



Tutorial: The Ouroboros Protocol Family

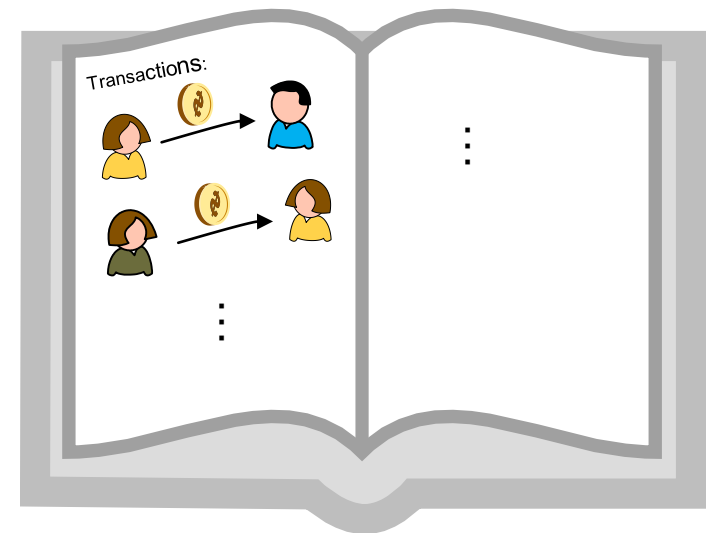
Advances in Financial Technologies, 2019

Christian Badertscher & Aggelos Kiayias

The University of Edinburgh
& IOHK

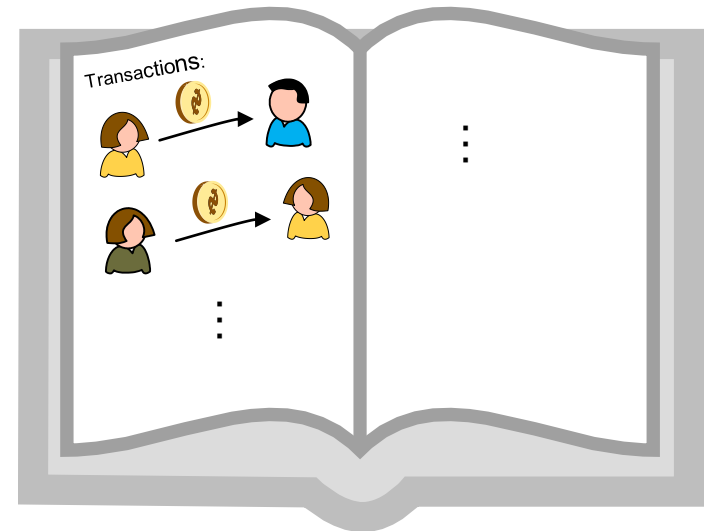
The Goal of Blockchain Protocols

- Implement an immutable transaction ledger...



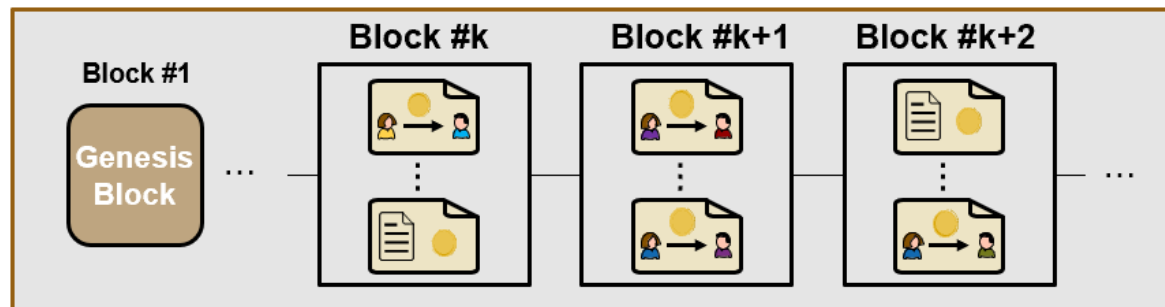
Immutable Ledger Properties

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



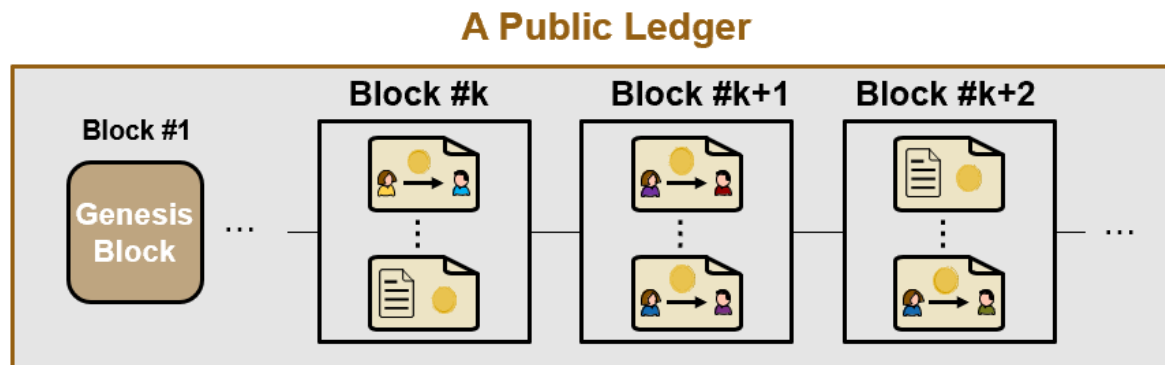
The Ledger Functionality

A Public Ledger



The Ledger Functionality

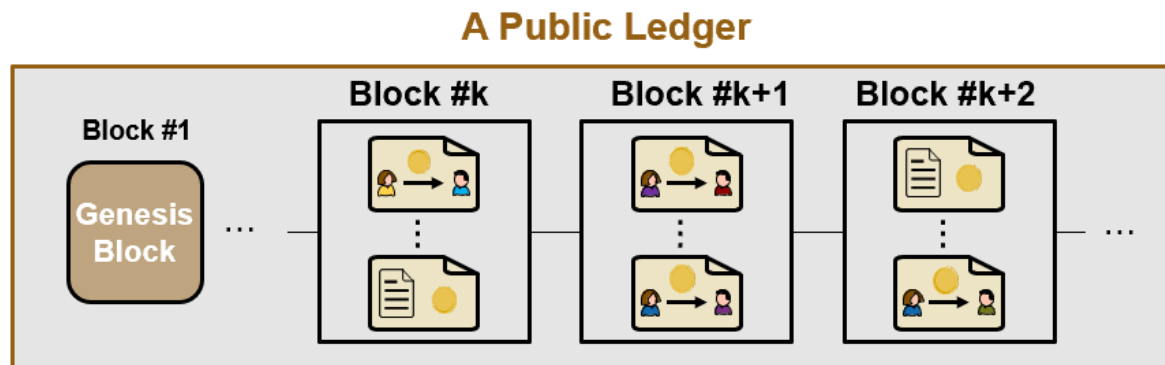
- The functionality formalizes the relevant blockchain properties and the limited capabilities of an adversary.
 - E.g., common state, well-formed blocks, recent transactions etc.



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

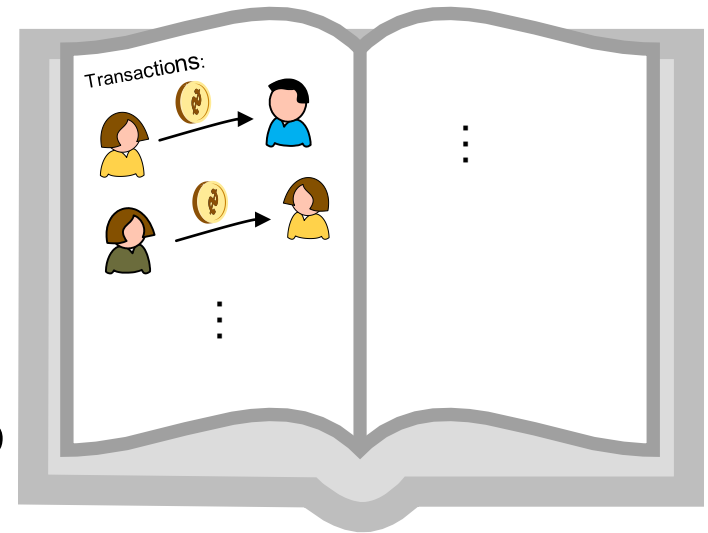


Realizing the Ledger

- 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

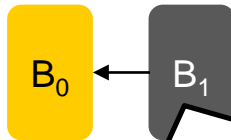
- **The case of Bitcoin:**
 - Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.



