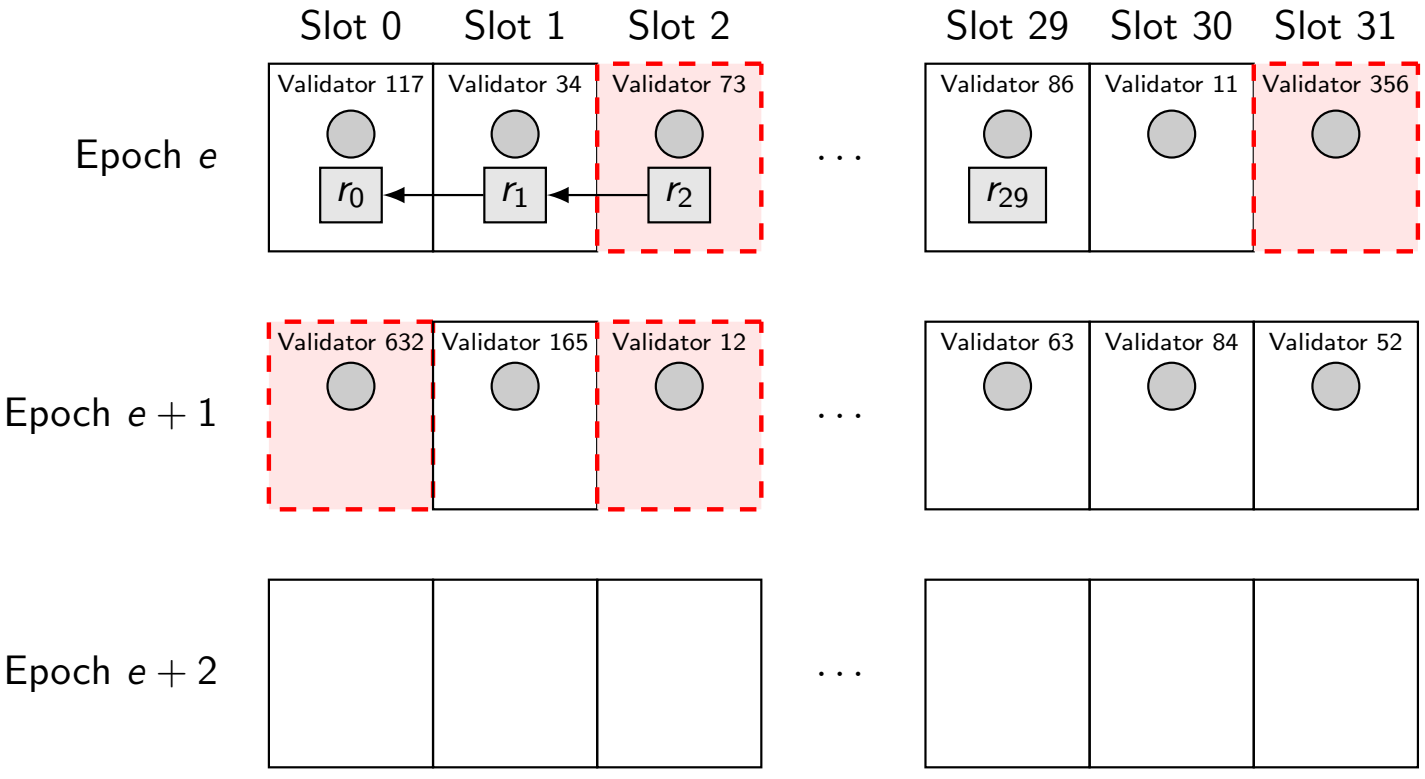
Full Epoch Process



Full Epoch Process

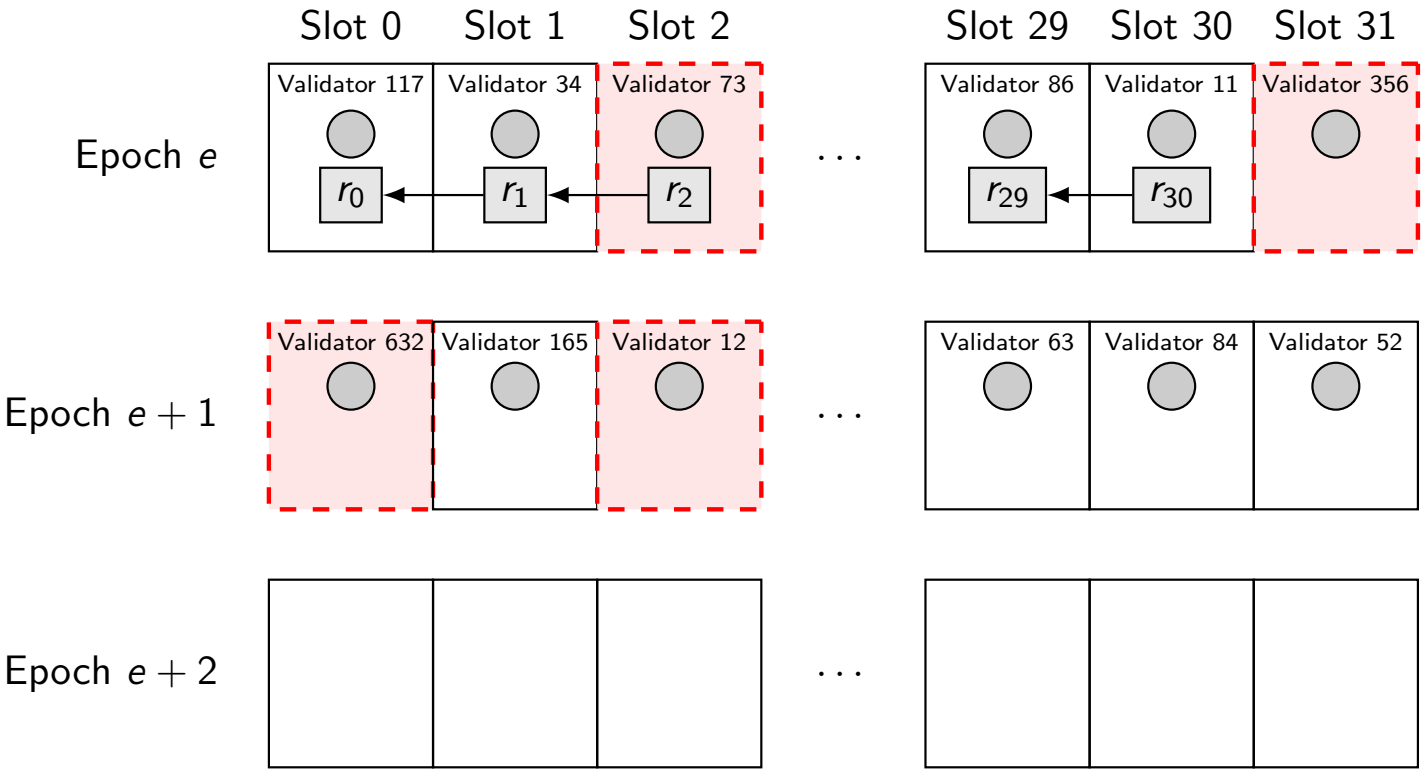


Selfish Mixing [NTSL25'CCS]



