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

Trustless Consensus Manipulation Through Bribing Contracts

Bence Soóki-Tóth

Aarhus University

eprint

Github repository

April 19, 2026

Joint work with

István András Seres

Kamilla Kara

Ábel Nagy

Balázs Pejó

Gergely Biczók

Eötvös Loránd University and

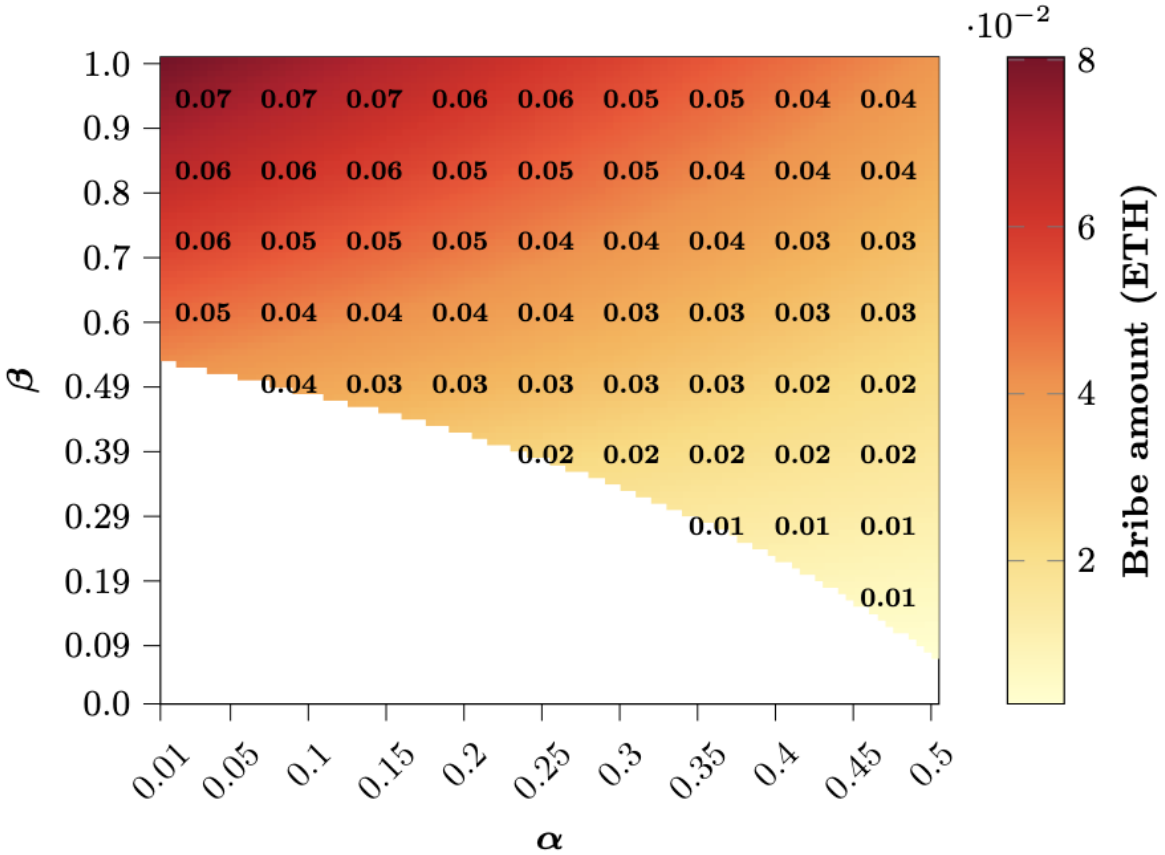
Budapest University of Technology and Economics

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

Ethereum Proof-of-Stake (PoS)

- **Permissionless protocol**
  - Anyone can *join or leave* the protocol at any time

