Tutorial:
The Ouroboros Protocol Family

Advances in Financial Technologies, 2019

Christian Badertscher & Aggelos Kiayias

The University of Edinburgh & IOHK
The Goal of Blockchain Protocols

- Implement an immutable transaction ledger...
Immutable Ledger Properties

- Submit transactions
- Get included into the ledger (if valid)
- Everyone can access ledger
- Ledger can’t be changed (immutable)
The Ledger Functionality

A Public Ledger
The Ledger Functionality

- The functionality formalizes the relevant blockchain properties and the limited capabilities of an adversary.
  - E.g., common state, well-formed blocks, recent transactions etc.
The Ledger Functionality

- The functionality formalizes the relevant blockchain properties and the limited capabilities of an adversary.
  - E.g., common state, well-formed blocks, recent transactions etc.
- Important: It captures the service provided to any cryptographic protocol.

**Applications:** Incentive Mechanisms, Poker, general MPC
Realizing the Ledger

- Implement an immutable transaction ledger...

But:

- Avoid a central trusted entity
- Allow dynamic and easy participation
- Be permissionless and accessible to anyone (read and write)
Realizing the Ledger

- **The case of Bitcoin:**
  - Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.
Realizing the Ledger

The case of Bitcoin:

- Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

\[ B_0 \]

Genesis Block
The case of Bitcoin:

Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

Lottery-style (simplified):

Find nonce \( N \) s.t. \( \text{Hash}(N, \text{tx}..., \text{Hash}(B_{i-1})) < T \)

Observation:
More hashing power \( \rightarrow \) better chances to produce blocks.
The case of Bitcoin:

- Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

A Tree Structure (Forks)
Realizing the Ledger

- **The case of Bitcoin:**
  - Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

Blockchain Properties [GKL15,PSS17]:

- **Common-prefix (CP):** Honest miners share a consistent common prefix.
- **Chain-growth (CG):** The number of blocks increases over time.
- **Chain-quality (CQ):** A guaranteed fraction of honestly contributed blocks.

→ Ledger can be realized assuming honest majority of hashing power
Realizing the Ledger

A very nice blockchain feature: **Dynamic availability (DA).**

- Parties join and leave at will. They need to bootstrap a chain when (re-) joining.
  → Easy in Bitcoin: “longest-chain rule” (general: most difficult chain).

- Number of online/offline parties changes over time
  → Analysis must account for that.

- No *a priori* knowledge of participation levels is required by the protocol.

- Unannounced disappearance.
The case of Bitcoin:

- Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

- **Bitcoin is not energy efficient** as the hash-based lottery consumes a lot of energy to ensure the protocol’s security.
The case of Bitcoin:
- Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.

- **Bitcoin is not energy efficient** as the hash-based lottery consumes a lot of energy to ensure the protocol’s security.

Proof-of-Stake to the rescue!
Realizing the Ledger

Proof-of-Stake Blockchains:

- Use **stake** (a virtual resource) instead of hashing power.
- Miners = **Stakeholders**.
- Next **stakeholder** to produce block **elected with probability proportional to stake**.
Realizing the Ledger

Proof-of-Stake Blockchains:
- Use Stake (a virtual resource) instead of hashing power.
- Miners = Stakeholders.
- Next stakeholder to produce block elected with probability proportional to stake.

Two categories:
Nakamoto-style consensus (e.g., Ouroboros, Snow White)
BFT-style consensus (e.g., Algorand, Casper, Ouroboros-BFT)
Realizing the Ledger

Proof-of-Stake Blockchains:
- Use Stake (a virtual resource) instead of hashing power.
- Miners = Stakeholders.
- Next stakeholder to produce block elected with probability proportional to stake.

Complications of PoS vs. PoW:
- **PoS has costless simulation:**
  No physical resources to create blocks: several transaction histories could be generated “in the adversaries head”.

- **Long-Range attacks in the threat model:**
  Adversary tries to deceive (new) participants into believing the “wrong” history (which are cheap to generate).
Tutorial Overview

- The development steps of a pure PoS-based blockchain protocol in the dynamic availability setting.
  - Security follows from the “honest majority of stake” assumption.

- Start with the initial version and refine it until all the security requirements are achieved.
Tutorial Overview – Main Content

Ouroboros
“Classic”
(Crypto 17)

Ouroboros
Praos
(Eurocrypt 2018)

Ouroboros
Genesis
(CCS 2018)

Ouroboros
Chronos
(In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework
Tutorial Overview – Main Content

Ouroboros
“Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)
Ouroboros “Classic” (Crypto 17)
Ouroboros Praos (Eurocrypt 2018)
Ouroboros Genesis (CCS 2018)
Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis
Ouroboros “Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis
Ouroboros “Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
Tutorial Overview – Additional Features

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros Chronos (In submission, 2019)

→ Ouroboros BFT

An extremely simple BFT protocol that follows from the Ouroboros Classic analysis

→ Ouroboros Crypsinous (S&P 2019)

Ouroboros with Privacy
Ouroboros – Protocol Design

Ouroboros “Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)
+ Full dynamic availability
+ Bootstrapping from Genesis
+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
The General Picture and Assumed Resources

Blockchain Protocol

Clock Functionality

Random Oracle

Diffusion Network

Genesis Block

Setup Resources

Blockchain Protocol

Arbitrary Strategy

Blockchain Protocol

Arbitrary Strategy
The General Picture and Assumed Resources

- **Synchrony**: Time-stamps, slots
- **Unpredictability**: Hash-functions
- **Communication**: Multicast/Broadcast, limited delay $\Delta$

Blockchain Protocol

- **Functionality**
  - Random Oracle
  - Diffusion Network

Genesis Block

Setup Resources

- **Arbitrary Strategy**
Ouroboros – Protocol Design

Epoch 1

Epoch 2
Ouroboros – Protocol Design

Public address: verification key \( vk_j \) of a signature scheme

\( \text{\#} \) = A number of coins (tokens) \( s_j \) associated to \( vk_j \)

Random seed

\( G \)

Epoch 1

Epoch 2
Ouroboros – Protocol Design

In each round:

1.) Determine the current longest valid chain.
2.) Determine Slot-Leadership
3.) Slot leader: Pack transactions, create and publish block