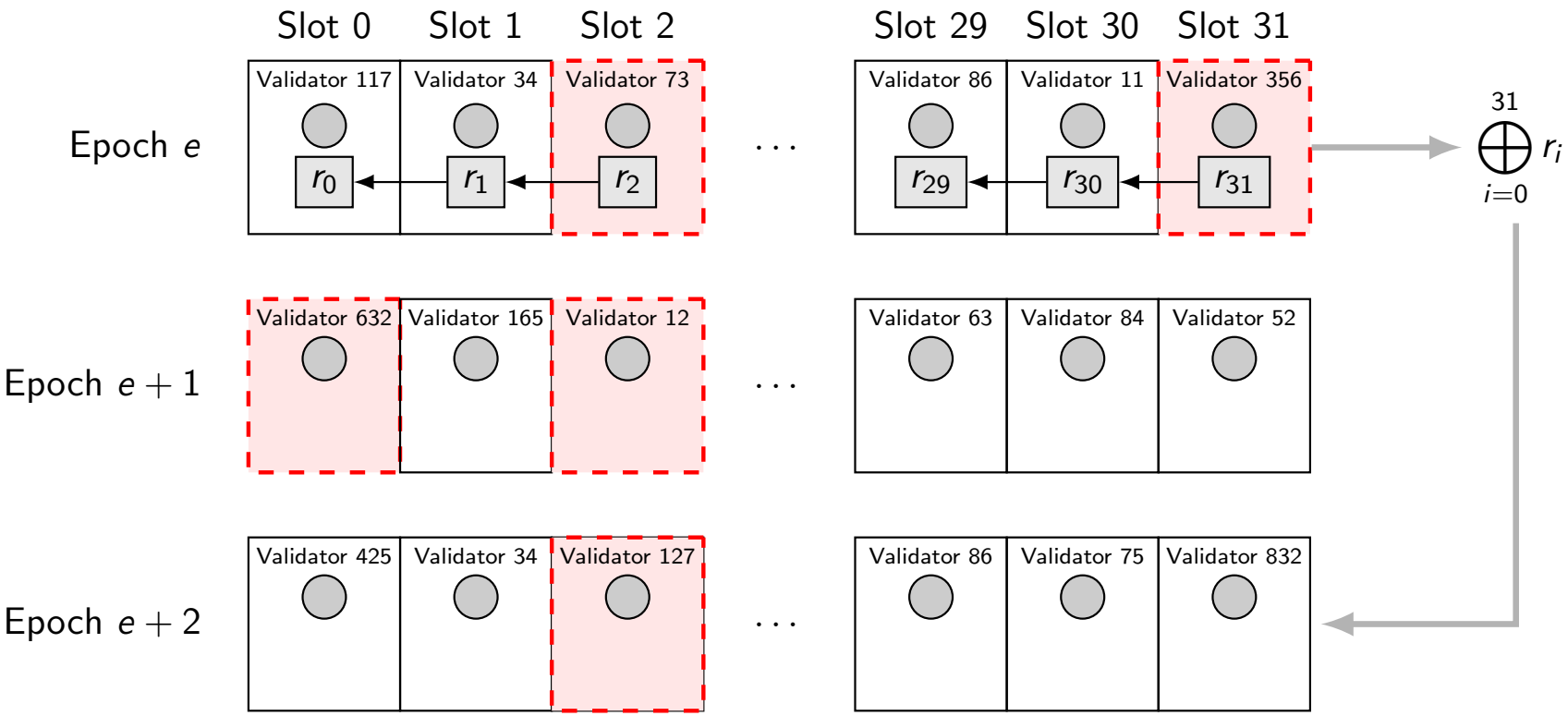
https://eprint.iacr.org/2025/037.pdf

Selfish Mixing [NTSL25'CCS]



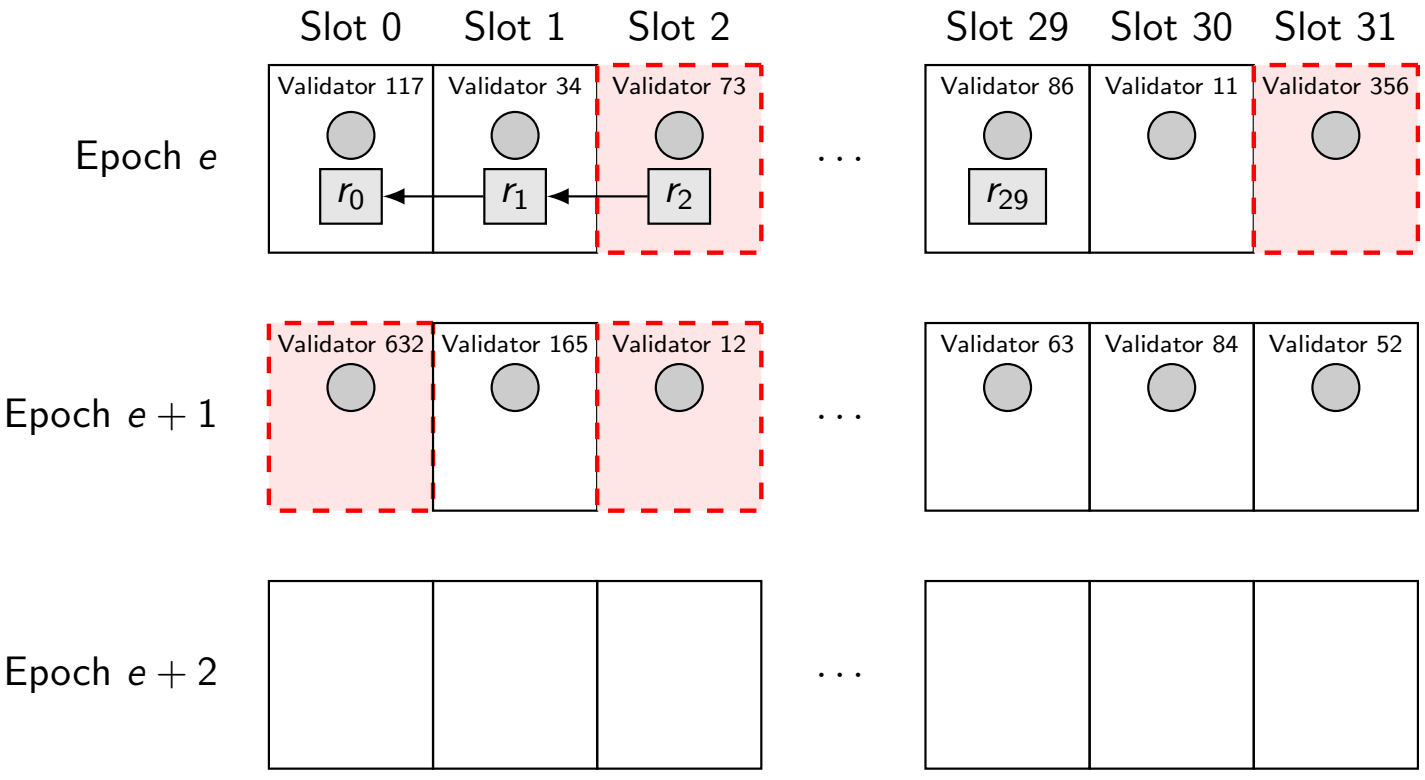
https://eprint.iacr.org/2025/037.pdf

Selfish Mixing [NTSL25'CCS]