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**

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

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

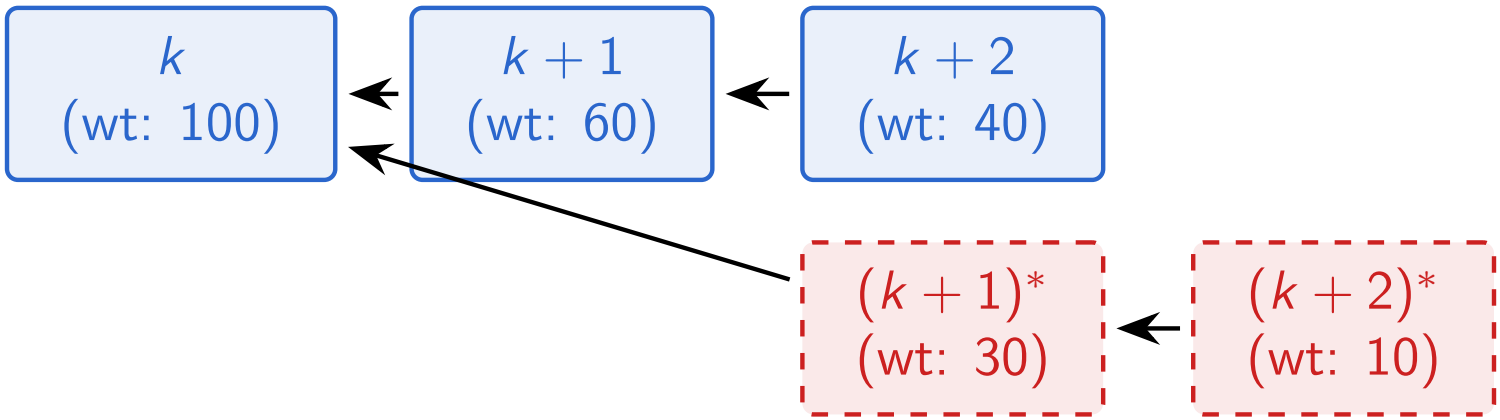


Figure: Simplified fork choice rule

Goals of Ethereum PoS Consensus

- **Safety:** The protocol does not produce conflicting states.

Goals of Ethereum PoS Consensus

- **Safety:** The protocol does not produce conflicting states.
- **Liveness:** The protocol consistently produces new blocks and finalizes transactions

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)

Latest Ethereum Updates

- **EIP-2537:** Adds efficient `BLS12-381` group and pairing operations

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

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

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 Signatures

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

BLS aggregate verification gas costs

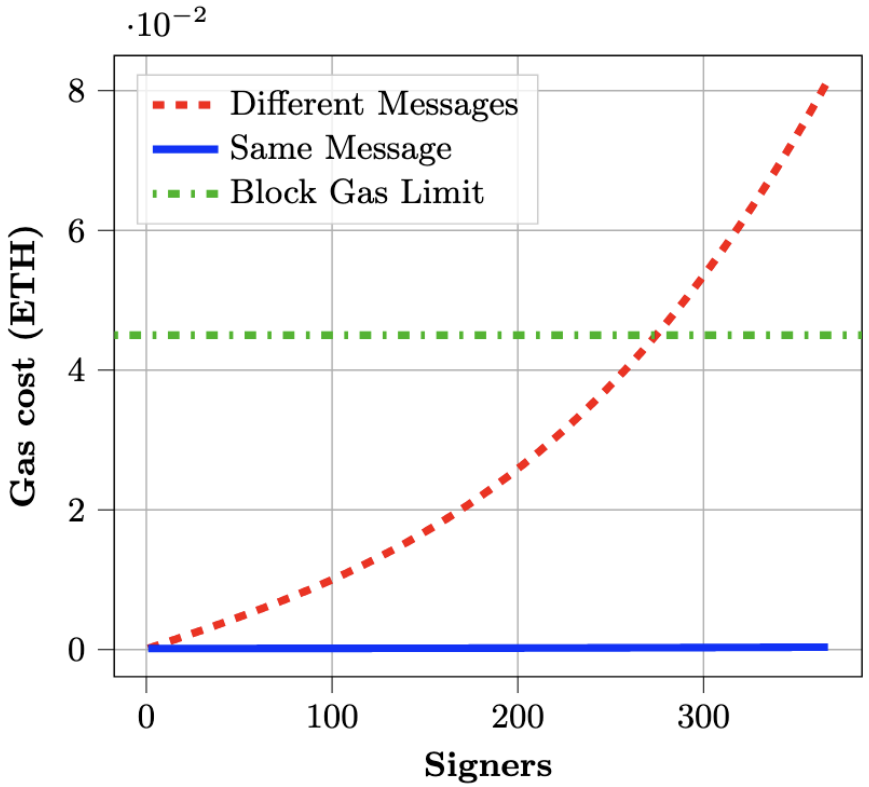


Figure: BLS batch verification gas costs.