Epoch 1

Epoch 2
Ouroboros – Protocol Design

In each round:

1.) Determine the current longest valid chain.
2.) Determine Slot-Leadership
3.) Slot leader: Pack transactions, create and publish block

Epoch 1

Epoch 2
Ouroboros – Protocol Design

Slot Leadership in Classic: Random process (“Coin Tossing”)

\[ F(\text{G}, \text{Random seed}, \text{seed, slot}) \rightarrow \text{G/}/\text{G/}/\text{G/} \]
Ouroboros – Protocol Design

Slot Leadership in Classic: Random process (“Coin Tossing”)

\[ F(\ldots, \ldots, \ldots, \text{seed}, \text{slot}) \rightarrow \text{biased coin toss} \]

For example:

Biased-Coin Toss
Slot Leadership in Classic: Random process ("Coin Tossing")

\[ F(\, \overset{\_}{\overset{\_}{\_}}, \, \overset{\_}{\overset{\_}{\_}}, \, \overset{\_}{\overset{\_}{\_}}, \overset{\_}{\overset{\_}{\_}}, \, \text{seed}, \, \text{slot}) \rightarrow \overset{\_}{\overset{\_}{\_}}/\overset{\_}{\overset{\_}{\_}}/\overset{\_}{\overset{\_}{\_}} \]

For example:

Biased-Coin Toss

1/6
Ouroboros – Protocol Design

Slot Leadership in Classic: Random process ("Coin Tossing")

\[ F(\ldots, \text{seed}, \text{slot}) \rightarrow \]
Ouroboros – Protocol Design

Slot Leadership in Classic: Random process (“Coin Tossing”)

\[ F(\ldots, \text{seed}, \text{slot}) \rightarrow \]
Slot Leadership in Classic: Random process (“Coin Tossing”)

\[ F(\ldots, \ldots, \ldots, \text{seed}, \text{slot}) \rightarrow \ldots \]

Simplified model “Semi-adaptive” (will be strengthened later):
- Adversary cannot adaptively react on the (public) slot-leader schedule.
  (As an approximation: think of a static corrupted set of parties)
Ouroboros – Protocol Design

In each round:

1.) Determine the current longest valid chain.
2.) Determine Slot-Leadership
3.) Slot leader: Pack transactions, create and publish block
Ouroboros – Protocol Design

Block structure:

- Hash pointer to prev. block
- Content / Transactions
- Slot number
- Signature of slot leader

Epoch 1

Epoch 2
Ouroboros – Protocol Design

In each round:

1.) Determine the current longest valid chain.
2.) Determine Slot-Leadership
3.) Slot leader: Pack transactions, create and publish block
Ouroboros – Protocol Design

Chain Selection Rule:

Adopt a valid new chain if it is longer and does not fork by more than k blocks from local chain.

Otherwise, keep local chain.

Simplified model: no newcomers, full participation (will be strengthened later).
Ouroboros – Protocol Design

Chain Selection Rule:

*Adopt a valid new chain if it is longer and does not fork by more than k blocks* from local chain.

Otherwise, keep local chain.

Protection against long-range attacks (to be discussed later).

Simplified model: no newcomers, full participation (will be strengthened later).
In each round:

1.) Determine the current longest valid chain.
2.) Determine Slot-Leadership
3.) Slot leader: Pack transactions, create and publish block
Ouroboros – Protocol Design

**Epoch Switch:**

1.) New Stake Distribution  
2.) New Epoch Seed
Ouroboros – Protocol Design

**Epoch Switch:**

1. **New Stake Distribution** → As reported by transactions
2. **New Epoch Seed**

---

**Diagram:**

- **G**
- Random seed
- Epoch 1
- Epoch 2
Ouroboros – Protocol Design

Epoch Switch:
1.) New Stake Distribution
2.) New Epoch Seed
   A slightly more complex process
Ouroboros – Protocol Design

Epoch Switch:

1.) New Stake Distribution

2.) New Epoch Seed
   A slightly more complex process

A secure implementation (MPC) that achieves a randomness beacon.
   - PVSS, messages packed into blocks
Ouroboros – Protocol Design

Epoch Switch:
1.) New Stake Distribution
2.) New Epoch Seed
   A slightly more complex process

Randomness-Beacon Functionality:
- Emits a random value at start of epoch
- Cannot be predicted ahead of time and not tampered
Ouroboros – Protocol Design Summary

Random seed + G
Ouroboros – Protocol Design Summary

Random seed

G

Randomness Beacon
Ouroboros – Protocol Design Summary

Note:
Majority of honest stake required from each new distribution.
Ouroboros – Protocol Design Summary

- Random seed
- Randomness Beacon

G

- Random seed
- Randomness Beacon
Ouroboros – Protocol Design Summary

Random seed + \( G \) + Random seed + Random seed

Randomness Beacon

Randomness Beacon
Ouroboros – Protocol Design Summary

Random seed + Random seed + Random seed

Randomness Beacon

Randomness Beacon
Ouroboros – Analysis

- Analysis of first epoch
- Lifting to multiple epochs (inductive argument)
Ouroboros – Analysis of First Epoch

Life is not perfect... and some forks will emerge...

We need a careful analysis!
A General Analytical Approach: The Forkable String Analysis

Slots are assigned a symbol from an alphabet. The symbol signifies whether honest parties speak, adversaries speak or no-one speaks.