Genesis Block

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.



Lottery-style (simplified):

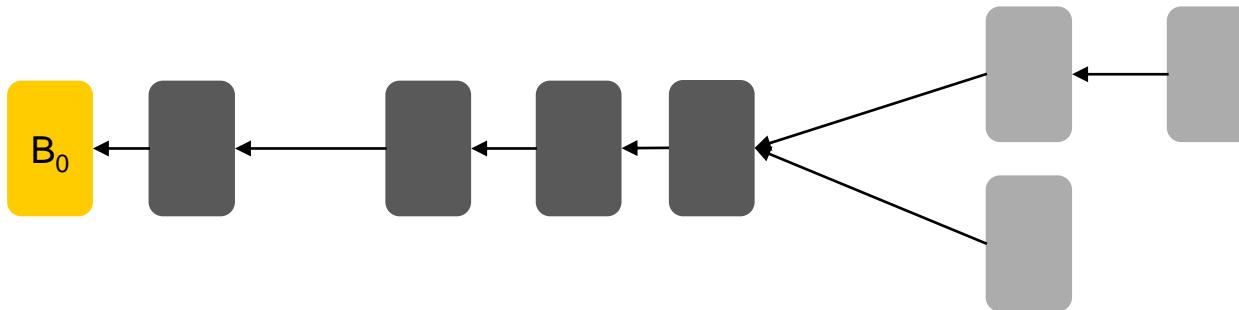
Find nonce N s.t. $\text{Hash}(N, \text{tx} \dots, \text{Hash}(B_{i-1})) < T$

Observation:

More hashing power \rightarrow better chances to produce blocks.

Realizing the Ledger

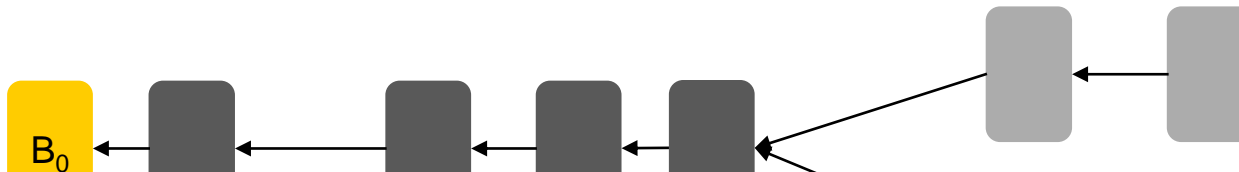
- **The case of Bitcoin:**
 - Parties repeatedly try to solve cryptographic puzzles. A solution allows to create a block and append it to the chain.



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

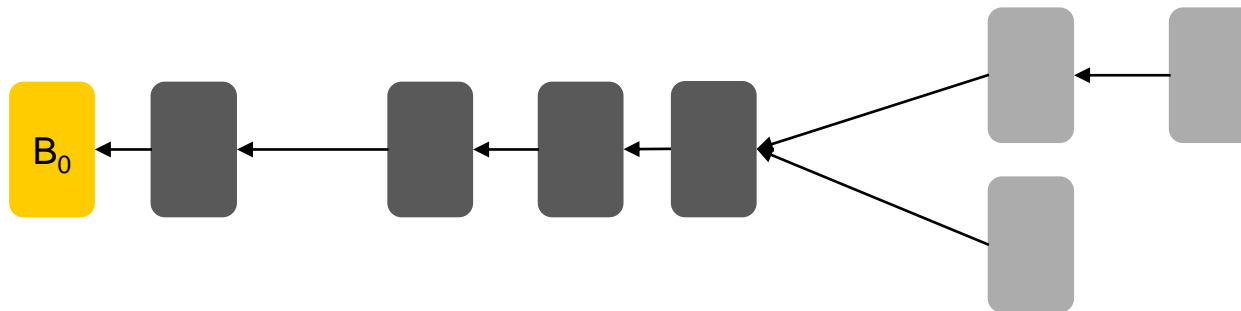
Realizing the Ledger

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.

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.

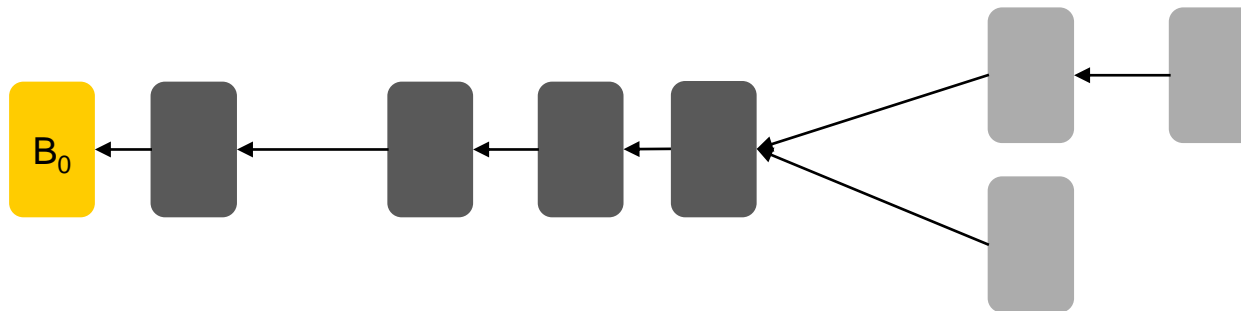


- **Bitcoin is not energy efficient** as the hash-based lottery consumes a lot of energy to ensure the protocol's security.

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.



- **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)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

Ouroboros
Praos
(Eurocrypt 2018)

Ouroboros
Genesis
(CCS 2018)

Ouroboros
Chronos
(In submission, 2019)

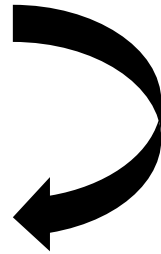
Tutorial Overview – Main Content

Ouroboros
“Classic”
(Crypto 17)

Ouroboros
Praos
(Eurocrypt 2018)

Ouroboros
Genesis
(CCS 2018)

