

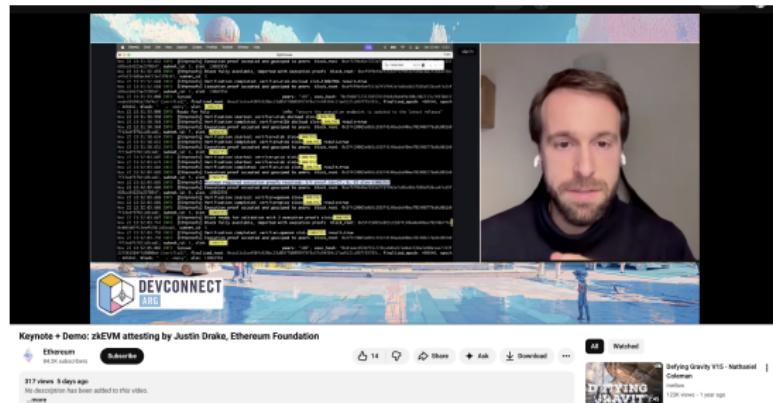
# Adversarial procurement for prover market design

Maryam Bahrani\*, **Mike Neuder**<sup>†</sup>, Matt Weinberg<sup>†</sup>  
2025 Columbia CryptoEconomics Workshop  
Wednesday – December 10<sup>th</sup>, 2025

\*Ritual, <sup>†</sup>Princeton University

# Motivation

## Ethereum's ZK endgame



- **tl;dr;** Scale the L1 with ZK proofs.
  - ★ Validators verify proofs instead of re-executing transactions.
- **Prediction:** proving, especially as we continue to scale, will remain a relatively complex task.
- **Bad scenario:** liveness of the chain depends on a handful of provers.
- **Hopeful scenario:** good market design leads to robust proof acquisition and broader participation.
- We focus on balancing **robustness** and **cost efficiency**.

## Sneak peek



- The optimal protocol trades-off between “**over-provisioning**” the number of proofs and the risk of **not receiving any** proofs.
- Many considerations change the optimal mechanism.
  1. how many players there are,
  2. how many of them are corruptible,
  3. how costly a liveness fault is,
  4. a byzantine versus crash fault adversary, etc...
- **A word of caution:** this is a subtle task!

# Model (informal)

## Base game

- A one-shot **procurement** game.
- A set of  $n$  players have a unit cost for proving.
- The “protocol” sets up a payment rule.
- The players decide whether or not to deliver (i.e., incur the cost).
- If no one delivers, the protocol incurs a **large penalty**,  $C$ .
- If at least one prover delivers, the protocol simply pays according to its payment rule.
- **Naïve solution:** tell player 1 you will pay them  $1 + \varepsilon$  if they prove.
  - ★ This payment rule induces an equilibrium! And achieves the minimal cost! ☺
  - ★ ... but it is super vulnerable to attack.

# Model (informal)

## Adversarial game

- A one-shot **procurement** game.
- A set of  $n$  players have a unit cost for proving.
- The “protocol” sets up a payment rule.
- An adversary can corrupt up to BYZ of the provers!
- The HON remaining players decide whether or not to prove.
- If no one delivers, the protocol incurs a large penalty,  $C$ .
- If at least one prover delivers, the protocol simply pays according to its payment rule.
- Naïve solution: tell player 1 you will pay them  $1 + \varepsilon$  if they prove.
  - ★ Under attack, this payment rule achieves a cost of  $C$ . ☹

# Relative sizes

1.  $C$  is big.
2. Proving cost is small.<sup>1</sup>

 MEV-Boost Bot  
@mevproposerbot

High Proposer Payment Alert! 💰  
Validator 0x8696... (id: 1899005) received 212.69 ETH.  
Builder: 0x850a00... (BuilderNet (Flashbots)) 🎉  
Slot: 12,119,999.  
Received through the [@bloXrouteLabs](#) relay.

  
beaconcha.in  
Transaction 0xd34aa2b5afdb7153c0c3a6d465b629a180...  
beaconcha.in makes Ethereum accessible to non-technical end users