011111111101111111111101100010101010101
A General Analytical Approach: The Forkable String Analysis

Slots are assigned a symbol from an alphabet. The symbol signifies whether honest parties speak, adversaries speak or no-one speaks.

\[
\begin{array}{cccccccccccc}
0 \& 1 \& 1 \& 1 \& 1 \& 1 \& 1 \& 0 \& 1 \& 1 \& 1 \& 1 \& 0 \& 1 \& 0 \& 0 \& 1 \& 0 \& 1 \& 1 \& 0 \& 1
\end{array}
\]

Such a string gives rise to a family of admissible graphs that describe all that can happen in an execution that follows longest chain:
A General Analytical Approach: The Forkable String Analysis

Slots are assigned a symbol from an alphabet. The symbol signifies whether honest parties speak, adversaries speak or no-one speaks.

\[
\begin{array}{cccccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Such a string gives rise to a family of admissible graphs that describe all that can happen in an execution that follows longest chain:

The analysis reveals that the vast majority of strings (under proper conditions) have admissible graphs that translate to well behaved protocol executions.
Forks: Abstracting Protocol Executions

Characteristic string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition
\(\perp\): Slot cannot be claimed (e.g. if election process would assign no leader)
Forks: Abstracting Protocol Executions

Character string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition
\(\perp\): Slot cannot be claimed (e.g. if election process would assign no leader)
Forks: Abstracting Protocol Executions

Characteristic string:

0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition
63: Slot cannot be claimed (e.g., if election process would assign no leader)

Genesis

W = 0 1 1 0 1 0 0 0 1 1 1 0

honest party produces block 1
adversary serves block 3 to honest party
Forks: Abstracting Protocol Executions

Characteristic string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition.
\(\perp\): Slot cannot be claimed (e.g. if process would assign no leader)

adversary serves block 6 to honest party
Forks: Abstracting Protocol Executions

Characteristic string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition.
\(\perp\): Slot cannot be claimed (e.g. if process would assign no leader).

adversary serves block 1 to honest party \((\Delta=3)\)
Forks: Abstracting Protocol Executions

Characteristic string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition.
\(\perp\): Slot cannot be claimed (e.g. if process would assign no leader).

Fork with delay \(\Delta=3\)

adversary serves block 1 to honest party (\(\Delta=3\))
Characteristic string:
0: Slot belongs to exactly one honest party.
1: Slot belongs to a malicious coalition
\(\downarrow\): Slot cannot be claimed (e.g. if process would assign no leader)

adversary serves block 12 to honest party
Forks: Abstracting Protocol Executions

Important Property:
Depth of honest nodes increases (from left to right) if more than $\Delta$ slots apart. (Lower bound on the depth of the fork)
Combinatorics of Characteristic Strings

- Given a characteristic string can we classify the family of 
  forks that it permits?
  - **Characteristic string** is drawn **according to a specific probability 
    distribution: bias toward 0** (by honest-majority assumption).
  - **Forkable string**: those strings that allow a fork with two 
    tines of length equal to the height of the fork.

```
<table>
<thead>
<tr>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