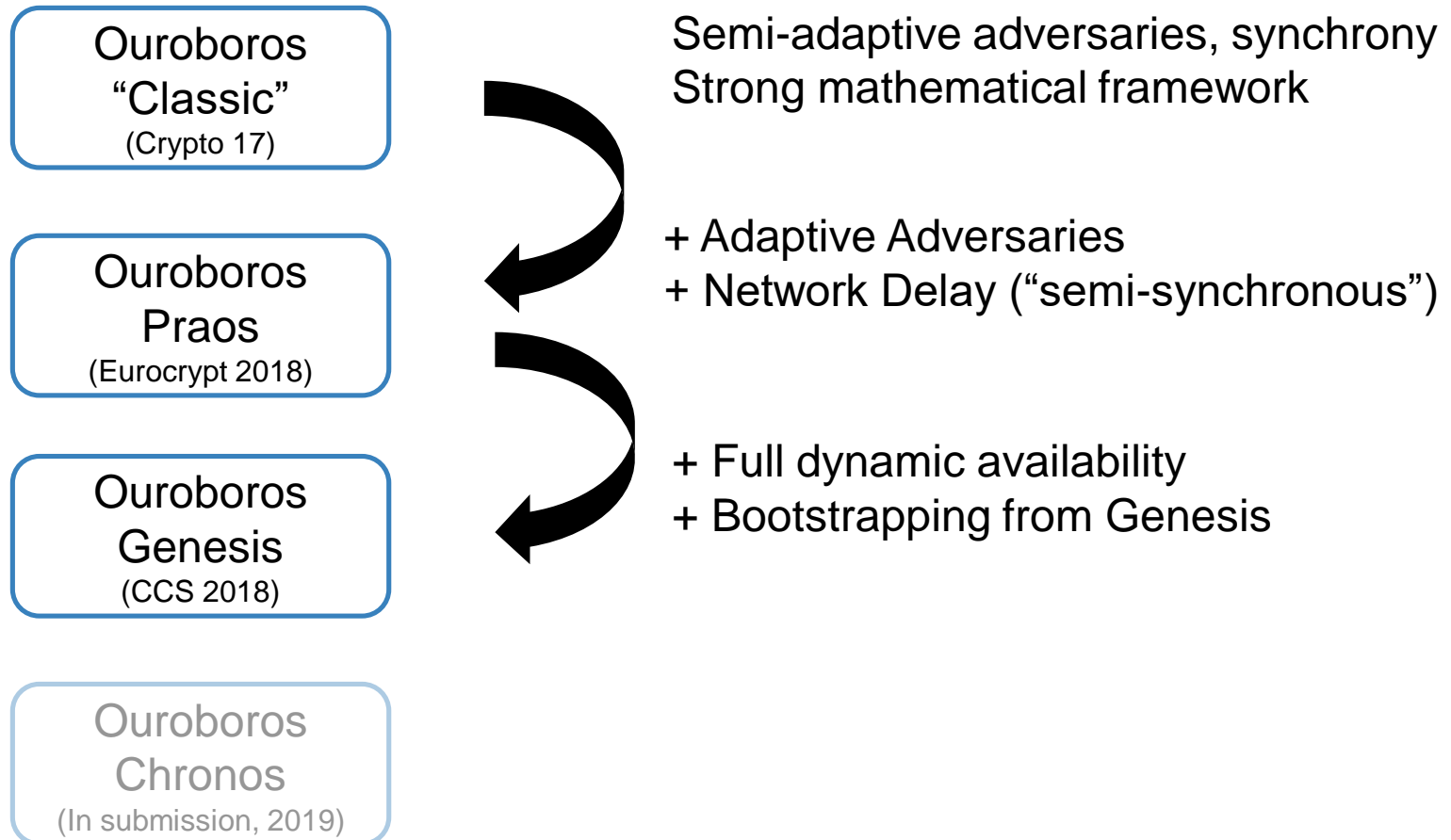
Ouroboros
Chronos
(In submission, 2019)



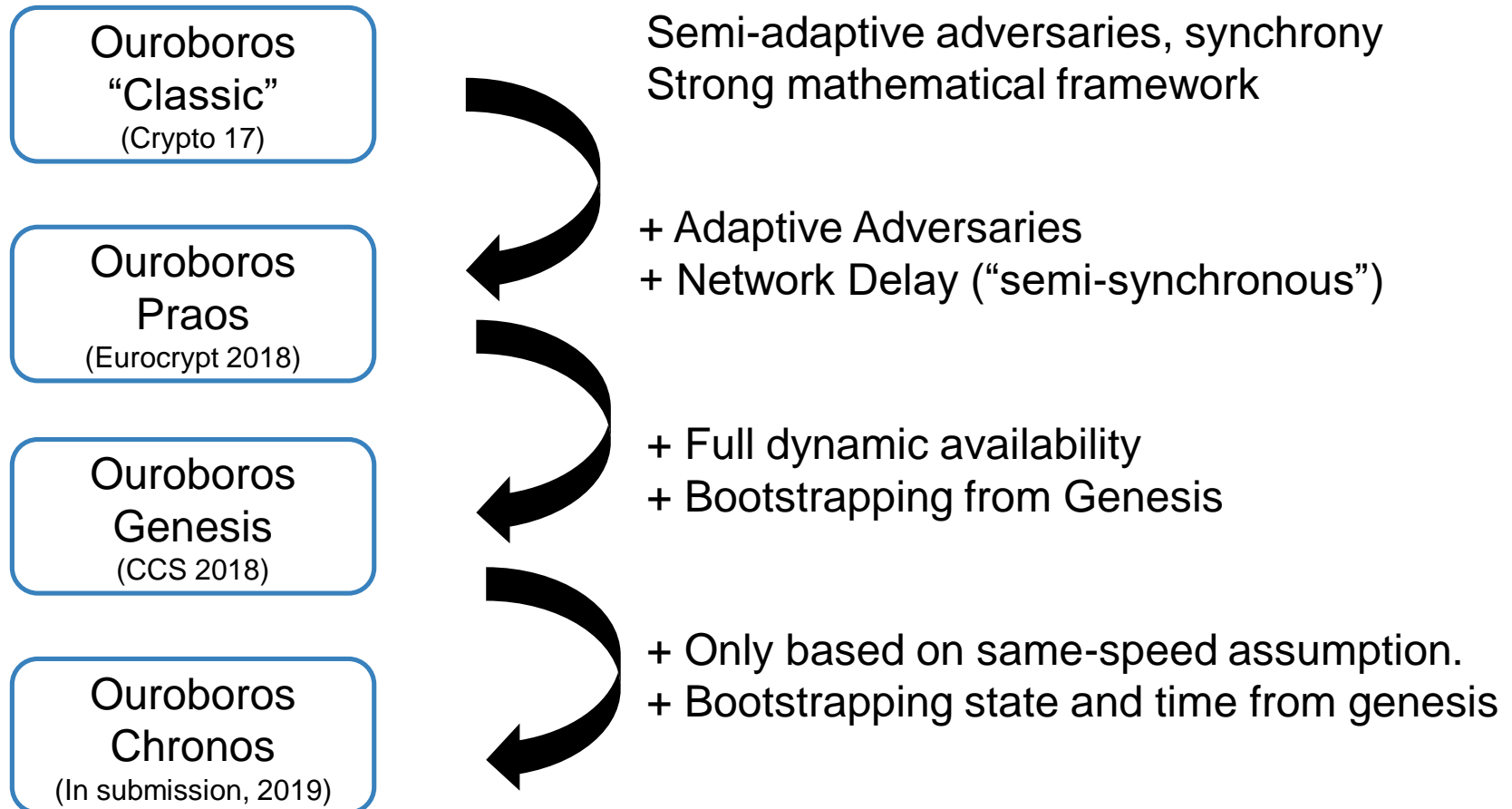
Semi-adaptive adversaries, synchrony
Strong mathematical framework

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

Tutorial Overview – Main Content



Tutorial Overview – Main Content



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

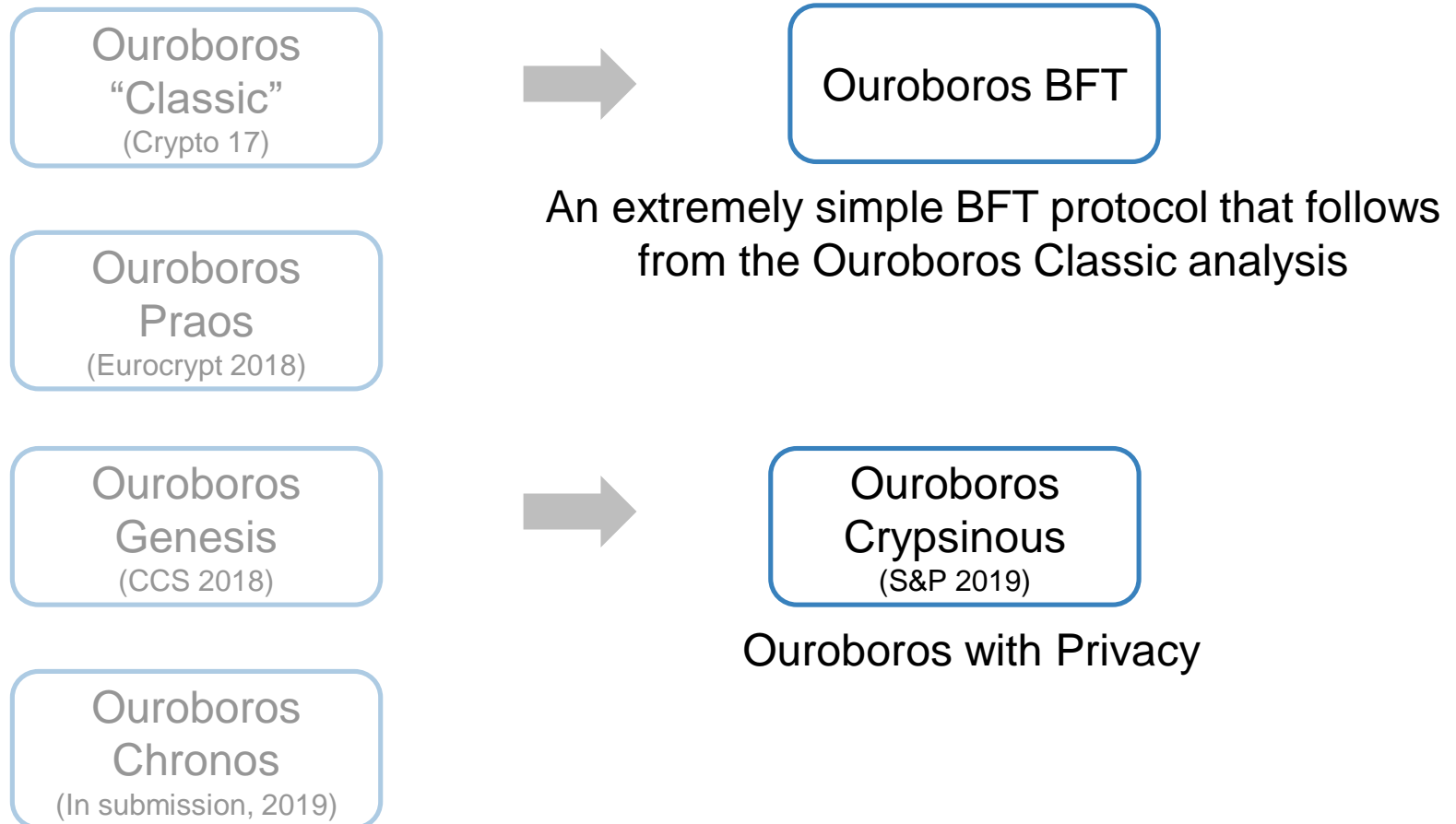
+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ 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 – 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

