Proof of Stake Blockchain Protocols

Aggelos Kiayias

aggelos.kiayias@ed.ac.uk

based on joint works with:
Alexander Russell, Peter Gazi, Bernardo David, Roman Oliynykov

gratefully acknowledging:
project CODAMODA  project PANORAMIX
The Ledger Objective

• First formal definition of the objective of a “robust transaction ledger” was formulated by Garay, K, Leonardos in [GKL15] https://eprint.iacr.org/2014/765

Follow up work refined model and definition:

[KP15] - property definitions
[PSS17] - partial synchrony / property definitions
[BMTZ17] - simulation based definition / composition
Realizing the Ledger

• The ledger can be realized by bitcoin

• But the protocol exhibits significant **scalability** and **energy efficiency** disadvantages.

• Can we realize it in a more efficient way?

“**Proof-of-Stake**” (PoS) was one alternative that was discussed early on in the bitcoin forum
PoS background

generating the next block in bitcoin is like an election

A miner is elected with probability proportional to its hashing power. **Collisions** may occur but they can be resolved by the longest chain rule or a similar concept.
PoS

Use **stake** instead of **hashing power**.

Define the set of miners to be the set of all stakeholders, as reported in the ledger.

Use a **randomised process** that takes the current stake into account to elect the next miner eligible to produce a block.
Performance vs. Decentralization

Initial stakeholder distribution should have honest majority, but this can shift over time

- no PoW requirement - blockchain runs at maximum synchronization speed and scales without increasing costs

- Performance / energy efficiency
- Decentralisation

- Centralized database
- Byzantine Agreement Protocols
- Proof of Stake
- Bitcoin
Design Challenges Specific to PoS

• **Grinding attacks.** The adversary may try to bias the random election process in its favor, trading hashing power for a more favorable protocol execution.

• **Nothing-at-stake.** the adversary may try multiple alternative histories (even from any point in the past).

• **Circularity.** if injecting fresh randomness is done via the blockchain - how can we avoid circularity in the security argument? (*blockchain is secure b/c of randomness, and randomness is unbiased b/c of blockchain*).
The Ouroboros PoS Protocols

- **Ouroboros**: synchronous setting, corruptions with delay, slow & clean beacon implementation (PVSS-based).

- **Ouroboros Praos**: semi-synchronous setting, fully adaptive corruptions, fast & dirty beacon implementation (hash-based).
Design Methodology

• **Stage 1. Static State.** Assume that initial stakeholder distribution remains the root of trust of the system.

• **Stage 2. Using a semi-trusted beacon (some leakage / adversarial control)** Assume a randomness beacon emits a seed in regular intervals and show how this can be utilized to let the roof to trust stakeholder distribution evolve.

• **Stage 3. Implement the beacon via protocol that uses the blockchain.** remove the trusted beacon by having the elected subset of stakeholders simulate it cryptographically.
Synchronous Setting

- Time is divided in rounds (we will call them \textit{slots})
- Messages are sent through a “diffusion” mechanism
- The adversary is rushing and may:
  1. spoof messages
  2. inject messages
  3. reorder messages
Ouroboros: Static Stake

Block Content:
- \( U_1, s_1 \)
- \( \vdots \)
- \( U_n, s_n \)

Seed

Blocks:
- \( B_0 \)
- \( B_1 \)
- \( B_2 \)
- \( B_3 \)
- \( B_4 \)
- \( \ldots \)
- \( B_m \)

Slots:
- \( L_1 \)
- \( L_2 \)
- \( L_3 \)
- \( L_4 \)
- \( L_5 \)
- \( L_6 \)
- \( \ldots \)
- \( L_R \)

Elected Leaders:
- \( T_1 \)
- \( \vdots \)
- \( T_n \)

SIG by \( L_2 \)

SIG by \( L_3 \)

SIG by \( L_5 \)

SIG by \( L_6 \)

SIG by \( L_R \)

When conflicts occur parties choose which chain to extend based on **longest chain rule**

weighted by stake sampling \( F(\text{seed}) \)
Forks and Protocol Executions

Characteristic string

\[ w_i = \begin{cases} 
0 & \text{if } i\text{-th slot belongs to an honest party} \\
1 & \text{if } i\text{-th slot belongs to a malicious coalition} 
\end{cases} \]
Analysis

• The adversary is at a much better position in this protocol execution compared to Bitcoin’s PoW-based execution.

• it can see ahead of time how stakeholders are activated.

• it can generate multiple different blocks for the same slot at any time without cost.

• it can wait and act just before an honest party comes online.
Forkable Strings

The characteristic strings the adversary prefers!

Not Forkable!