System Model

BAR model for validators

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

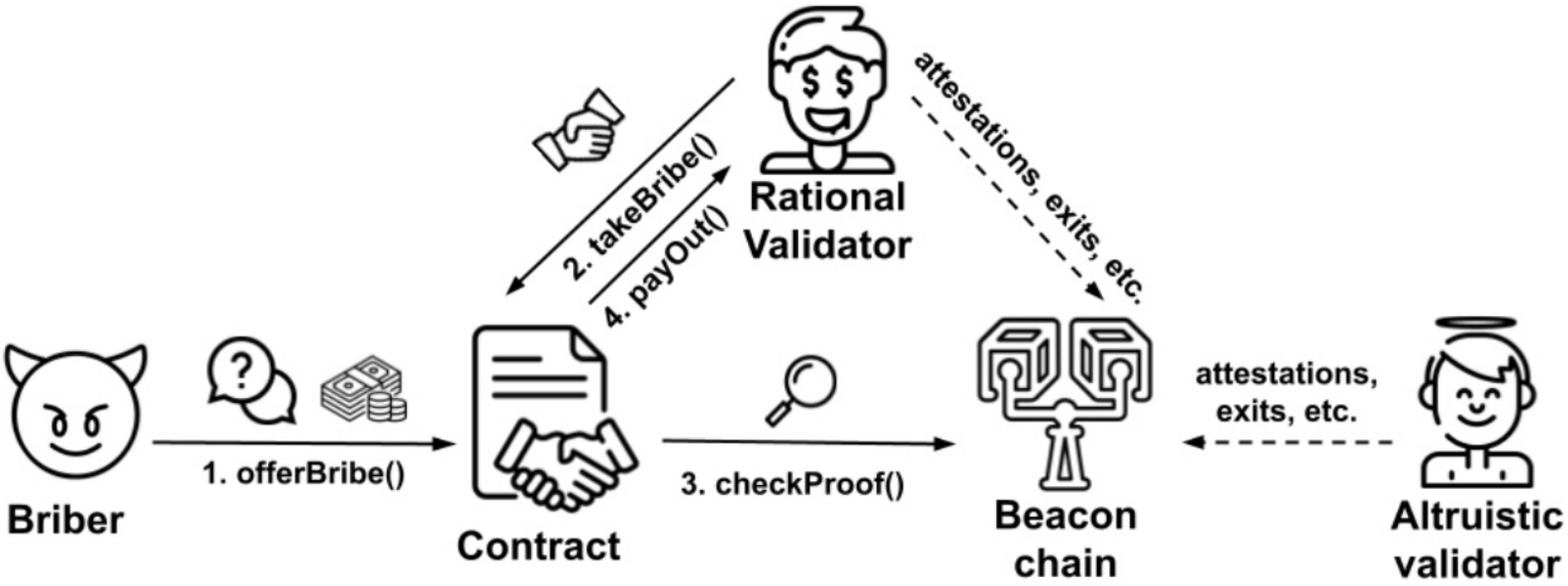


Figure: Our simple bribery system model.

Three Bribery Attacks

Bribers, Bribers on The Chain!

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 (threaten **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 (threaten **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.