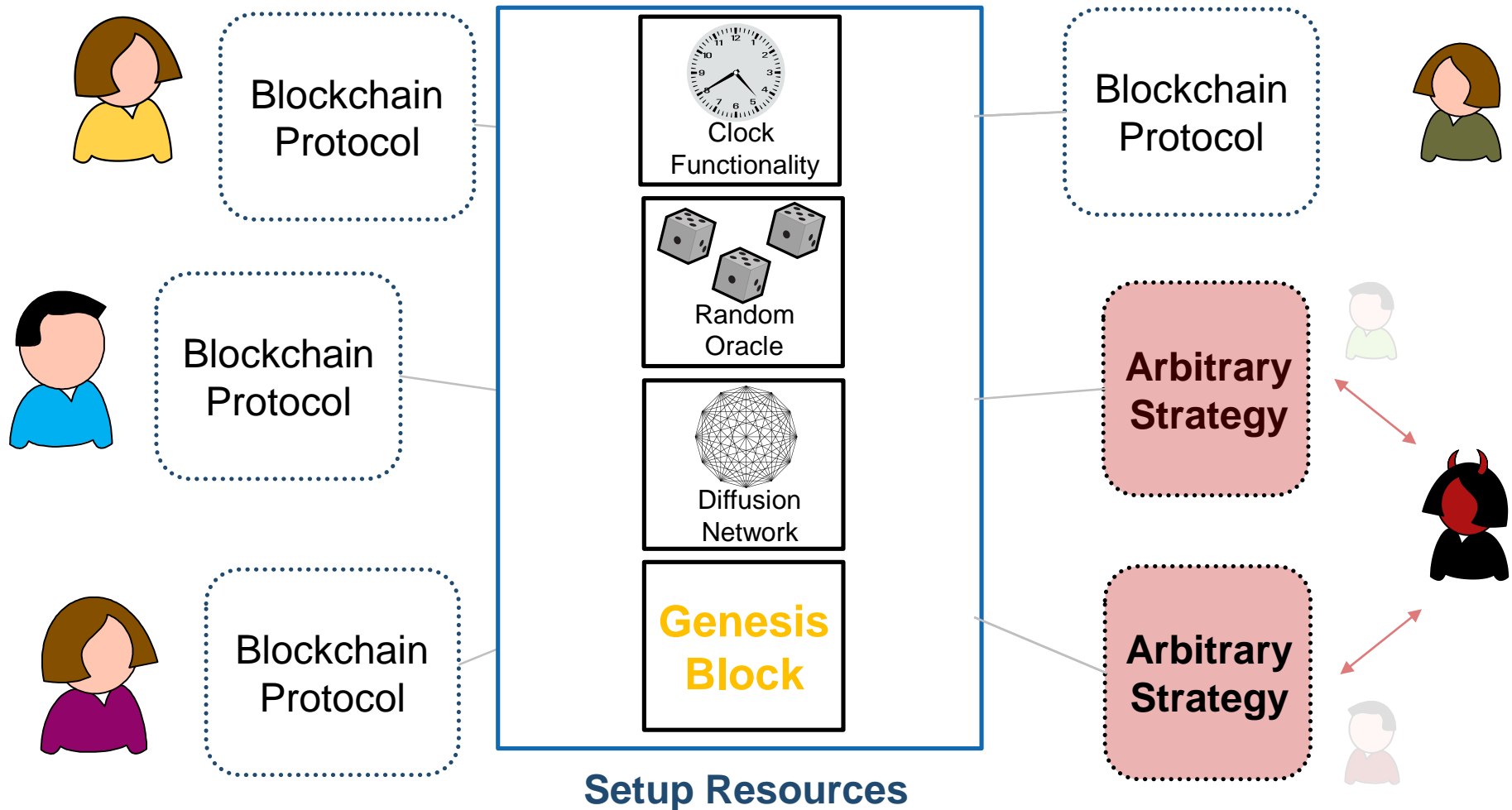
+ Adaptive Adversaries
+ Network Delay (“semi-synchronous”)

+ Full dynamic availability
+ Bootstrapping from Genesis

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

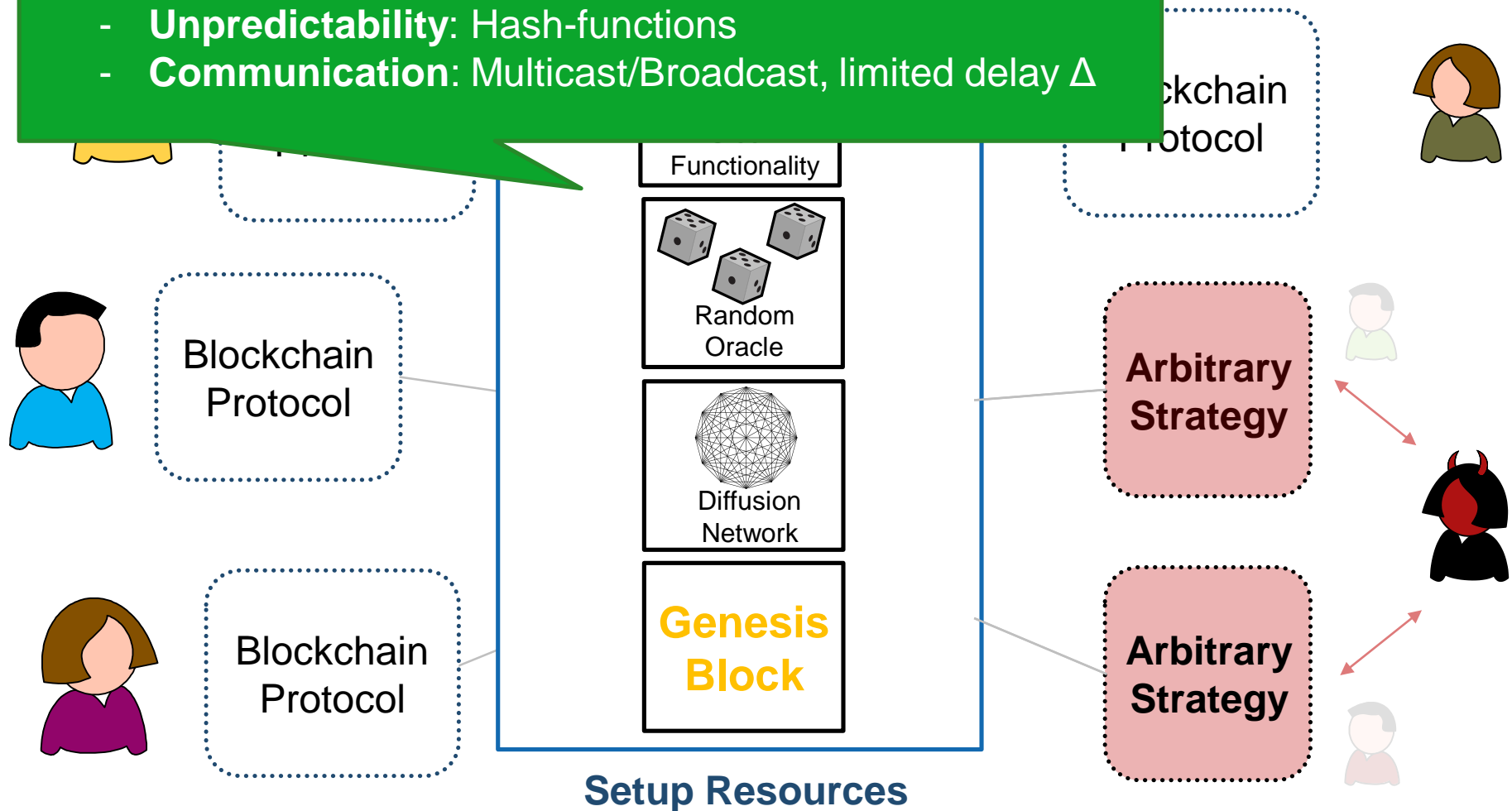
**= PoS blockchain in the DA setting
without global clocks.**