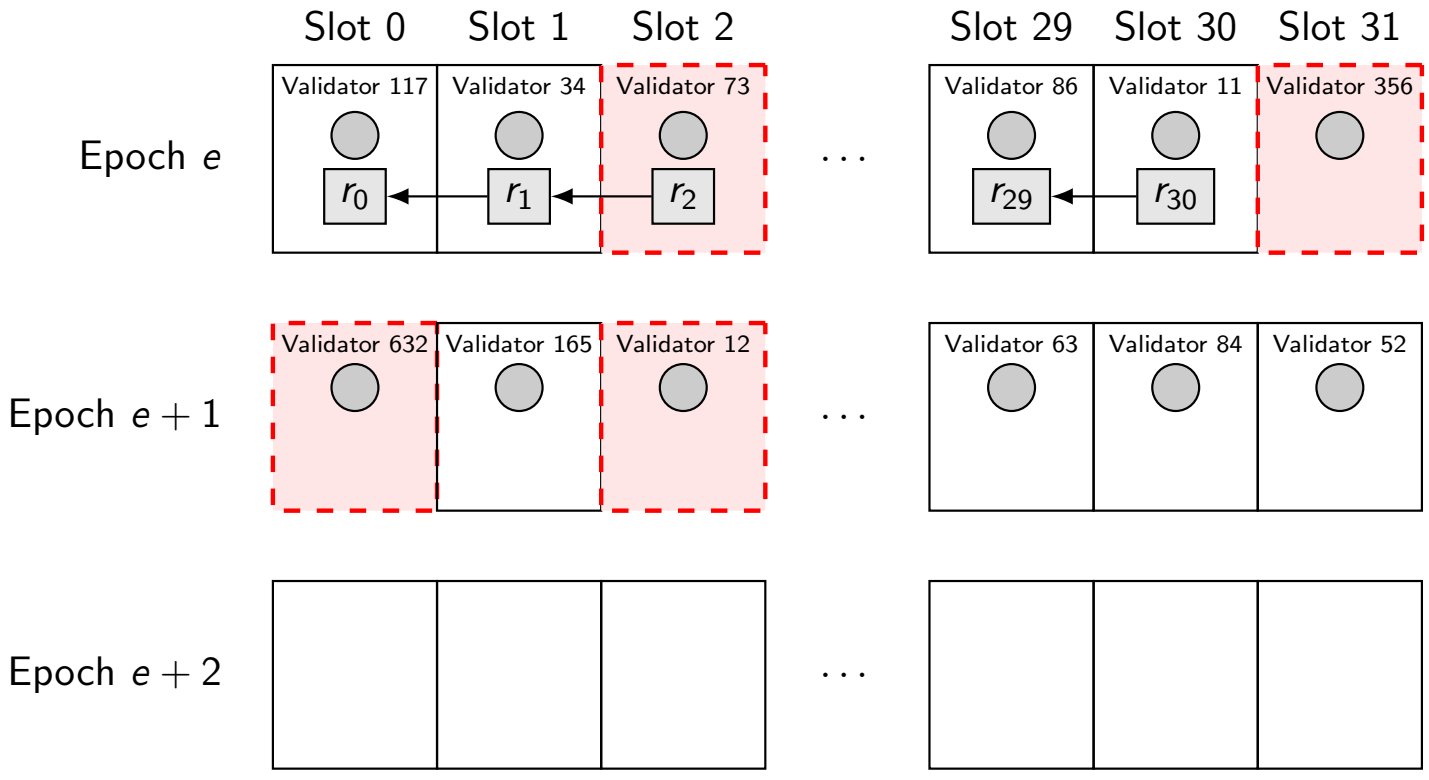


https://eprint.iacr.org/2025/037.pdf

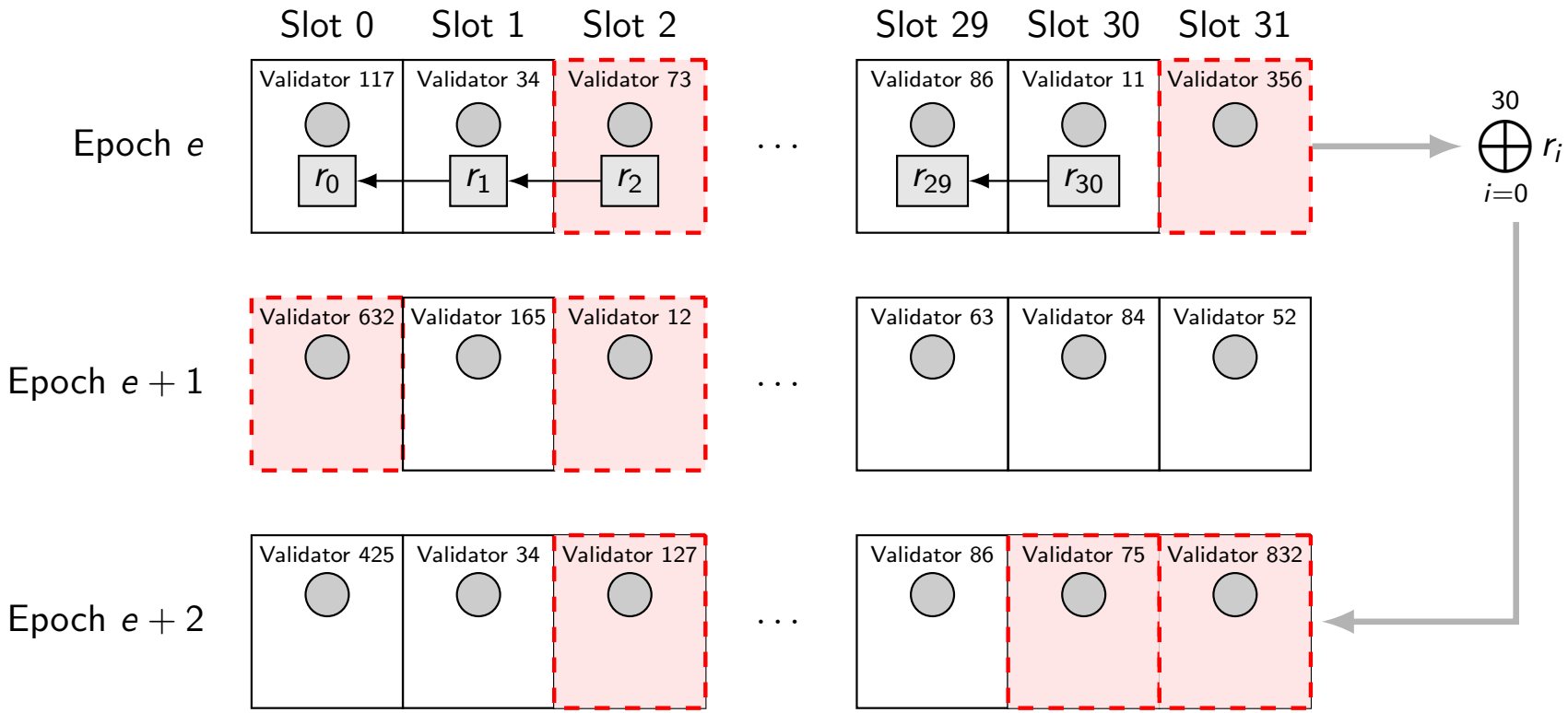
Maximizing proposer opportunities



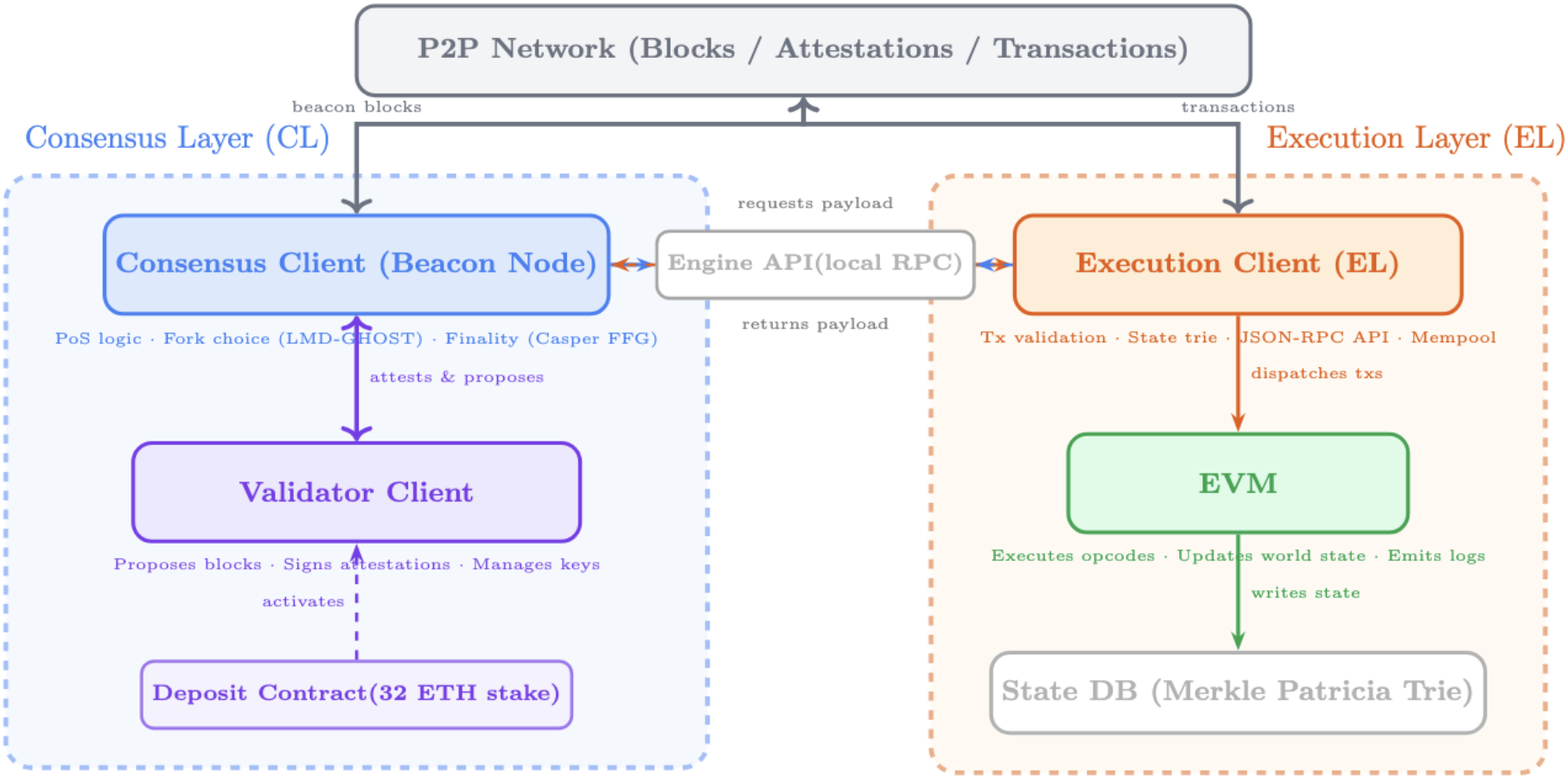
Maximizing proposer opportunities



Maximizing proposer opportunities



Ethereum Validator Architecture



Latest Ethereum Updates

- **EIP-2537:** Adds efficient `BLS12-381` group and pairing operations
- **EIP-7002:** Validators can exit the validator set via the execution layer
- **EIP-4788:** Exposes the Beacon Chain block root to the EVM
- **EIP-7549:** Uses identical attestation data for all validators