PayToExit: attack cost

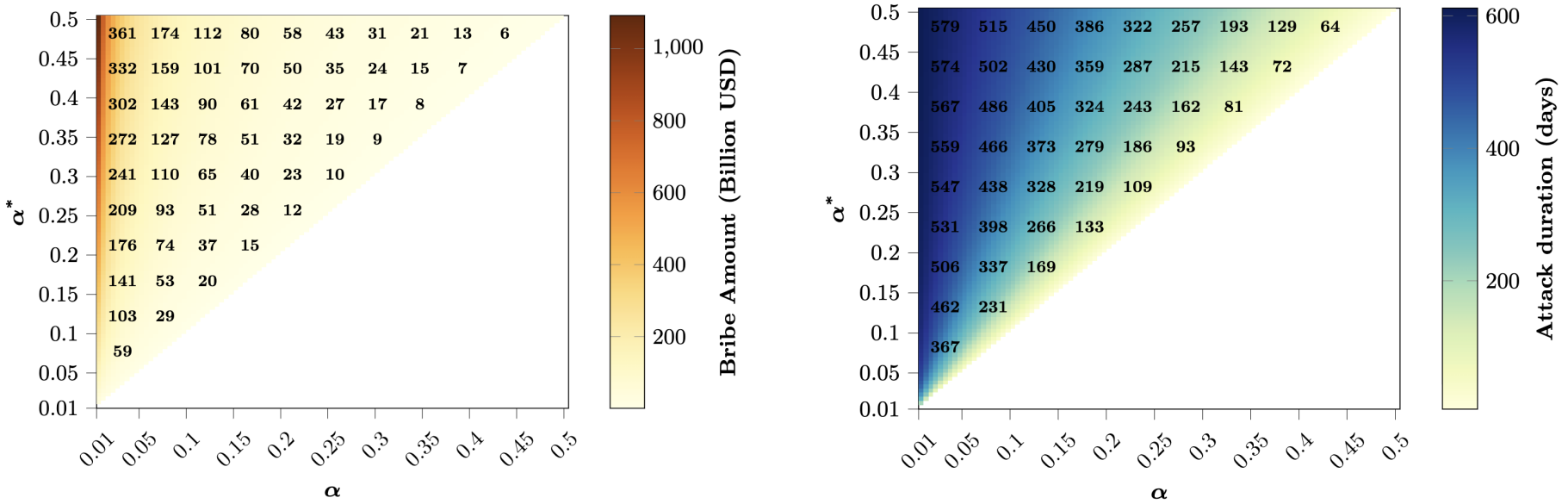


Figure: PayToExit attack cost (left) using the optimal bribe amount and duration (right).

PayToExit: attack cost

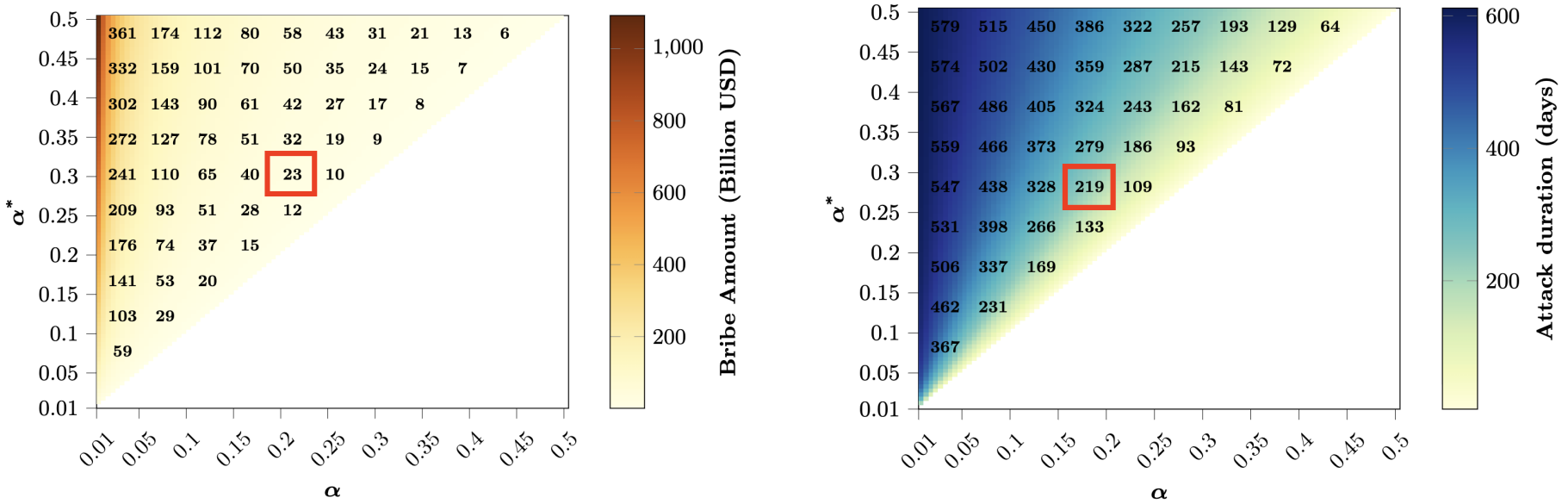


Figure: PayToExit attack cost (left) using the optimal bribe amount and duration (right).

Conclusion, Mitigations, Open problems

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**:
  - reward evidence of bribery attempts or coordination
  - challenge: privacy, false reports, and what constitutes proof

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality designs

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality designs
- **PayToBias: Make randomness unbiasable:**
  - better beacon constructions; stronger unpredictability + bias-resistance

Mitigations: targeted

- **PayToAttest: Reduce forkability / narrow bribery windows:**
  - e.g., faster finality designs
- **PayToBias: Make randomness unbiasable:**
  - better beacon constructions; stronger unpredictability + bias-resistance
- **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?

Open problems (crypto + mechanism design)

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

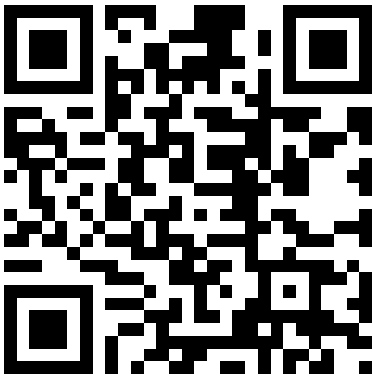
Open problems (crypto + mechanism design)

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