```
0 -> 2 -> 3 -> 6
1 -> 4 -> 5
```
Given a characteristic string can we classify the family of forks that it permits?

- The characteristic string is drawn according to a specific probability distribution.
- A forkable string: those strings that allow a fork with two tines of length equal to the height of the fork.

Focus on this particular structure:

- Analysis shows that this is **a very unlikely structure** to occur (as a function of the length of the sampled string).
- Note: Also **unlikely as a subgraph of any execution**, i.e., no execution has such a bad divergence point (and thus we have CP).
At the core of the analysis lies a 1D Random Walk

\[ \beta \]  

(Probability the adversary finds a POW)

\[ \gamma \approx \alpha - \alpha^2 \]  

“Uniquely successful rounds”  

(\( \alpha = \) Probability an honest party finds a POW)

**Difference:**

1.) #PoWs of adversary in time segment

2.) #PoWs of honest parties in time segment
A favorable step is downwards.

Such a step is more likely by assumption.

Drawing from Bitcoin analysis

At the core of the analysis lies a 1D Random Walk

\[ \beta \]

(Probability the adversary finds a POW)

\[ \gamma \approx \alpha - \alpha^2 \]

“Uniquely successful rounds”

(\(\alpha\) = Probability an honest party finds a POW)

1.) #PoWs of adversary in time segment

2.) #PoWs of honest parties in time segment

- A favorable step is downwards.
- Such a step is more likely by assumption \(\gamma > \beta\).
from PoW to PoS

- Winning a slot for the honest parties (even uniquely) does not necessarily constitute a favorable step in the random walk.

“Nothing-at-stake”:
The adversary may reuse an opportunity to issue a block in multiple paths of a fork.
Forkable Strings

(for simplicity we do only the \{0,1\} case)

For a time \( t \), the following quantities are of interest to the adversary:

- \( \text{gap}(t) \): length difference with leading honest node.
- \( \text{reserve}(t) \): number of adversarial slots after end of \( t \).
- \( \text{reach}(t) := \text{reserve}(t) - \text{gap}(t) \).
**Forkable Strings**

For a time $t$, the following quantities are of interest to the adversary:

- **$\text{gap}(t)$**: length difference with leading honest node.
- **$\text{reserve}(t)$**: number of adversarial slots after end of $t$.
- **$\text{reach}(t) := \text{reserve}(t) - \text{gap}(t)$**.

![Diagram illustrating forkable strings](image_url)
For a time $t$, the following quantities are of interest to the adversary:

- **$\text{gap}(t)$**: length difference with leading honest node.
- **$\text{reserve}(t)$**: number of adversarial slots after end of $t$.
- **$\text{reach}(t)$**: $\text{reserve}(t) - \text{gap}(t)$.

\[
\begin{align*}
\text{reserve}(t) & = 3 \\
\end{align*}
\]
For a time $t$, the following quantities are of interest to the adversary:

- $\text{gap}(t)$: length difference with leading honest node.
- $\text{reserve}(t)$: number of adversarial slots after end of $t$.
- $\text{reach}(t) := \text{reserve}(t) - \text{gap}(t)$.
Forkable Strings

Can the adversary catch up with “longest chain” with this time?

- \( \text{reach}(t) \): number of adversarial slots after end of \( t \).
- \( \text{reach}(t) := \text{reserve}(t) - \text{gap}(t) \).

reach\( (t) = -1 \)
Forkable Strings

Looking at a fork $F$ in general, we are interested in:

- $\text{reach}(F)$: max $\text{reach}(t)$
- $\text{margin}(F)$: second highest & disjoint $\text{reach}(t')$

![Diagram of fork F with reach(t) = -1, t' with reach(t') = -1, and t with reach(t) = 0]
Forkable Strings

Looking at a fork $F$ in general, we are interested in:

- **reach($F$)**: max reach($t$)
- **margin($F$)**: second highest & disjoint reach($t'$)

$$
\text{reach}(F) = 0 \\
\text{reach}(t) = 0 \\
\text{reach}(t') = -1 \\
\text{margin}(F) = -1
$$
Looking at a fork $F$ in general, we are interested in:

- $\text{reach}(F)$: max $\text{reach}(t)$
- $\text{margin}(F)$: second highest & disjoint $\text{reach}(t')$

If second highest allows to catch up with longest: margin non-negative.
Define:

\[
\rho(w) = \max_F \text{reach}(F) \\
\mu(w) = \max_F \text{margin}(F)
\]

- Fact: \( w \) is forkable (adversary wins) iff \( \mu(w) \geq 0 \).

- We want to prove that the density of forkable strings among all strings is tiny (assuming Hamming weight is below 1/2).

- We consider a 2D random walk defined by the pair \( (\rho(w), \mu(w)) \) where \( w \) is a binomial random variable.
Recursive Formula for Reach & Margin

\[
[\rho(w1), \mu(w1)] = [\rho(w) + 1, \mu(w) + 1]
\]

\[
[\rho(w0), \mu(w0)] = \begin{cases} 
[\rho(w) - 1, 0] & \rho(w) > \mu(w) \geq 0 \\
[0, \mu(w) - 1] & \rho(w) = 0 \\
[\rho(w) - 1, \mu(w) - 1] & \text{otherwise}
\end{cases}
\]
Recursive Formula for Reach & Margin

\[ [\rho(w1), \mu(w1)] = [\rho(w) + 1, \mu(w) + 1] \]

\[ [\rho(w0), \mu(w0)] = \begin{cases} 
[\rho(w) - 1, 0] & \rho(w) > \mu(w) = 0 \\
[0, \mu(w) - 1] & \rho(w) = 0 \\
[\rho(w) - 1, \mu(w) - 1] & \text{otherwise}
\end{cases} \]

reach and margin decrement
Recursive Formula for Reach & Margin

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

\[
[\rho(w_0), \mu(w_0)] = \begin{cases} 
[\rho(w) - 1, 0] & \rho(w) > \mu(w) = 0 \\
[0, \mu(w) - 1] & \rho(w) = 0 \\
[\rho(w) - 1, \mu(w) - 1] & \text{otherwise}
\end{cases}
\]
Recursive Formula for Reach & Margin

\[
[\rho(w1), \mu(w1)] = [\rho(w) + 1, \mu(w) + 1]
\]

\[
[\rho(w0), \mu(w0)] = \begin{cases} 
[\rho(w) - 1, 0] & \text{if } \rho(w) > \mu(w) = 0 \\
[0, \mu(w) - 1] & \rho(w) = 0 \\
[\rho(w) - 1, \mu(w) - 1] & \text{otherwise}
\end{cases}
\]

- It is possible for the adversary to compensate for the margin, by sacrificing reach.
- Reach never drops below 0.
- Reach and margin decrement.
2D Random Walk

$\gamma \approx \alpha - \alpha^2$

probability

$\alpha$

an honest party wins a slot

$\beta$

probability

the adversary wins a slot
2D Random Walk

margin

reach

\[ \gamma \approx \alpha - \alpha^2 \]

\[ \alpha \quad \text{probability an honest party wins a slot} \]

\[ \beta \quad \text{probability the adversary wins a slot} \]
2D Random Walk

margin

reach

\[ \gamma \approx \alpha - \alpha^2 \]

\( \alpha \)  probability an honest party wins a slot

\( \beta \)  probability the adversary wins a slot
2D Random Walk

probability

\[ \gamma \approx \alpha - \alpha^2 \]

\[ \alpha \] an honest party wins a slot

\[ \beta \] probability the adversary wins a slot
2D Random Walk

probability an honest party wins a slot
\( \alpha \)

probability the adversary wins a slot
\( \beta \)

\( \gamma \approx \alpha - \alpha^2 \)
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