4:16 PM · Jul 11, 2025 · 1,158 Views



<sup>1</sup><http://ethproofs.org/>

# Model (more formal)

## Adversarial game

1. Nature announces HON 😊, BYZ 😈, and  $C$  (numbers).
2. The protocol announces a payment rule  $p : \text{✓} \times \dots \times \text{✓} \mapsto \mathbb{R}_{\geq 0}^n$ .
3. Each prover sees  $n = \text{HON} + \text{BYZ}$  and  $p$  and decides on  $s_i \in [0, 1]$ .
4. The attacker sees  $p$  and  $s$ , and chooses who is BYZ.
5. Each HON decides  $a_i \in \{\text{✗}, \text{✓}\}$  according to  $s_i$ . Simultaneously, the attacker selects  $a_j \in \{\text{✗}, \text{✓}\}$  for  $j \in \text{BYZ}$  arbitrarily.
6. The protocol pays  $p_i(\text{✓} \times \dots \times \text{✓})$  and incurs  $C$  if all  $a_i = \text{✗}$ .

① HON=2  BYZ=2 					
--	--	--	--	--	--

① HON=2 	② 				
---	--	--	--	--	--

## Example payment rule

BYZ +1

1. HON 😇 = 1, BYZ 😈 = 2, and  $C = 3$ .
2. The protocol announces a payment rule  $p_i : \checkmark \mapsto 1 + \varepsilon$  for everyone who delivers.
3. Each prover decides  $\checkmark$ .
4. The attacker sees  $p$  and  $s$ , and chooses who is BYZ.
5. Each HON  $a_i = \checkmark$ . Simultaneously, the attacker selects  $a_j = \checkmark$  for  $j \in \text{BYZ}$ .
6. The protocol pays  $p_i(\checkmark \checkmark \checkmark) = 1 + \varepsilon$ ; total payment is  $3 + 3\varepsilon$ .
  - This is **maximally robust**, but expensive.
  - If BYZ = 1000. Do we really want to pay  $BYZ + 1$  every slot? Can we be more efficient?

# Protocol cost

1<sup>st</sup> of two math slides...

- A payment rule takes an action vector (s & s), and outputs per-player payments:  $p_i(\mathbf{a}) \mapsto$  player  $i$  payment.
  1. Pay a single player  $1 + \varepsilon$  if they deliver.
  2. Pay  $BYZ+1$  players  $1 + \varepsilon$  each if they deliver.
  3. Choose a lottery prize and give it to one random person who delivers.

## Definition (Single outcome cost)

The protocol cost under a payment rule  $p$  and action profile  $\mathbf{a} \in \{0, 1\}^n$  is the total payment to provers, plus the penalty  $C$  if no proofs are delivered,

$$C_p(\mathbf{a}) := \underbrace{\sum_{i=1}^n p_i(\mathbf{a})}_{\text{payments}} + C \cdot \underbrace{\prod_{i=1}^n (1 - a_i)}_{\begin{array}{l} 1 \text{ if no one delivers} \\ \end{array}}$$

# The protocol's objective

2<sup>nd</sup> of two math slides...

- Let  $A$  denote the attacker set and  $H$  the honest set.

## Definition (Loss)

The loss for a payment rule  $p$  and an equilibrium strategy profile  $\mathbf{s} \in [0, 1]^n$  is

$$\ell(p, \mathbf{s}) := \max_{A, \mathbf{a}_A} \left\{ \mathbb{E}_{\mathbf{a}_H \leftarrow \mathbf{s}_H} \left[ C_p(\mathbf{a}_A, \mathbf{a}_H) \right] \right\}$$

- Protocol's objective is to minimize loss.
- The *optimal* payment rule is,  $p^* := \arg \min_p \min_{\mathbf{s}} \{\ell(p, \mathbf{s})\}$ .
- Turns out ↑ is a hard math problem... but **good** news, we can get within 1 proof-cost (5¢) of  $p^*$ !

# The optimization problem

Our “off-by-one” equilibrium

- We find that sometimes, the optimal equilibrium sets  $s_1 = 1$ .
- **Tiny example:** HON= 1, BYZ= 1,  $C = 3$ .
  - ★ Optimal equilibrium is  $\mathbf{s}^* = (1, 1/2)$ .
  - ★ Payment rule

$$p_1(10) = 3/2, p_1(11) = 1/2 \\ p_2(11) = 1.$$

- ★ Optimal loss is  $3/2$ .
- **Good news:** whenever  $s_1 = 1$ , we have the optimal payment rule.
- **Bad news:** sometimes  $s_1 < 1$ .
- **Silver lining:** we can always round  $s_1 \rightarrow 1$ , and implement that version of the payment rule.
- Asymptotically, the optimal loss scales as  $\mathcal{O}(\log C)$ .