Forkable!
Reach/Margin of Closed Forks

Theorem: A string is forkable iff for some closed fork it holds that margin is $\geq 0$

Reach: maximum reach across all tines

Margin: the penultimate reach across edge-disjoint tines
Recursive Formula for Reach & Margin

\[
[r(w_1), \mu(w_1)] = [r(w) + 1, \mu(w) + 1]
\]

\[
[r(w_0), \mu(w_0)] = \begin{cases} 
[r(w) - 1, 0] & \text{if } r(w) > \mu(w) = 0 \\
[0, \mu(w) - 1] & \text{if } r(w) = 0 \\
[r(w) - 1, \mu(w) - 1] & \text{otherwise}
\end{cases}
\]

it is possible for the adversary to compensate for margin, by spending reach

reach never drops below 0

reach and margin decrement
2D Random Walk

Start at 0, flip a biased coin: right for “1” and left for “0”

Variable **reach** performs a 1D random walk when positive

Variable **margin** performs a 1D random walk when negative

Variable **margin** sticks to 0 when **reach** is positive
Forkable strings are rare!

$w = 0101 \ldots$

\[ \sqrt{n} \quad \sqrt{n} \quad \sqrt{n} \]

$\rho \geq \delta \sqrt{n} \land (\mu \geq -\delta \sqrt{n})$

$-\delta \sqrt{n} \leq \mu \leq \rho < \delta \sqrt{n}$

$\mu < -\delta \sqrt{n}$

density is $2^{-\Omega(\sqrt{n})}$

error is $2^{-\sqrt{n}}$
Divergence

• Given a string & a pair of viable tines for it, their **divergence** is the # of blocks after the two tines fork.

• The divergence of a string is the maximum divergence over all possible tines.

• **Theorem.** (informal) The divergence of any string, is realized by a specific forkable substring of length at least as large as the divergence.

  • **Corollary.** Forkable strings are rare, thus the divergence of a string length $R$ is bounded by $k << R$ with error $\exp(-\Omega(\sqrt{k}) + \log R)$ assuming adversarial stake ratio at most $(1 - \epsilon)/2$
Semi-Synchronous Setting

- Time is divided in \textbf{slots} — as before.

- Messages are sent through a \textbf{diffusion} mechanism + the adversary is rushing — as before.

- Messages can be \textbf{delayed} by $\Delta$ slots, an unknown to the honest parties parameter, which is only determined at execution time.
Ouroboros Praos

• Designate periods of **silence** to allow the opportunity to parties to synchronize in the semi-synchronous setting.

• Determine slot leader eligibility **independently** via a private test based on a VRF, to prevent the adversary from learning the slot leader schedule ahead of time.

• Use a key-evolving signature to **protect** past slots that were assigned to honest parties.
Analysis Approach

- Realize key-evolving signature and VRF with malicious key generation, in the UC setting.

- Analyze combinatorially, using a generalization of forkable strings, the hybrid protocol.
Crypto Primitives

• Universally Composable Key-evolving signature; we prove it can be derived by a game based key-evolving signature.

• Universally Composable VRF (that retains randomness even under malicious key generation): we prove a construction in the RO model based on 2Hash-DH construction [JKK14].

\[ x \mapsto H(x, H(m)^k) \]

\[ \text{public-key } y = g^k \quad \text{EQDL}(k : u = H(m)^k \land y = g^k) \]
Praos Slot Leader Selection

Rule. you are a slot leader if

\[ H(rnd, \text{slot}, H(rnd, \text{slot})^k) < T_{stk} \]

\[ \text{public-key } y = g^k \]

Calibrate the target so that a party with relative stake \( \alpha \)
will succeed with probability \( 1 - (1 - f)^\alpha \)

\( f \) is a protocol parameter = rate of non-silent slots
$\Delta$-Forks

If two honest slots $i, j$ satisfy $i + \Delta < j$ then the $j$ player should be aware of the actions of the $i$ player.
Δ-Divergence

• Δ-Divergence of a string is the corresponding notion of divergence but now taking Δ delays into account.

• Objective (as before): show that Δ-Divergence of the protocol execution is small.
Semi-Synchronous to Synchronous Reduction

\[
\begin{align*}
\rho_{\Delta}(\varepsilon) &= \varepsilon, \\
\rho_{\Delta}(\bot \parallel w') &= \rho_{\Delta}(w'), \\
\rho_{\Delta}(1 \parallel w') &= 1 \parallel \rho_{\Delta}(w'), \\
\rho_{\Delta}(0 \parallel w') &= \begin{cases} 0 \parallel \rho_{\Delta}(w') & \text{if } w' \in \bot^{A-1} \parallel \{0, 1, \bot\}^*, \\ 1 \parallel \rho_{\Delta}(w') & \text{otherwise.} \end{cases}
\end{align*}
\]

Lemma

Let \( w \in \{0, 1, \bot\}^* \). Then \( \text{div}_{A}(w) \leq \text{div}_{0}(\rho_{\Delta}(w)) \).