BLS aggregate verification gas costs

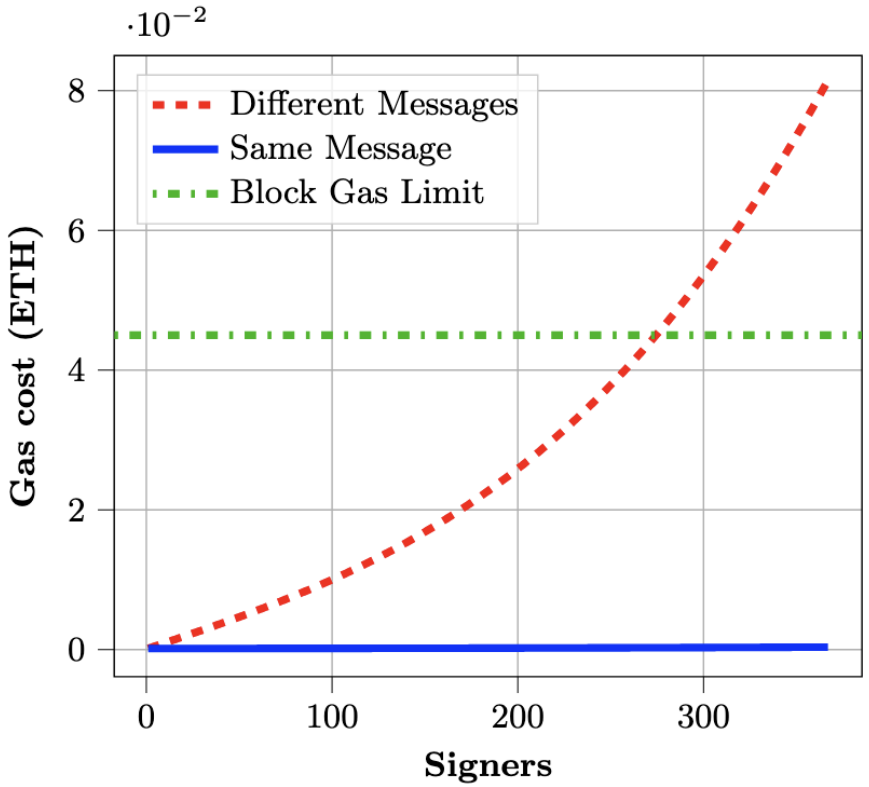


Figure: BLS batch verification gas costs.

System Model

BAR model for validators [Aiyer et al., SOSP'05]

Partition stake (or validators) into:

- **Byzantine / Bribers** (α): try to induce protocol deviation.
- **Altruistic** $(1-\alpha)(1-\beta)$: follow protocol regardless of payoffs.
- **Rational** $(1-\alpha)\beta$: deviates if compensated enough.

Key assumption:

$$\text{Bribe} > \text{Honest (protocol) reward} + \mathbb{E}[\text{penalty / slashing}].$$

$$0 \leq \alpha \leq 0.5; \quad 0 \leq \beta \leq 1$$

System Model

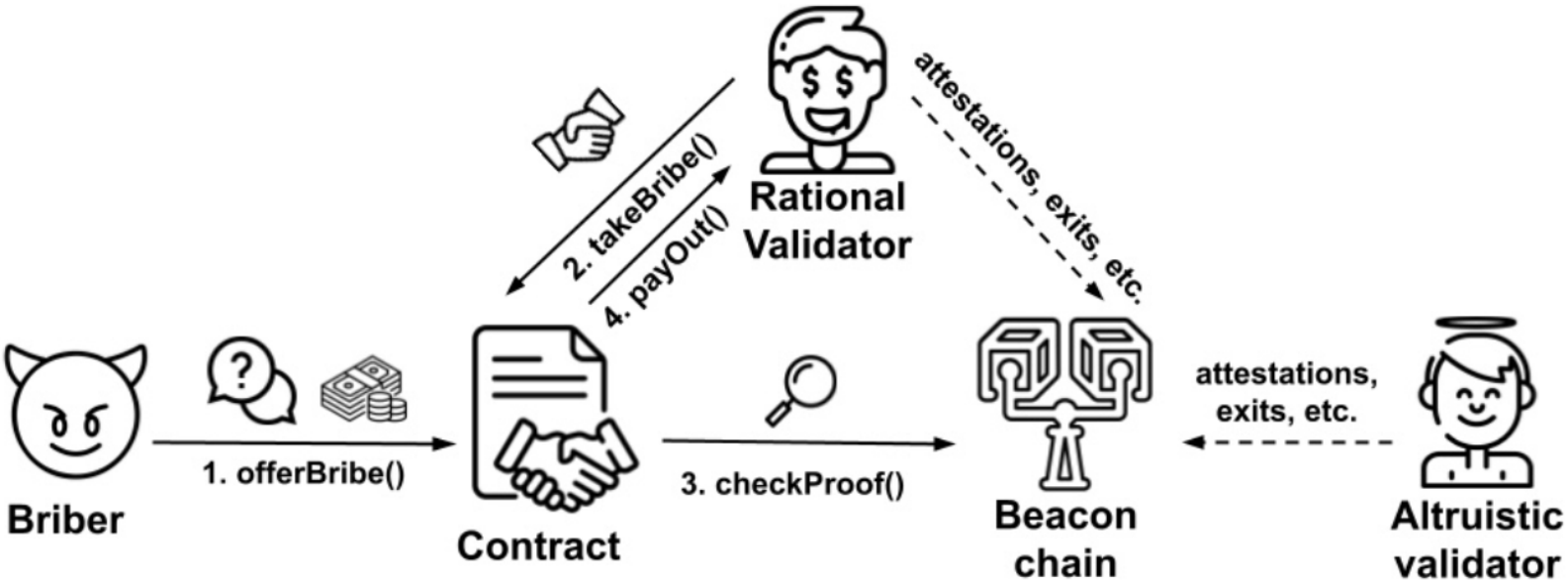
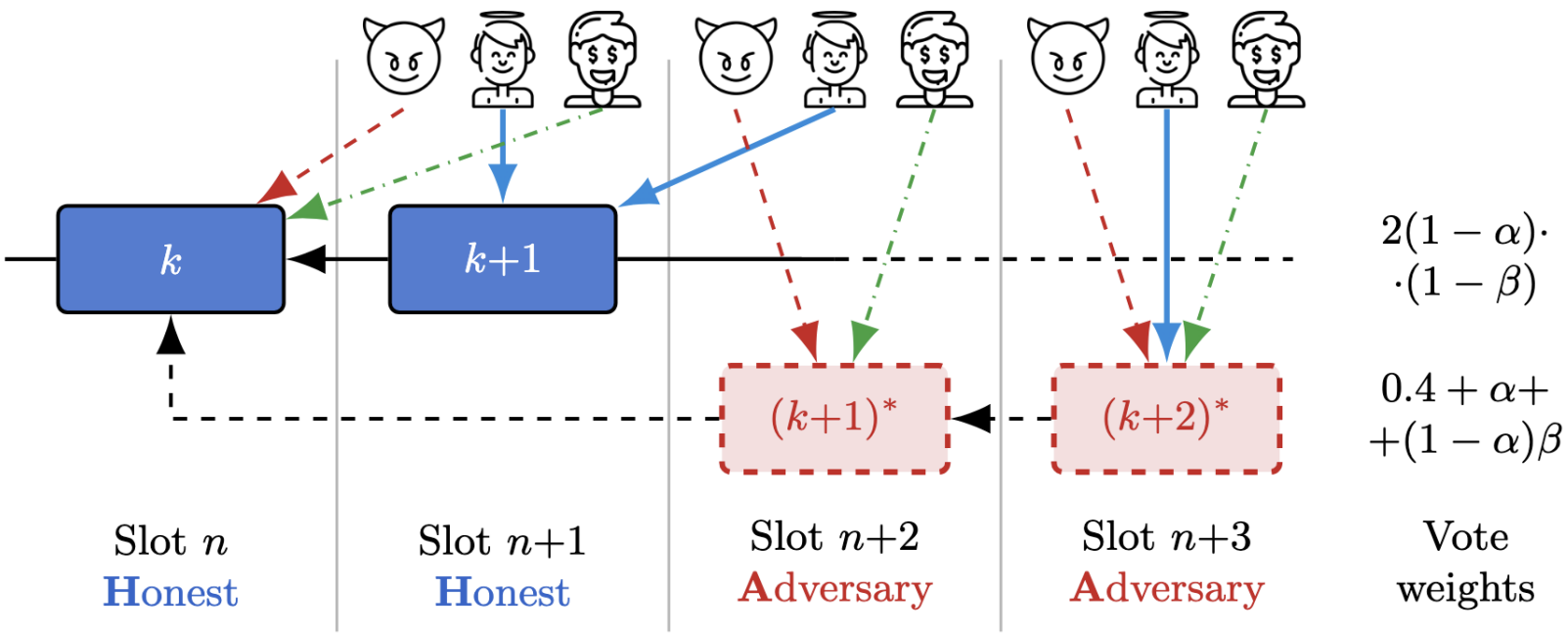