The General Picture and Assumed Resources

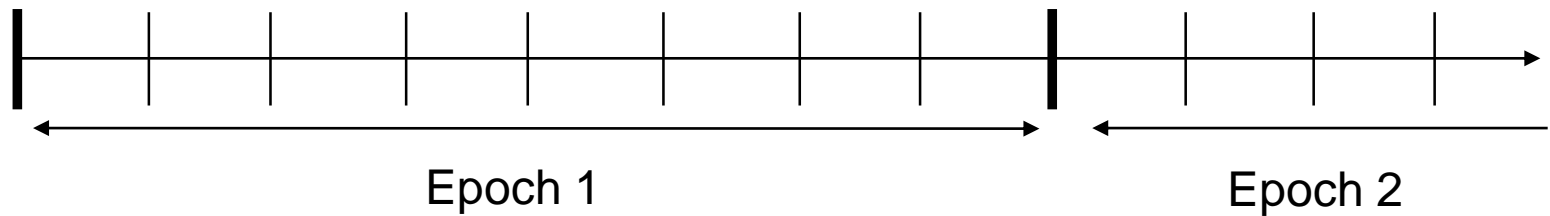


Resources

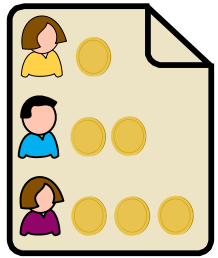
- **Synchrony:** Time-stamps, slots
- **Unpredictability:** Hash-functions
- **Communication:** Multicast/Broadcast, limited delay Δ



Ouroboros – Protocol Design



Ouroboros – Protocol Design





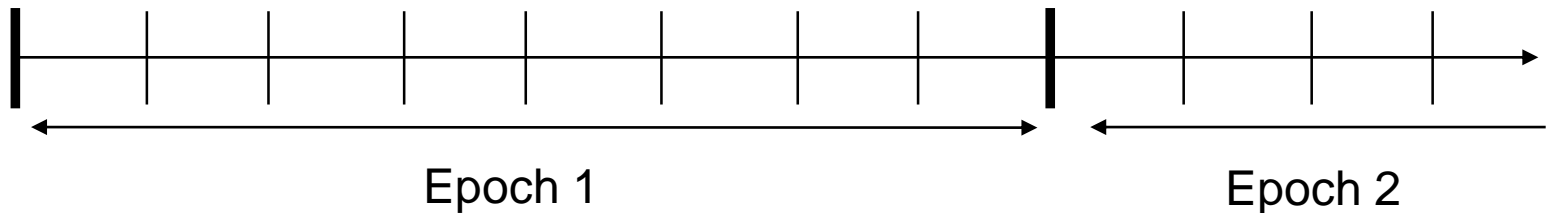
+

Random
seed

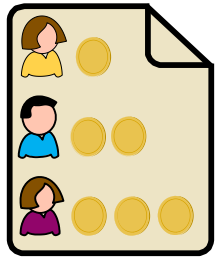


G

 = Public address: verification key vk_j of a signature scheme
 = A number of coins (tokens) s_j associated to vk_j



Ouroboros – Protocol Design



+

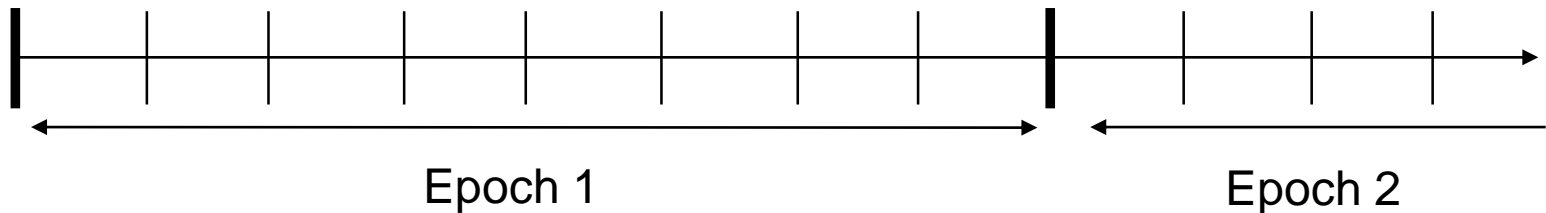
Random
seed



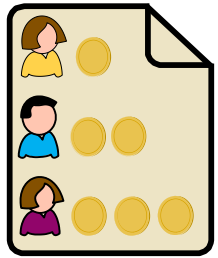
G

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



+

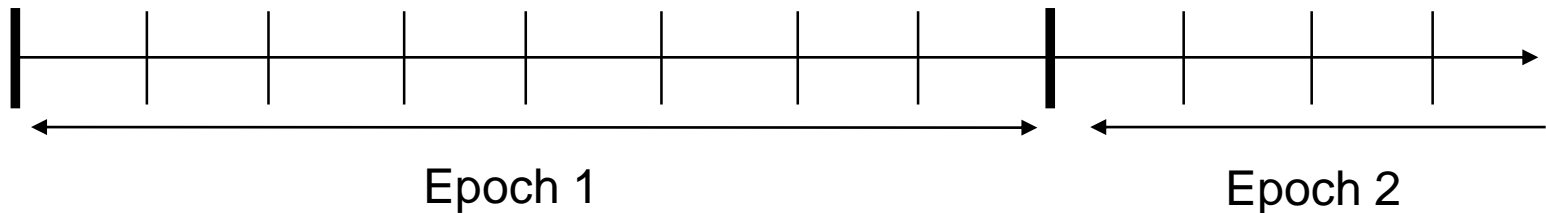
Random
seed



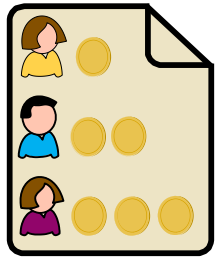
G

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



+

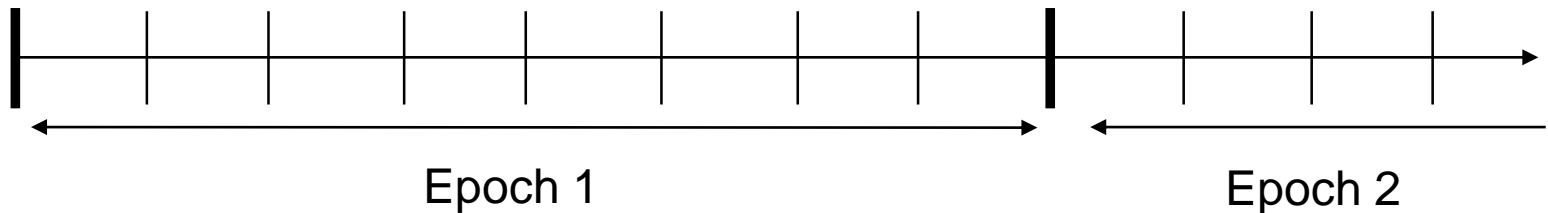
Random
seed



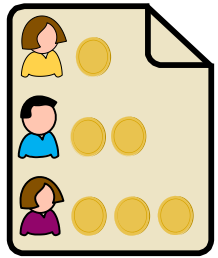
G

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

$$F(\text{person}_1, \text{person}_2, \text{person}_3, \text{seed}, \text{slot}) \rightarrow \text{person}_1 / \text{person}_2 / \text{person}_3$$