**Hot** We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).

**Volatile** We let \( \text{Vol}_t \) denote the event that \( -\delta \sqrt{n} \leq M(t) \leq R(t) < \delta \sqrt{n} \).

**Cold** We let \( \text{Cold}_t \) denote the event that \( M(t) < -\delta \sqrt{n} \).

**Analysis shows:**

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] &\geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] &\geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] &\leq 2^{-\Omega(\sqrt{n})}.
\end{align*}
\]
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

- **Hot**: We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).
- **Volatile**: We let \( \text{Vol}_t \) denote the event that \( -\delta \sqrt{n} < R(t) < \delta \sqrt{n} \).
- **Cold**: We let \( \text{Cold}_t \) denote the event that \( R(t) < -\delta \sqrt{n} \) or \( M(t) > \delta \sqrt{n} \).

\[ R(t), M(t) : \]
Reach resp. margin after \( t \sqrt{n} \) steps of the random walk ("coarse grained steps of the walk").

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} | \text{Cold}_t] & \geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} | \text{Vol}_t] & \geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} | \text{Vol}_t] & \leq 2^{-\Omega(\sqrt{n})}.
\end{align*}
\]
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

**Hot** We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).

**Volatile** We let \( \text{Vol}_t \) denote the event that \( -\delta \sqrt{n} \leq M(t) \leq R(t) < \delta \sqrt{n} \).

**Cold** We let \( \text{Cold}_t \) denote the event that \( M(t) < -\delta \sqrt{n} \).

**Analysis shows:**

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] &\geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] &\geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] &\leq 2^{-\Omega(\sqrt{n})}.
\end{align*}
\]
Analysis

\[ R_t = \rho (w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu (w_1 \ldots w_t). \]

**Hot** We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).

**Volatile** We let \( \text{Vol}_t \) denote the event that \( -\delta \sqrt{n} \leq M(t) \leq R(t) < \delta \sqrt{n} \).

**Cold** We let \( \text{Cold}_t \) denote the event that \( M(t) < -\delta \sqrt{n} \).

**Analysis shows:**

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] & \geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] & \geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] & \leq 2^{-\Omega(\sqrt{n})}. 
\end{align*}
\]
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

Hot We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).

Volatile We let \( \text{Vol}_t \) denote the event that \( -\delta \sqrt{n} \leq M(t) \leq R(t) < \delta \sqrt{n} \).

Cold We let \( \text{Cold}_t \) denote the event that \( M(t) < -\delta \sqrt{n} \).

**Analysis shows:**

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] & \geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] & \geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] & \leq 2^{-\Omega(\sqrt{n})}.
\end{align*}
\]
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

We let \( \text{Hot}_t \) denote the event that \( R(t) \geq \delta \sqrt{n} \) and \( M(t) \geq -\delta \sqrt{n} \).

\[ \sqrt{n} \leq M(t) \leq R(t) < \delta \sqrt{n}. \]

\[ \sqrt{n} < -\delta \sqrt{n}. \]

An improved analysis shows an error bound of

\[ 2^{-\Omega(n)} \]

"The Combinatorics of the Longest-Chain Rule: Linear Consistency for Proof-of-Stake Blockchains" by Erica Blum and Aggelos Kiayias and Cristopher Moore and Saad Quader and Alexander Russell.
Analysis

\[ R_t = \rho(w_1 \ldots w_t) \quad \text{and} \quad M_t = \mu(w_1 \ldots w_t). \]

Conclusion:
- Characteristic string not forkable (w.h.p.)
  \( \rightarrow \) No long diverging paths
- Common prefix achieved (w.h.p.)

Analysis shows:

\[
\begin{align*}
\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] & \geq 1 - 2^{-\Omega(\sqrt{n})}, \\
\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] & \geq \Omega(\epsilon), \\
\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] & \leq 2^{-\Omega(\sqrt{n})}.
\end{align*}
\]
Also: Chain Growth & Quality

**Chain Quality**: any sufficiently long section along a (viable) tine must contain an honest node with overwhelming probability. 
(→ Otherwise, #0’s < #1’s)

**Chain Growth**: The #0’s support growth and by the above, the growth is reflected in any viable tine (with a small discount).
Also: Chain Growth & Quality

**Chain Quality**: any sufficiently long section along a (viable) tine must contain an honest node with overwhelming probability. 
(→ Otherwise, #0’s < #1’s)

**Chain Growth**: The #0’s support growth and by the above, the growth is reflected in any viable tine (with a small discount).

Viable tines: Correspond to chains that are long enough to be adopted by an honest party at a given time.

\[
w = \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
\end{array}
\]
Lifting To Multiple Epochs
Lifting To Multiple Epochs

Random seed +

Randomness Beacon
Lifting To Multiple Epochs

Properties CP, CG, CQ

Random seed

Randomness Beacon
Lifting To Multiple Epochs

- **CG+CQ**: All honest parties report a **green block**.
- **CP**: Agreement on **green block**.

**Random seed**

**Randomness Beacon**
Lifting To Multiple Epochs

- **CG+CQ**: All honest parties report a green block.
- **CP**: Agreement on green block.

**Epoch Randomness**: Has to be released after fixing the stake distribution.
Lifting To Multiple Epochs

“Smooth Epoch Boundaries”
Distribution of entire characteristic string is uniquely defined for this execution and dominated by a binomial distribution favoring 0’s over 1’s (as before).
Incentives

How to make **honest parties participate**?

- **Costs**
  - Such as verifying transactions, packaging them in the right order.
- **Rewards**
  - Such as collecting fees.

**Problem**: Pure chain quality underrepresents the honest parties’ effort: Effort in maintaining the inputs is not rewarded proportionally.
Incentives

How to make honest parties participate?

- **Costs**
  - Such as verifying transactions, packaging them in the right order.
- **Rewards**
  - Such as collecting fees.

**Key Idea:**
Main effort is related to input contribution $\rightarrow$ Declare it to be a separate task.
Incentives

Solution: Input Endorsers

• Each slot elects an additional stakeholder (or a set of stakeholders) to contribute inputs.
  • Using a parallel lottery.
  • Like the 2-for-1 mechanism in PoW as in GKL analysis or Fruitchains.

• Endorsed inputs are permitted in the blockchain any time within a small window following and inclusive the slot that elects them.
Incentives

Solution: Input Endorsers

- **Each slot elects an additional stakeholder** (or a set of stakeholders) to contribute inputs.