Figure: Our simple bribery system model.

Three Bribery Attacks

Bribers, Bribers on The Chain!

Ex-post reorg [ES14'FC, Daian et al.19'IEEESP]



Contract 1: PayToAttest (forking / reorg)

Goal: violate safety!

Allows and adversary to **buy attestations** for a particular block header m.

Protocol:

- briber publishes target aggregate public key:

$$pk = \prod_{i \in S} pk_i$$

- briber publishes message m (block header / attestation payload)
- contract pays if presented with valid aggregate BLS signature σ on m

Contract 2: PayToExit (validator-set manipulation)

Goal violate liveness!

Pay validators to voluntarilty **exit the active validator set**, increasing the attacker's **relative influence**.

Protocol:

- Briber funds a contract that pays anyone who provides proof they initiated a validator exit.
- Proof can be based on an execution-layer verifiable authorization (e.g., an ECDSA signature, Merkle membership proof for the deposit contract's Merkle root).

Contract 3: PayToBias (RANDAO manipulation market)

Goal violate fairness!

Allows an adversary to **bias** the output of Ethereum's distributed **randomness** beacon (aka RANDAO).

Protocol:

- A validator controlling k tail slots, can pre-compute and publish their k randomness contributions.
- Other validators bid on their preferred outcome by submitting a publish/withhold configuration ($c_i \in \{0,1\}^k$)
- The manipulator is then incentivized to execute configuration c_i that received the highest bid.

Incentive Analysis

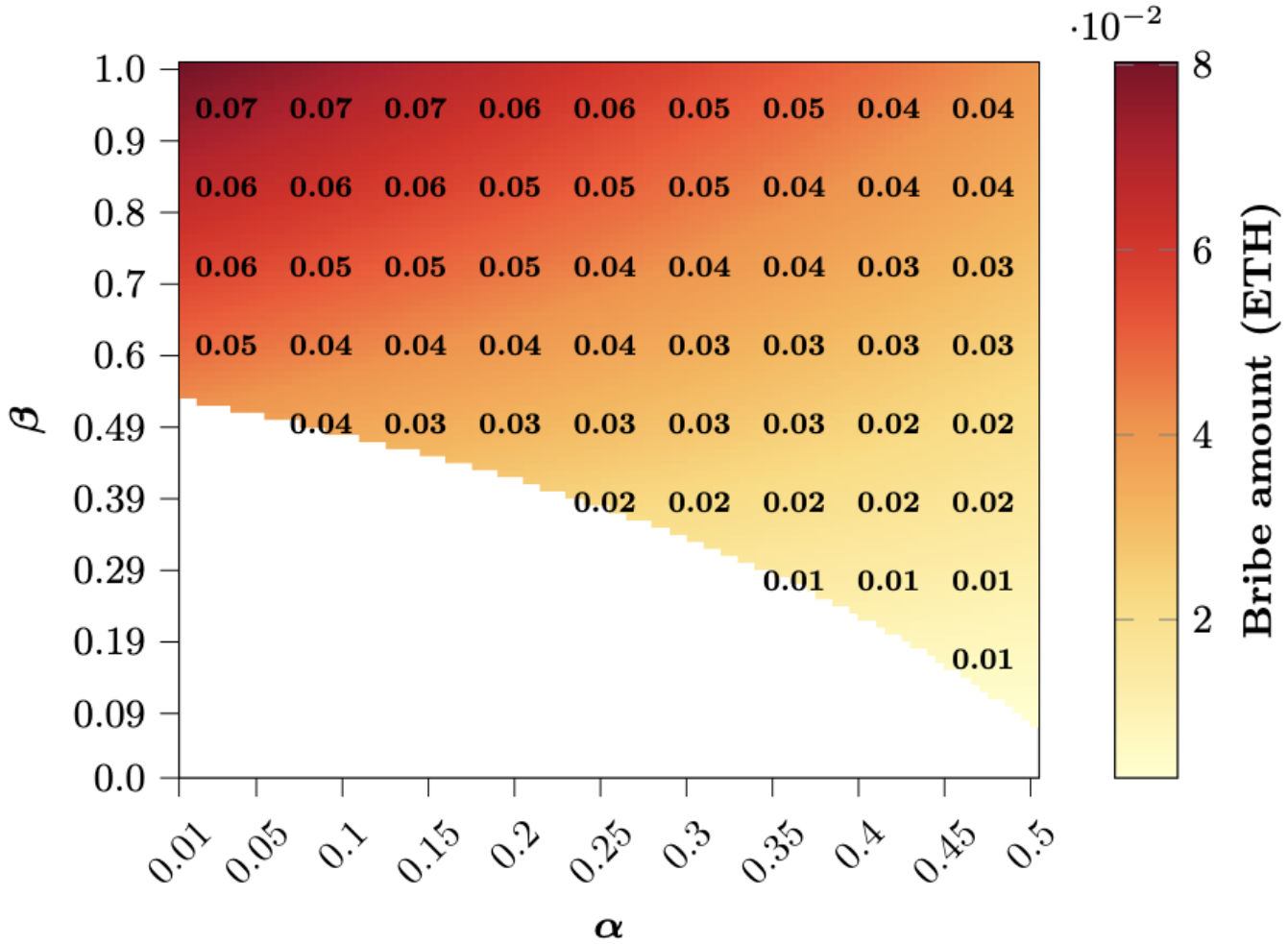
Is Resisting All in Vain?