Ouroboros – Protocol Design



+

Random
seed



G

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

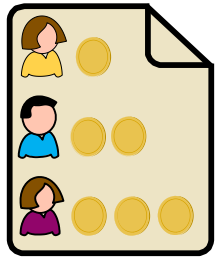
$$F(\text{person 1 coin}, \text{person 2 coin}, \text{person 3 coin}, \text{seed}, \text{slot}) \rightarrow \text{person 1} / \text{person 2} / \text{person 3}$$

For example:

Biased-Coin Toss

Epoch 2

Ouroboros – Protocol Design



+

Random
seed

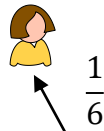


G

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

$$F(\text{person 1}, \text{person 2}, \text{person 3}, \text{seed}, \text{slot}) \rightarrow \text{person 1} / \text{person 2} / \text{person 3}$$

For example:

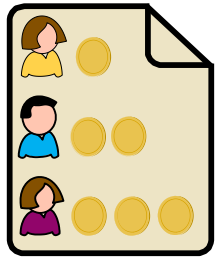


$\frac{1}{6}$

Biased-Coin Toss

Epoch 2

Ouroboros – Protocol Design



+

Random
seed

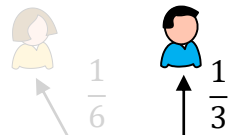


G

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

$$F(\text{person}_1, \text{person}_2, \text{person}_3, \text{seed}, \text{slot}) \rightarrow \text{person}_1 / \text{person}_2 / \text{person}_3$$

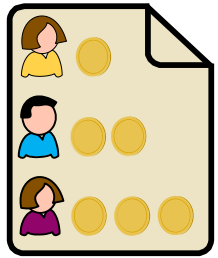
For example:



Biased-Coin Toss

Epoch 2

Ouroboros – Protocol Design



+

Random
seed

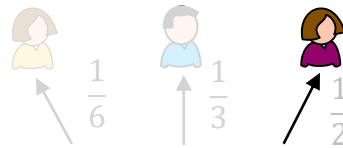


G

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

$$F(\text{person 1}, \text{person 2}, \text{person 3}, \text{seed}, \text{slot}) \rightarrow \text{person 1} / \text{person 2} / \text{person 3}$$

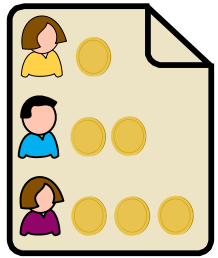
For example:



Biased-Coin Toss

Epoch 2

Ouroboros – Protocol Design



+

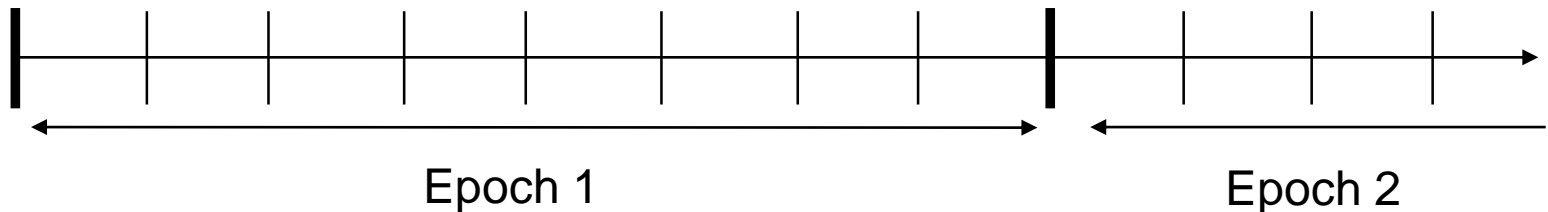
Random
seed



G

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

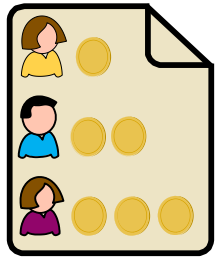
$$F(\text{person}_1, \text{person}_2, \text{person}_3, \text{seed}, \text{slot}) \rightarrow \text{person}_1 / \text{person}_2 / \text{person}_3$$