  - Using a parallel lottery.
  - Like the 2-for-1 mechanism in PoW as in GKL analysis or Fruitchains.

- Endorsed inputs are permitted in the blockchain any time within a small window following and inclusive the slot that elects them.

→ Protocol becomes a Nash equilibrium for an appropriate reward function (that rewards *input blocks in an aggregate fashion over a sequence of blocks*).

→ Overall #Input blocks proportional to stake.
Ouroboros BFT

An extremely simple BFT protocol that follows from the Ouroboros Classic analysis

Ouroboros BFT

Ouroboros Classic (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Ouroboros Crypsinous (S&P 2019)

Ouroboros with Privacy
**Ouroboros BFT**

Central Observation:
A characteristic string (assume binary) with Hamming-weight less than 1/3 is not forkable.
Ouroboros – Praos & Genesis

- **Ouroboros “Classic”** (Crypto 17)
- **Ouroboros Praos** (Eurocrypt 2018)
- **Ouroboros Genesis** (CCS 2018)
- **Ouroboros Chronos** (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay ("semi-synchronous")
+ Full dynamic availability
+ Bootstrapping from Genesis
+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
Ouroboros – Praos & Genesis

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

- Adaptive Adversaries
- Network delay (“semi-synchronous”)
- Full dynamic availability
- Bootstrapping from Genesis
- Only based on same-speed assumption.
- Bootstrapping state and time from genesis

Stronger cryptographic primitives needed:
- To enable private lottery
- To fully mitigate adaptive corruptions

= PoS blockchain in the DA setting without global clocks.
Ouroboros – Praos & Genesis

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay ("semi-synchronous")

+ Full dynamic availability
+ Bootstrapping from Genesis

Newcomers should be able to join the system without the extra help of existing parties.
Ouroboros – Praos & Genesis

- **Ouroboros “Classic”** (Crypto 17)
- **Ouroboros Praos** (Eurocrypt 2018)
- **Ouroboros Genesis** (CCS 2018)
- **Ouroboros Chronos** (In submission, 2019)

**Semi-adaptive adversaries, synchrony**
- Strong mathematical framework

**+ Adaptive Adversaries**
**+ Network Delay ("semi-synchronous")**

**+ Full dynamic availability**
**+ Bootstrapping from Genesis**

**= PoS blockchain in the DA setting without global clocks.**
Cryptographic Tools to Protect against Adaptive Adversaries
Cryptographic Tools to Protect against Adaptive Adversaries

Verifiable random functions (VRF) – with unpredictability under malicious key generation.

\[(VRF.\text{Gen}, VRF.\text{Eval}, VRF.\text{Verify})\]

- Output appears pseudo-random (for a new input)
- Input and output are verifiably tied together
- Output cannot be biased by crafting strange keys
- Purpose: \textbf{Allow private leader-election} and thereby a more realistic attacker model (mitigate adaptive attacks).
Key evolving signature scheme (KES)

\[(KES.\text{Gen}, KES.\text{Sign}, KES.\text{Update}, KES.\text{Verify})\]

\[\rightarrow\text{Operates as normal signature scheme (unforgeable)}\]
\[\rightarrow\text{Key updates: All values signed “in the past” remain unforgeable even if party gets corrupted after the update.}\]
\[\rightarrow\text{Purpose: Protect previous actions and thereby allow realistic corruption model (tolerate “immediate corruptions”).}\]
Ouroboros Praos/Genesis: Basic Operation

Epoch - 2  Epoch - 1  Current epoch
Ouroboros Praos/Genesis: Basic Operation

- In each slot, each party evaluates slot-leadership.
  Private election, proportional to stake, including recent randomness from the chain

- A slot leader extends a chain by creating the block for this slot.
Ouroboros Praos/Genesis: Basic Operation
Ouroboros Praos/Genesis: Basic Operation

Epoch - 2  Epoch - 1  Current epoch
1) Agreement on **stake distribution**
Ouroboros Praos/Genesis: Basic Operation

Simple Randomness-Beacon Implementation:
Current seed := Hash of verifiably random values from the chain.

\[ VRF.Eval_{sk_i}("NONCE", \text{previous seed}, \text{slot}) \]

1) Agreement on stake distribution.
2) Agreement on randomness.
3) Randomness affected by honest block(s)
Ouroboros Praos/Genesis: Basic Operation

**Lottery in each slot:**

A party $i$ is leader if and only if $\text{VRF. Eval}_{sk_i}(\text{"TEST"}, \text{seed}, \text{slot}) < T(\text{stake}_i)$

- Empty slots possible
- Multiple leaders possible
- Leadership proof from VRF.

1) Agreement on **stake distribution**.
2) Agreement on **randomness**.
3) **Randomness** affected by honest block(s)
Ouroboros Praos/Genesis: Details on Leader Election

\[ VRF.\text{Eval}_{sk_i}("TEST", \text{seed}, \text{slot}) < T(stake_i) \]

\[ \Rightarrow T(stake_i) = 2^{\ell_{VRF}} \varphi_f(\text{rel. stake}_i) \]

\[ \Rightarrow \varphi_f(x) = 1 - (1 - f)^x \]
Ouroboros Praos/Genesis: Details on Leader Election

\[ VRF.\text{Eval}_{sk_i}("TEST", \text{seed}, \text{slot}) < T(\text{stake}_i) \]

\[ \rightarrow T(\text{stake}_i) = 2^{\ell_{VRF}} \varphi_f(\text{rel. stake}_i) \]

\[ \rightarrow \varphi_f(x) = 1 - (1 - f)^x \]

Some remarks:

1.) Active slot coefficient: \( \varphi_f(1) = f \); slot empty with prob. \( 1 - f \).

2.) Independent aggregation property: \( 1 - \varphi_f \left( \sum_i x_i \right) = \prod_i (1 - \varphi_f(x_i)) \)

\( \rightarrow \) Probability of leadership independent of distribution to addresses.

\( \rightarrow \) The concave (and subadditive as \( \varphi_f(0) = 0 \)) property eases the analysis.
Recall: Chain-Selection Rule
Recall: Chain-Selection Rule
Recall: Chain-Selection Rule
Recall: Chain-Selection Rule
Attention: Longest Chain Rule Does not Work

Local Family of long-range attacks (e.g., stake-bleeding [GKR18])
Recall: Chain-Selection Rule

Chain Selection Rule [e.g., DGKR18] :

Adopt a valid new chain...

1) …if it is longer and does not fork by more than k blocks from local chain.

Otherwise, keep local chain.
At first sight...

… it seems one would require one of the following:

1.) Online parties **maintain a moving checkpoint**
   → Joining parties need advice.

2.) A **fixed and known lower bound** on **participation**
   → No flexible participation, protocol might be stalled.
Ouroboros – Praos & Genesis

Ouroboros “Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
The Genesis Chain-Selection Rule

... it seems one would require one of the following:

1.) Online parties maintain a moving checkpoint → Joining parties need advice

2.) A fixed and known lower bound on participation → No flexible participation, protocol might be stalled

We do not require either of these!

Thanks to a more involved Chain-Selection Rule
The Genesis Chain-Selection Rule