Attack gas costs

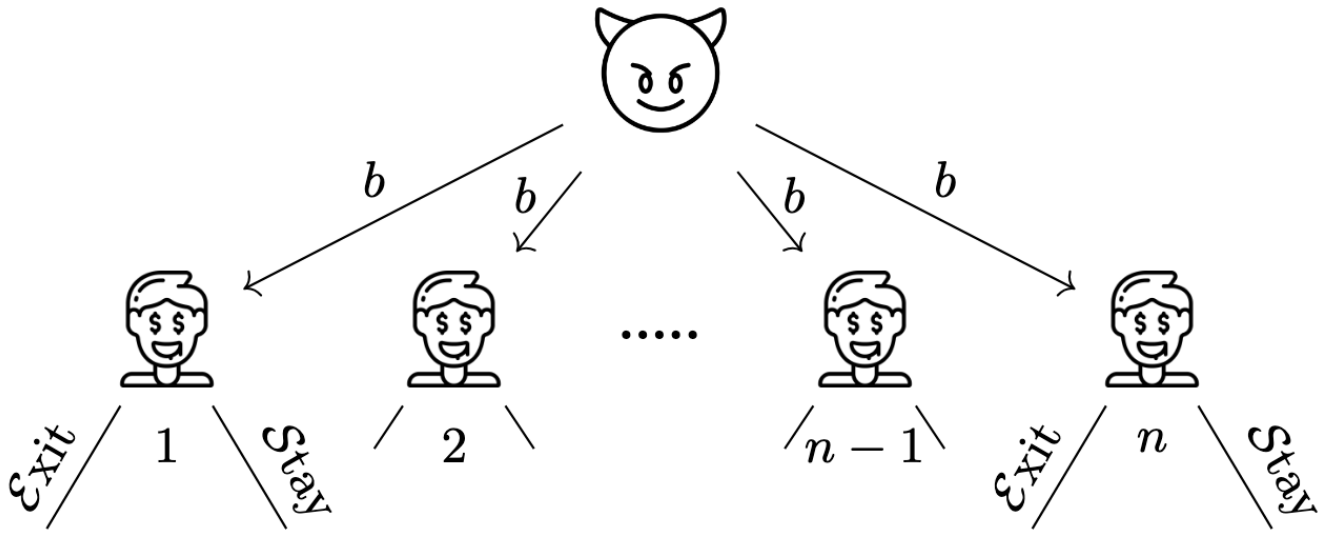
Contract \ Function	Constructor(·)	offerBribe(·)	takeBribe(·)
PayToAttest	805,876 (4.87 USD)	361,117 (2.18 USD)	252,110 (1.52 USD)
PayToExit	862,575 (5.21 USD)	225,123 (1.36 USD)	260,643 (1.58 USD)
PayToBias	1,296,940 (7.84 USD)	94,944 (0.57 USD)	137,897 (0.83 USD)

Figure: The most important functions' gas costs in our bribing contracts. ETH/USD exchange price and gas prices (3, 708 USD/ETH and 1.63 Gwei gas price) on July 25th, 2025.

Cost of ex-post reorg



Single-leader multiple-follower Stackelberg market exit game



PayToExit: attack cost

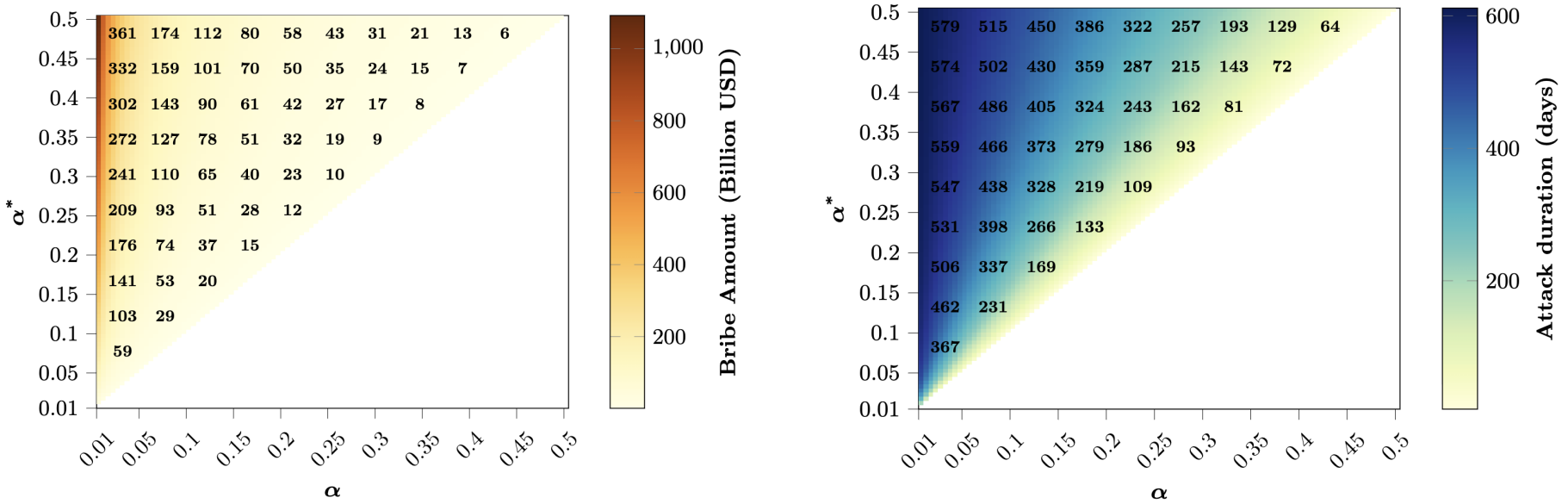


Figure: PayToExit attack cost (left) using the optimal bribe amount and duration (right).