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



+

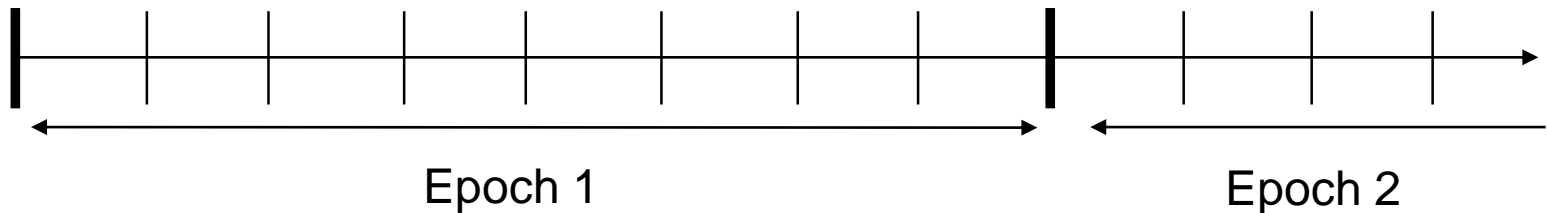
Random
seed



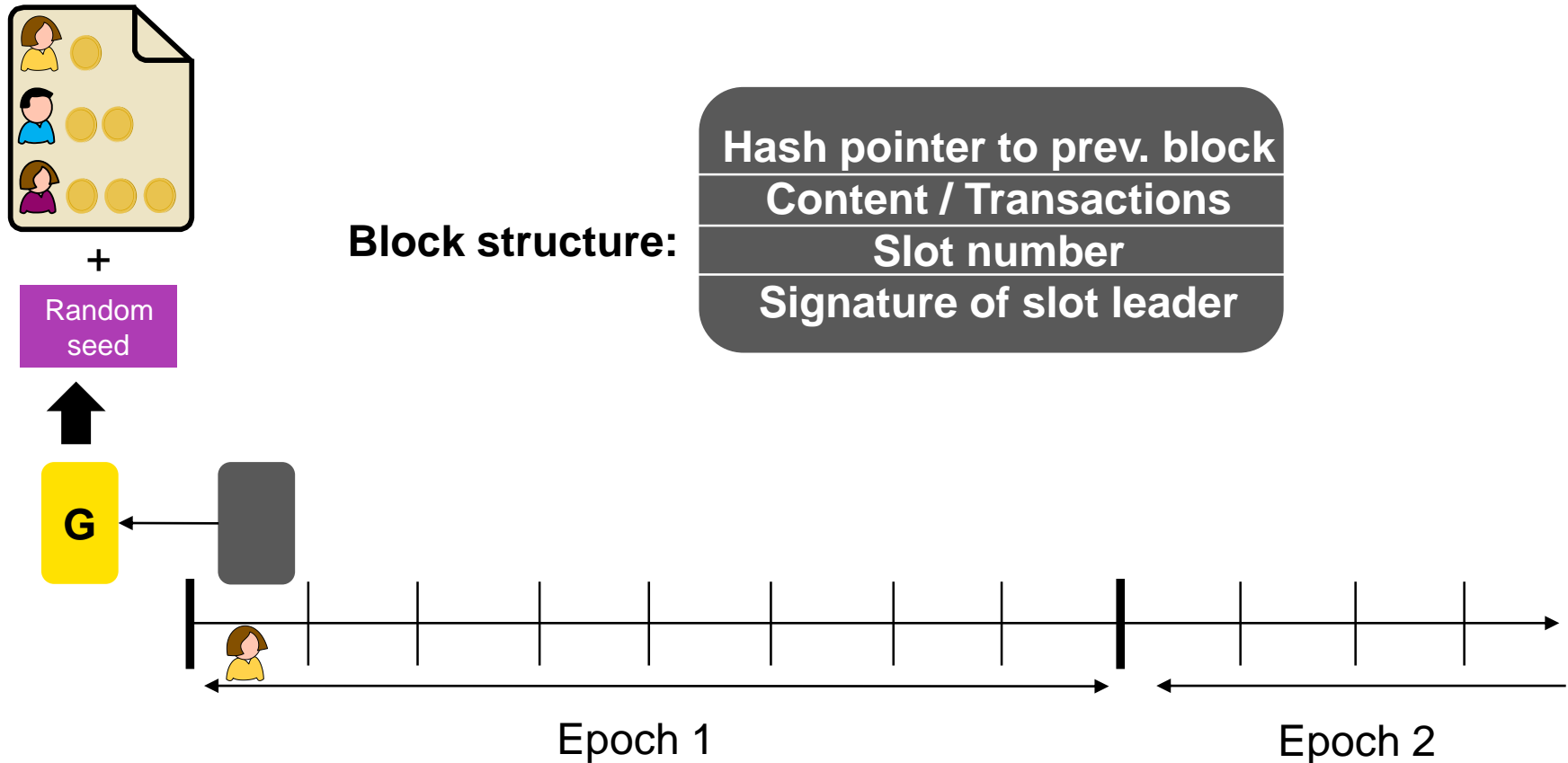
G

In each round:

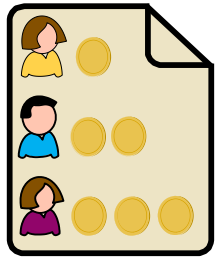
- 1.) Determine the current longest valid chain.
- 2.) Determine Slot-Leadership
- 3.) **Slot leader: Pack transactions, create and publish block**



Ouroboros – Protocol Design



Ouroboros – Protocol Design



+

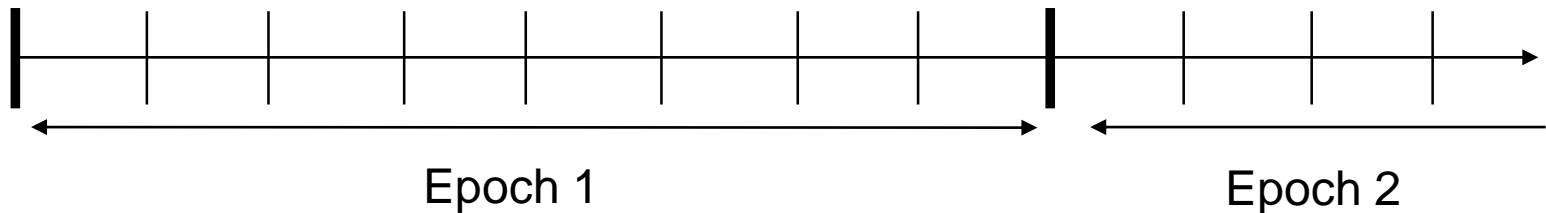
Random
seed



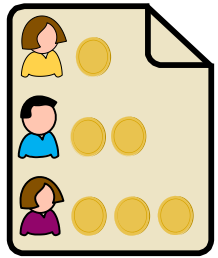
G

In each round:

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

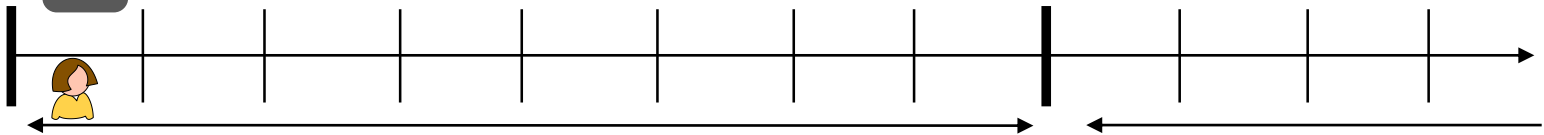


Ouroboros – Protocol Design



+

Random
seed



Epoch 1

Epoch 2

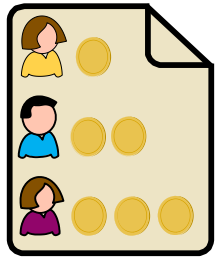
Chain Selection Rule:

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

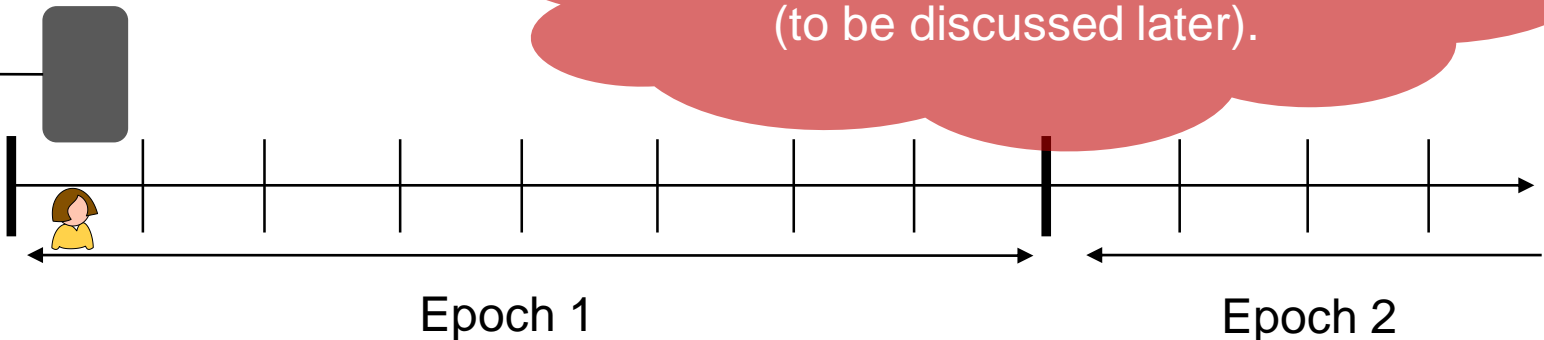


+

Random
seed

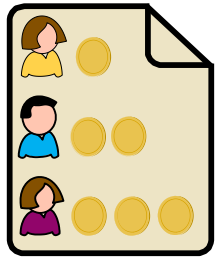


G



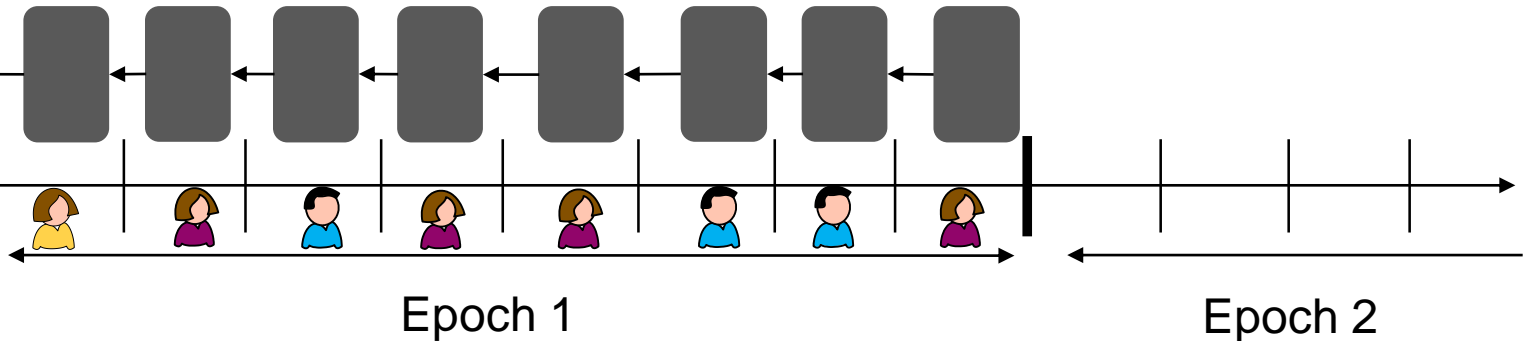
Simplified model: no newcomers, full participation (will be strengthened later).