The Genesis Chain-Selection Rule
The Genesis Chain-Selection Rule

Time-Interval after fork

(INT (Protocol parameter: size of INT))
The Genesis Chain-Selection Rule

Genesis Rule:

**Adopt** a valid new chain…

1) …if it is longer and does not fork by more than k blocks from local chain.

2) … or **if it forks by more than k blocks but has higher block density** on interval INT.

Otherwise, keep local chain.
Ouroboros – Praos & Genesis

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay ("semi-synchronous")

+ Full dynamic availability
+ Bootstrapping from Genesis

= PoS blockchain in the DA setting without global clocks.

Genesis Security Proof
Roadmap of Security Proof of Genesis

- Security of Ouroboros Genesis with the old chain selection rule (=Praos) and *dynamic participation* but no newly joining parties.

- Security for joining parties: *new Genesis rule in action.*
Roadmap of Security Proof of Genesis

- Security of Ouroboros Genesis with the old chain selection rule (=Praos) and dynamic participation but no newly joining parties.

- Security for joining parties: new Genesis rule in action.
Security under dynamic Participation

- Recall the Fork abstraction:
Security under dynamic Participation

- Recall the Fork abstraction:

A substantial **CP-violation (divergence)** occurs only with negligible probability if:

**Majority of active stake is honest.**
Security under dynamic Participation

- Recall the Fork abstraction:

A substantial **CP-violation (divergence)** occurs only with negligible probability if:

\[ \alpha \cdot (1 - f)^{A+1} \geq (1 + \epsilon)/2 \]
Security under dynamic Participation

- Dynamic Participation: Dependent variables, biased lottery in favor of honest parties $\rightarrow$ Martingales
- We show **Common Prefix, Chain Growth, Chain Quality**
- Realizes the ledger (composable analysis)
Roadmap of Security Proof of Genesis

- Security of Ouroboros Genesis with the old chain selection rule (=Praos) and dynamic participation but no newly joining parties.

- Security for joining parties: new Genesis rule in action.
Security of the Genesis Rule

Claim 1:

If a party is always up-to-date and using the Genesis chain-selection rule, she will never adopt a chain that forks by more than k blocks (compared to her local chain in any round).

Claim 2:

Using the Genesis chain-selection rule, a newly joining party will adopt a recent chain with large common prefix w.r.t. honest parties. No other advice than the genesis block is needed.
Security of the Genesis Rule

Claim 1:
If a party is always up-to-date and using the Genesis chain-selection rule, she will never adopt a chain that forks by more than k blocks (compared to her local chain in any round).

Claim 2:
Using the Genesis chain-selection rule, a newly joining party will adopt a recent chain with large common prefix w.r.t. honest parties. No other advice than the genesis block is needed.
Claim 1 – Proof Idea

Interval INT
Claim 1 – Proof Idea

Interval INT
Claim 1 – Proof Idea

Interval INT

First honest slot after INT
Claim 1 – Proof Idea

A substantial divergence! Hence, situation does not occur.

Covered by previous “Praos” analysis.
Security of the Genesis Rule

Claim 1:
If a party is always up-to-date and using the Genesis chain-selection rule, she will never adopt a chain that forks by more than $k$ blocks (compared to her local chain in any round).

Claim 2:
Using the Genesis chain-selection rule, a newly joining party will adopt a recent chain with large common prefix w.r.t. honest parties. No other advice than the genesis block is needed.
Claim 2 – Proof Idea

First newly joining party

\[ t_0 \]
Claim 2 – Proof Idea

First newly joining party

$t_0$
Claim 2 – Proof Idea

If 🖤 decides to adopt a “good” chain C, then so does 🌿.
Claim 2 – Proof Idea

G

C

$\mathbf{t}_0$
Claim 2 – Proof Idea
Claim 2 – Proof Idea

A substantial divergence does not occur.

Covered by previous analysis of new chain-selection rule
Privacy in Ouroboros: Crypsinous

- Ouroboros "Classic" (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros Chronos (In submission, 2019)

Ouroboros BFT

Ouroboros Crypsinous (S&P 2019)

Genesis with Privacy
Privacy in Ouroboros: Crypsinous

**Problem Summary:**

Public verifiability of leader schedule vs. Hide amount of stake possessed

- Ouroboros Genesis (CCS 2018)
- Ouroboros Crypsinous (S&P 2019) Genesis with Privacy
- Ouroboros Chronos (In submission, 2019)
Privacy in Ouroboros: Crypsinous

Ouroboros "Classic" (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Ouroboros BFT

Ouroboros Crypsinous (S&P 2019)

Genesis with Privacy
- Zero-knowledge proof systems
- SNARKs
Ouroboros: Real-World Implementations

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros BFT

Cardano is running on Ouroboros PoS and other companies are implementing versions of it.
Ouroboros – Chronos

- Ouroboros “Classic” (Crypto 17)
- Ouroboros Praos (Eurocrypt 2018)
- Ouroboros Genesis (CCS 2018)
- Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (”semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
So far: Dependency on a Good Timing Service

Time axis:

Clock Functionality
So far: Dependency on a Good Timing Service

Slot Axis:

Time axis:
So far: Dependency on a Good Timing Service

Slot Axis:

Time axis:
So far: Dependency on a Good Timing Service

Slot Axis:

Recall Lottery (Nakamoto-Style Consensus):
- A party $i$ is leader if and only if $VRF_{sk_i} (TEST, seed, slot) < T(stake_i)$
- All have the same idea of future and past
  - Example: “future chains are rejected - because bad anyway”.

Time axis:
Strong Assumption:
- Perfect time coordination for everyone, including newly joining parties.
- Needs another protocol eventually, e.g. NTP.
Better: Same-Speed Assumption

Slot Axis:

Time axis:
Coping with Imperfect Coordination

Slot Axis:

Time axis:

NOW
Genesis: Situation not that bad…

If we manage to keep all honest parties somewhat close we’re kind of good.
Genesis: Situation not that bad...

If we manage to keep all honest parties somewhat close we’re kind of good.

- Close together: Honest parties’ timestamps are never more than $\Delta$ apart (order of network delay).
- Small adjustments needed to Ouroboros Genesis to deal with future chains
Genesis: Situation not that bad…

If we manage to keep all honest parties somewhat close we’re kind of good.

- **Close together**: Honest parties’ timestamps are never more than $\Delta$ apart (order of network delay).
- **Small adjustments** needed to Ouroboros Genesis to deal with future chains.

$\rightarrow$ Same-speed: Initial parties do stay close.

$\rightarrow$ **Joining parties** have a harder life…