PayToExit: attack cost

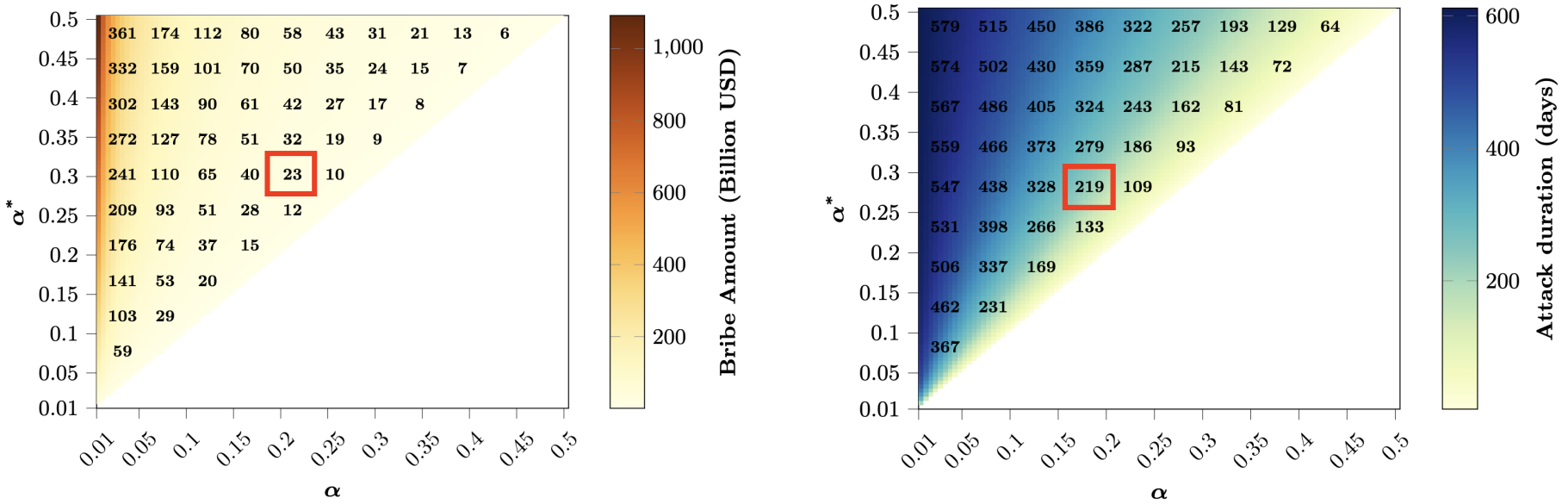


Figure: PayToExit attack cost (left) using the optimal bribe amount and duration (right).

Conclusion, Mitigations, Open problems

Comparison to previous work

Attack	Transaction reversal	Transaction ordering	Transaction exclusion	Transaction triggering	Required interference with consensus	Requires smart contract	**Payment**	Trustless for attacker	Trustless for collaborator	**Subsidy**	Compensates if attack fails
PayToAttest (this work)	✓	✗	(✓)	✗	Deep fork	✓	in-band	✓	✓	✓	(✓)
PayToExit (this work)	✗	✗	✗	✗	No fork	✓	in-band	✓	✓	✓	✓
PayToBias (this work)	✗	✗	✗	✗	No fork	✓	in-band	✓	✓	✓	✓
Simple Attack [37]	✓	✓	✓	✗	Fork	✗	in-band	✓	(✓)	✓	✗
Strong Simple Attack [37]	✓	✓	✓	✗	Fork	✗	in-band	✓	(✓)	✓	✗
Extended Attack [37]	✓	✓	✓	✗	Deep fork	✗	in-band	✓	(✓)	✓	✗
BriDe Arbitrager [49]	–	✓	✓	✓	No fork	✓	in-band	✓	(✓)	✓	✓
Guided Bribing [26]	✓	✓	✓	✗	No fork	✗	out-of-band	✗	✗	✓	✓
Effective Bribing [26]	✓	✓	✓	✗	No fork	✗	out-of-band	✗	(✓)	✓	✗
Bribe & Fork [4]	✓	✓	✓	✓	Deep fork	✓	in-band	✓	✓	✓	✓
Bribery Semi-Selfish Mining [22]	✗	✓	✓	✗	Deep fork	✗	out-of-band	✗	✗	✗	✗
Bribery Stubborn Mining [22]	✗	✓	✓	✗	Fork	✓	out-of-band	✗	✗	✗	✗
CensorshipCon [29]	✗	(✓)	✓	(✓)	No forks	✓	in-band	∼	✗	✓	✗
HistoryRevisionCon [29]	✓	✗	✓	(✓)	Deep fork	✓	in-band	✓	∼	✓	✗
GoldfingerCon [29]	–	–	✓all	(✓)	No fork	✓	out-of-band	✓	✗	✓	✗
P2W Tx Excl. & Ord [24]	✗	✓	✓	(✓)	No fork	✓	out-of-band	✓	✓	✗	✓
P2W Tx Rev., Excl. & Ord. [24]	✓	✓	✓	(✓)	Deep fork	✓	out-of-band	✓	✓	✗	✓
P2W Tx Ord. [24]	✗	✓	✗	(✓)	No fork	✓	in-band	✓	✓	✗	✗
P2W Tx Excl. [24]	✗	✗	✓	(✓)	No fork	✓	in-band	✓	✓	✗	✗

Mitigations: general

- **Increase honest rewards**:
  - raises the bribe needed to flip rational validators
  - but costs inflation / issuance (undesirable tradeoff)

Mitigations: general

- **Increase honest rewards**:
  - raises the bribe needed to flip rational validators
  - but costs inflation / issuance (undesirable tradeoff)
- **Whistleblowing incentives**: [Kelkar et al., CCS 2025]
  - reward evidence of bribery attempts or coordination
  - challenge: privacy, false reports, and what constitutes proof

https://eprint.iacr.org/2025/1582

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality [LLT22'CCS,SGSK22'CCS,Gao et al.,CCS'22]

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality [LLT22'CCS,SGSK22'CCS,Gao et al.,CCS'22]
- **PayToBias: Make randomness unbiasable:**
  - better beacon constructions [HLY20'CRYPTO]

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality [LLT22'CCS,SGSK22'CCS,Gao et al.,CCS'22]
- **PayToBias: Make randomness unbiasable:**
  - better beacon constructions [HLY20'CRYPTO]
- **PayToExit: Time-locks** to limit attacks

Open problems (crypto + mechanism design)

- **Privacy-preserving bribery contracts**:
  - can a briber **hide** targets/conditions (ZK / commitments / TEE) yet keep payments atomic? [MSSSD22'USENIX]

Open problems (crypto + mechanism design)

- **Privacy-preserving bribery contracts**:
  - can a briber **hide** targets/conditions (ZK / commitments / TEE) yet keep payments atomic? [MSSSD22'USENIX]
- **Equilibrium analysis** in richer models:
  - multiple bribers, dynamic participation, etc.

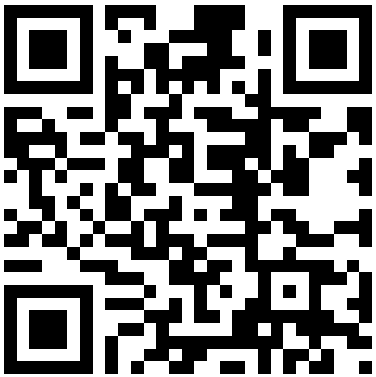
Open problems (crypto + mechanism design)

- **Privacy-preserving bribery contracts**:
  - can a briber **hide** targets/conditions (ZK / commitments / TEE) yet keep payments atomic? [MSSSD22'USENIX]
- **Equilibrium analysis** in richer models:
  - multiple bribers, dynamic participation, etc.
- **Protocol design question**:
  - how to structure messages/signatures so that "misbehavior proofs" are either impossible or too costly to verify on-chain?

Conclusion

- Smart contracts enable **trustless, scalable bribery markets**.
- This can **weaken** standard PoS "economic security" intuitions.
- Defenses require a mix of:
  - cryptographic design (what is provable/efficient on-chain?)
  - incentive design (rewards/penalties/whistleblowing)
  - consensus design (finality/randomness/participation dynamics)

Q&A



eprint

Questions?

Github repository