Note however that forkable string density does not imply immediately (as before) small divergence as the reduction mangles the underlying distribution.
Dominant Distribution

relative honest stake = \( \alpha \)

\[
\begin{align*}
\Pr[w_i = \bot] &= 1 - f \\
\Pr[w_i = 0] &= (1 - f)(1 - (1 - f)^\alpha) \\
\Pr[w_i = 1] &= f - (1 - f)(1 - (1 - f)^\alpha)
\end{align*}
\]

Theorem: the dominant distribution via \( \rho_\Delta(\cdot) \)

still yields a sequence of Bernoulli trials with honest slot probability \( \geq \alpha(1 - f)^\Delta \)

- \( \Delta \)-divergence of a string length \( R \) is bounded by \( k << R \)

with error \( \exp(-\Omega(\sqrt{k}) + \log R) \)

assuming \( \alpha(1 - f)^\Delta \geq (1 + \epsilon)/2 \)
**Design Methodology**

- **Stage 1. Static State.** Assume that initial stakeholder distribution remains the root of trust of the system.

- **Stage 2. Using a semi-trusted beacon (some leakage / adversarial control)** Assume a randomness beacon emits a seed in regular intervals and show how this can be utilized to let the roof to trust stakeholder distribution evolve.

- **Stage 3. Implement the beacon via protocol that uses the blockchain.** remove the trusted beacon by having the elected subset of stakeholders simulate it cryptographically.
Dynamic Stake

\[ U_1, s_1 \rightarrow U_{n+1}, s_{n+1} \rightarrow U'', s'' \]

Beacon

randomness beacon
Leakage/Adversarial Control

- [Ouroboros] Adversary is allowed to learn the beacon value ahead of time.

- [Ouroboros Praos] Adversary is additionally allowed to reset the value of the beacon a number of times.
Design Methodology

- **Stage 1. Static State.** Assume that initial stakeholder distribution remains the root of trust of the system.

- **Stage 2. Using a semi-trusted beacon (some leakage / adversarial control)** Assume a randomness beacon emits a seed in regular intervals and show how this can be utilized to let the roof to trust stakeholder distribution evolve.

- **Stage 3. Implement the beacon via protocol that uses the blockchain.** Remove the trusted beacon by having the elected subset of stakeholders simulate it cryptographically.
Ouroboros: GOD coin tossing

• For every stakeholder when each epoch starts:

Use publicly verifiable secret-sharing (PVSS) for distributing commitment openings
Ouroboros-Praos: Hash-based Beacon

• Collect a vector of the VRF values inserted by each slot leader in a block.

• Set the beacon value to the hash of the VRF values.

• **Grinding** becomes possible, but can be controlled by using the [GKL15] $q$-bounded model (the adversary has a quota of hashing queries per slot).
Incentive Structure

How to incentivise parties to execute the protocol?

We introduce the concept of “Input-Endorsers”

A sequence of transactions need to be endorsed in order to be included in a block.

Endorsed sequence can be included in any upcoming block up to $2k$ slots in the future (inclusive).
Assumptions about protocol costs

• Our Assumptions:
  
  • Issuing blocks is easy (blocks contain only endorsed sequences of transactions, hence effort to verify transactions is passed to the endorsers).

  • **Expensive actions** are:

  • Running the GOD protocol to simulate the randomness beacon. (need to issue commitments and open them)

  • Endorsing sets of transactions (need to verify them)
Reward Mechanism

• Epoch based.

• After each epoch stabilizes, provide rewards for the following acts:
  1) (just) being a slot leader.
  2) endorsing a set of inputs.
  3) sending messages for the MPC protocol.

Theorem. Ouroboros is an approximate Nash equilibrium.
Ouroboros Performance

Implementation: Parity Rust-based Extensible Ethereum
40 nodes in Amazon Cloud, Slot length = 5s
median TPS = 257.6
Related Works

- NXT, PeerCoin
- Cryptocurrencies w/o PoW, Bentov, Gabizon, Mizrahi
- Algorand by Jing Chen and S. Micali
- Snow White by P. Daian, R. Pass and E. Shi
- Sleepy consensus by R. Pass and E. Shi.
- Fruitchain by R. Pass and E. Shi
Reading


- Forkable Strings are Rare. https://eprint.iacr.org/2017/241

Proof of Stake Blockchain Protocols

Aggelos Kiayias

aggelos.kiayias@ed.ac.uk

based on joint works with:
Alexander Russell, Peter Gazi, Bernardo David, Roman Oliynykov

gratefully acknowledging:
project CODAMODA  
project PANORAMIX