# Generalizing our nearly optimal payments

- **Payment rule recipe:**
  1. Pre-commit to paying the same amount (or nothing) **no matter how many proofs were delivered.**
  2. Tell player 1 to deliver **100% of the time.**
  3. Tell a **committee** of the remaining players to **mix**, but they only get paid if player 1 delivers.
  4. Solve a well-defined math problem to determine the **committee size** and their **mixing probability**.
- **What this gets us:**
  - ★ The attacker only has two options:
    1. Corrupt  $s_1$  and try to get the committee to fail, or
    2. Not corrupt  $s_1$  and make us pay (a constant).
  - ★ So we can just **balance** the two.
- **Counter-intuitive:** we only pay when we *least* need extra proofs.
- **Counter-counter intuition:** we *detect* when we are being attacked!

# Takeaways for practitioners

## Designate and committee sizing

- **Key takeaway!** Designate to a single player.
  - ★ Can think of it as a rotating leader election, with backup provers delivering some of the time.
  - ★ If there is an attacker, this limits the damage they can do.
- Committee **sizing** and **delivery probability** are key factors.
  - ★ Consider HON=4, BYZ=3. With one designate, there are six potential committee members.
    - $C = 10 \implies$  opt. committee **size 6**, each **delivers 28%** of the time.  
**Intuition:** the attacker corrupts fewer mixers.
    - $C = 40 \implies$  opt. committee **size 4**, each **delivers 70%** of the time.  
**Intuition:** drop two of the mixers to give everyone else more mass and reduce cross-term failures.
    - $C = 200 \implies$  opt. committee **size 3**, each **delivers 98%** of the time.  
**Intuition:** we converge to  $\text{BYZ}+1 = 4$  designates.

## A natural objection

- Everyone's payment depends on one player!
- **Counter-point:** Ethereum *already* correlates attester rewards to proposer behavior.
- We really think this payment rule is cost-efficient, implementable, and robust. But if it's not feasible for your use-case, then...
  1. You can condition payments on more than one player.
    - **Ethereum's inactivity leak:** Validators are only paid if 2/3 of the chain is online.
  2. You can randomly select a set of provers to pay  $1 + \varepsilon$  on delivery.
    - This weakens the attacker, who now only corrupts a random subset.
    - **Ethereum's secret leader election:** Proposers selected in private.
  3. You might only be concerned with the crash fault adversary.
    - This weakens the attacker, who can only cause  $a_i = 0$ .
    - Under this adversary, we can always achieve optimal loss.

# Takeaways for practitioners

## Staking & slashing

- Thought experiment:
  - ★ Let people signal, **with stake**, the **provers** that they think are the **most reliable** and **offer the best revenue-share**.
  - ★ If the **prover shirks** when the protocol designates it, the stake is **slashed** and the loss is socialized among those who staked with that **prover**.
- Does this sound weird? Or familiar? ... What do staking pools do?
  - ★ Let people signal, **with stake**, the **validator** that they think are the **most reliable** and **offer the best revenue-share**.
  - ★ If the **validator equivocates** when the protocol designates it, the stake is **slashed** and the loss is socialized among those who staked with that **validator**.
- A pretty reasonable way to “insure” against attacks!
- With  $\text{stake} \geq C - 1$ , the protocol can procure with a **cost of 1**. ☺  
*Can you think of a payment rule that works?* 
$$\begin{cases} p_1 = 1 & \text{if } a_1 = 1 \\ p_1 = -(C - 1) & \text{otherwise} \end{cases}$$

# Takeaways for practitioners

This is a delicate task – YMMV!

- Many considerations change the optimal mechanism
  1. how many players there are,
  2. how many of them are corruptible,
  3. how costly a liveness fault is,
  4. a byzantine versus crash fault adversary,
  5. (new) importance of uncorrelated payments,
  6. (new) the attackers objective,
  7. (new) when the equilibrium is revealed,
  8. (new) whether you can slash, etc...
- Also *this project is WIP!* More details to follow soon™ ☺

thanks :)