Genesis: Situation not that bad...
Genesis: Situation not that bad…

Joining party:
Genesis: Situation not that bad…

Joining party:

Genesis chain selection rule:
- **Good prefix** is the **densest prefix**
- **Genesis rule prefers** densest prefix
Genesis: Situation not that bad...

Joining party:

- Good prefix is the densest prefix
- Genesis rule prefers densest prefix

No reliable local time:
- No cut-off possible
- No reliable ledger state
The Synchronization Problem

• Joining parties: Need to **bootstrap a good timestamp**
  • **Only source of information:** network traffic and genesis block.
  • **Good:** Within the Δ-interval of existing honest parties.
  • From before: Good timestamp → Good state.

• Bootstrapping **under the same assumptions.**
  • Same-speed, honest majority, diffusion network, RO
  • The dynamic availability setting (similar to the Bitcoin setting for fixed difficulty).
The Synchronization Problem

- Joining parties: Need to **bootstrap a good timestamp**
  - **Only source of information**: network traffic and genesis block.
  - **Good**: Within the $\Delta$-interval of existing honest parties.
  - From before: Good timestamp $\rightarrow$ Good state.

- Bootstrapping **under the same assumptions**.
  - Same-speed, honest majority, diffusion network, RO
  - The dynamic availability setting (similar to the Bitcoin setting for fixed difficulty).

This is what **Chronos** achieves
Chronos Overview
Chronos Overview

- **Alert parties:** broadcast *time-beacons* and leave *evidence* of beacons in the blockchain.
Chronos Overview

- Alert parties: broadcast time-beacons and leave evidence of beacons in the blockchain.

- They perform local-clock adjustments based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.
Chronos Overview

- **Alert parties:** broadcast *time-beacons* and leave *evidence* of beacons in the blockchain.

- They perform *local-clock adjustments* based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and *retrace the evidence*.
  - Perform the very same clock adjustments to compute a good timestamp
Chronos Overview

- **Alert parties**: broadcast **time-beacons** and leave **evidence** of beacons in the blockchain.

- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties**: Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence**.
  - Perform the very same clock adjustments to compute a good timestamp
Addition to the "Timing Lottery" in the first part of the epoch:

- IF $VRF_{sk_i}(\text"SYNC", seed, slot) < T(stake_i)$ THEN
  - Broadcast Sync-Beacon:
  - Normal slot leaders pack transactions + beacons.
Chronos Overview

- **Alert parties**: broadcast *time-beacons* and leave evidence of beacons in the blockchain.

- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties**: Once hooked up on a prefix of the densest chain, record beacons and *retrace the evidence*.
  - Perform the very same clock adjustments to compute a good timestamp
Chronos: Synchronization Procedure

- **Throughout the epochs:** Alice records the arrival times of valid beacons (filter out duplicates, invalid ones etc.)
- **At the end of each epoch:** Compute local clock-adjustment.
**Chronos: Synchronization Procedure**

- **Throughout the epochs:** Alice records the arrival times of valid beacons (filter out duplicates, invalid ones etc.)
- **At the end of each epoch:** Compute local clock-adjustment.

- **At the end of epoch:** for each recorded beacon, do:
  
  \[ \text{RECOM} := \text{slot} - \text{ARRIVALTIME} \]
Adjustment rule:

- At the end of epoch: add the median of recommendations to local time:
Example

\[ s_{l_{\text{bob}}} = T - y \]

\[ s_{l} := T - x \]

\[ s_{l_{\text{alice}}} = T - z \]
Example

- $s_{l_{\text{bob}}} = T - y$
- $s_{l} := T - x$
- $s_{l_{\text{alice}}} = T - z$

$T \rightarrow T + \delta \rightarrow T + \delta'$
\[(T-z + r) + (T-x) - (T-z + \delta) = r + (T-x) - \delta\]
Example

\[ sI_{bob} = T - y \]

\[ sI := T - x \]

\[ sI_{alice} = T - z \]

\[ r' + (T - x) - \delta' \]

\[ r' + (T - x) - \delta \]
Properties of the Synchronization Procedure

Local Clocks are $\Delta$-close!
(because $|\delta' - \delta| \leq \Delta$)

\[
r' + (T-x) - \delta
\]
Properties of the Synchronization Procedure

Furthermore, by honest-majority assumption: → Median, i.e., \textbf{adjustment is bounded}.
Chronos Overview

- **Alert parties**: broadcast time-beacons and leave evidence of beacons in the blockchain.

- They perform local-clock adjustments based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties**: Once hooked up on a prefix of the densest chain, record beacons and retrace the evidence.
  - Perform the very same clock adjustments to compute a good timestamp
Chronos: Joining Procedure
Chronos: Joining Procedure
Chronos: Joining Procedure
Chronos: Joining Procedure

- Densest chain wins, good prefix.
Chronos: Joining Procedure
Chronos: Joining Procedure
Chronos: Joining Procedure
Chronos: Joining Procedure

**Required Beacon Properties:**
- **Fresh information:** Only generated after becoming online.
- **Validated and filtered w.r.t. fresh lottery.**
- Contained in common prefix.
- **Bootstrapping the local clock is possible thanks to**
  - Agreement on evidence
  - Freshness of beacons: reasoning as before to get $\Delta$-close
- **Bootstrapping the local clock is possible thanks to**
  - Agreement on evidence
  - Freshness of beacons: reasoning as before to get $\Delta$-close

- **Clock adjustments of** alert parties **can be retraced**
  - Stop when computed timestamp is before the next sync-slot.
- **Bootstrapping the local clock is possible thanks to:**
  - Agreement on evidence
  - Freshness of beacons: reasoning as before to get $\Delta$-close

- **Clock adjustments of alert parties can be retraced**
  - Stop when computed timestamp is before the next sync-slot.

- **Good time-stamp $\rightarrow$ Good blockchain**
  - Cut-off future blocks and the genesis rule guarantees the rest.
Playing With Ouroboros

Check out the interactive Ouroboros animation:

[ouroboros.iohk.io]
End of the Tutorial – Thank you!

Ouroboros “Classic” (Crypto 17)

Ouroboros Praos (Eurocrypt 2018)

Ouroboros Genesis (CCS 2018)

Ouroboros Chronos (In submission, 2019)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

+ Only based on same-speed assumption.
+ Bootstrapping state and time from genesis

= PoS blockchain in the DA setting without global clocks.
Contact information and credits

Email: christian.badertscher@ed.ac.uk

References:


**Chronos:** C. Badertscher, P. Gaži, A. Kiayias, A. Russell, V. Zikas. Ouroboros Chronos: Permissionless Clock-Synchronization via Proof-of-Stake. ia.cr/2019/838


---


Images: https://openclipart.org/