Ouroboros – Protocol Design

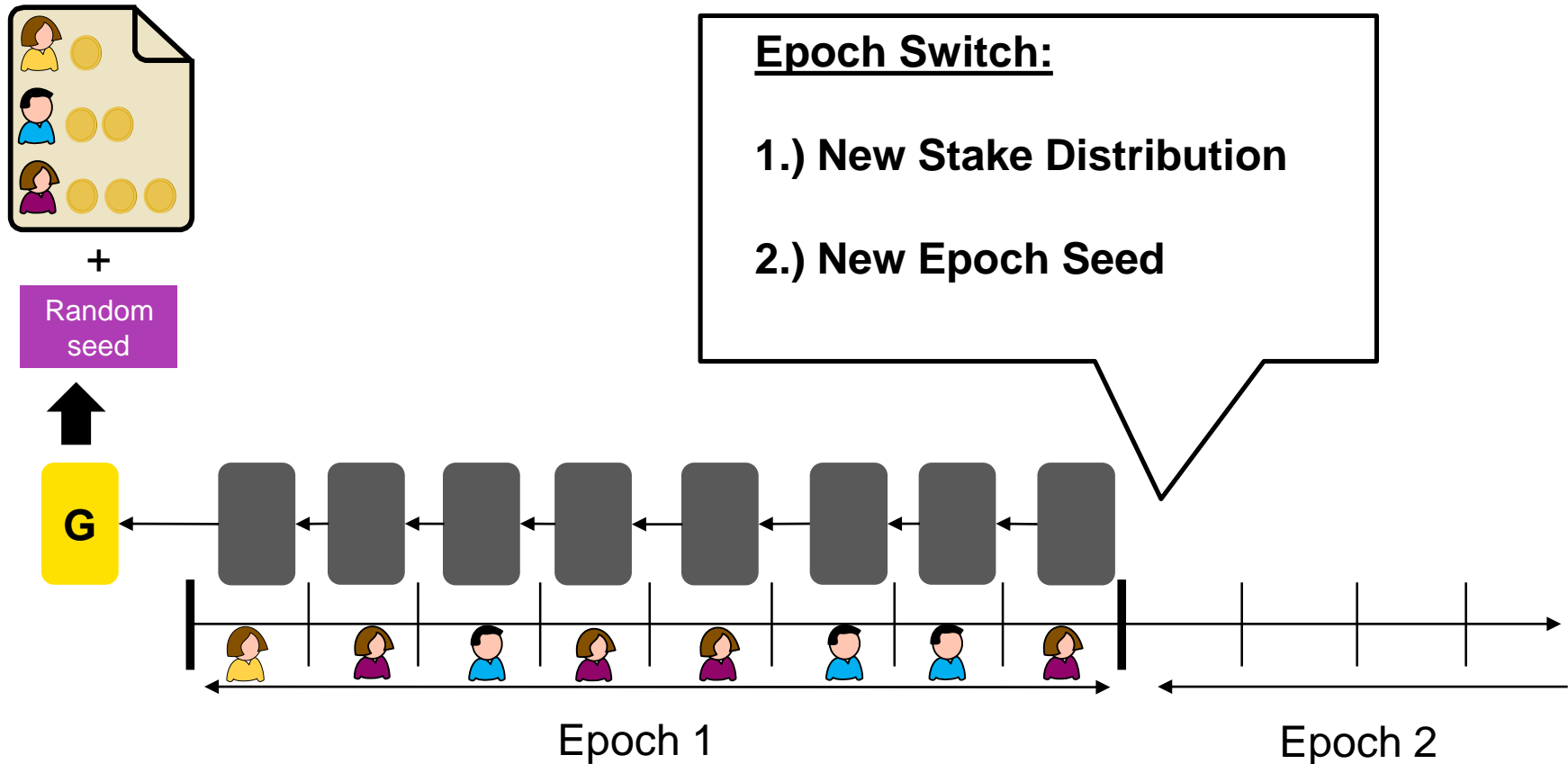


+

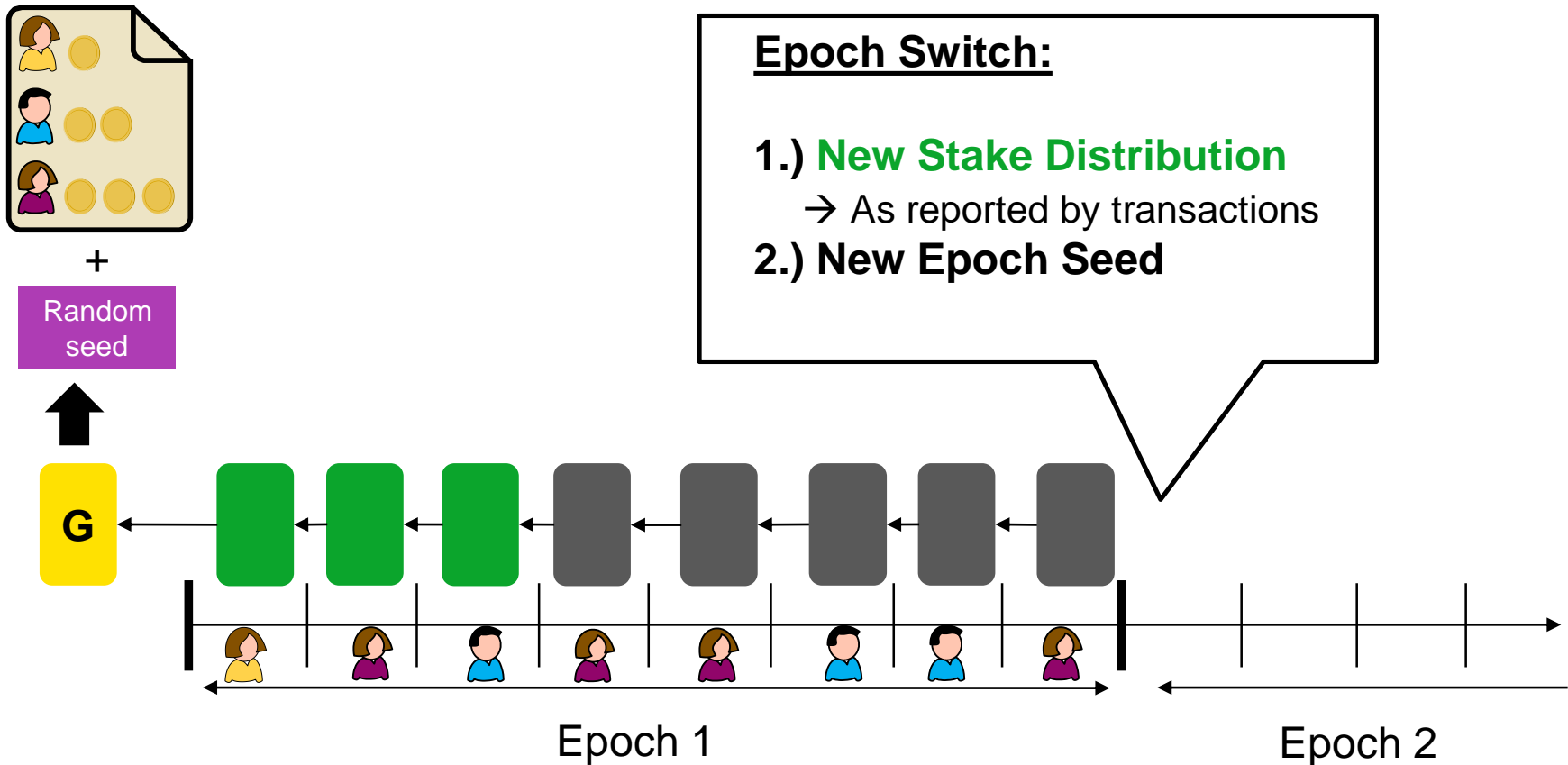
Random
seed



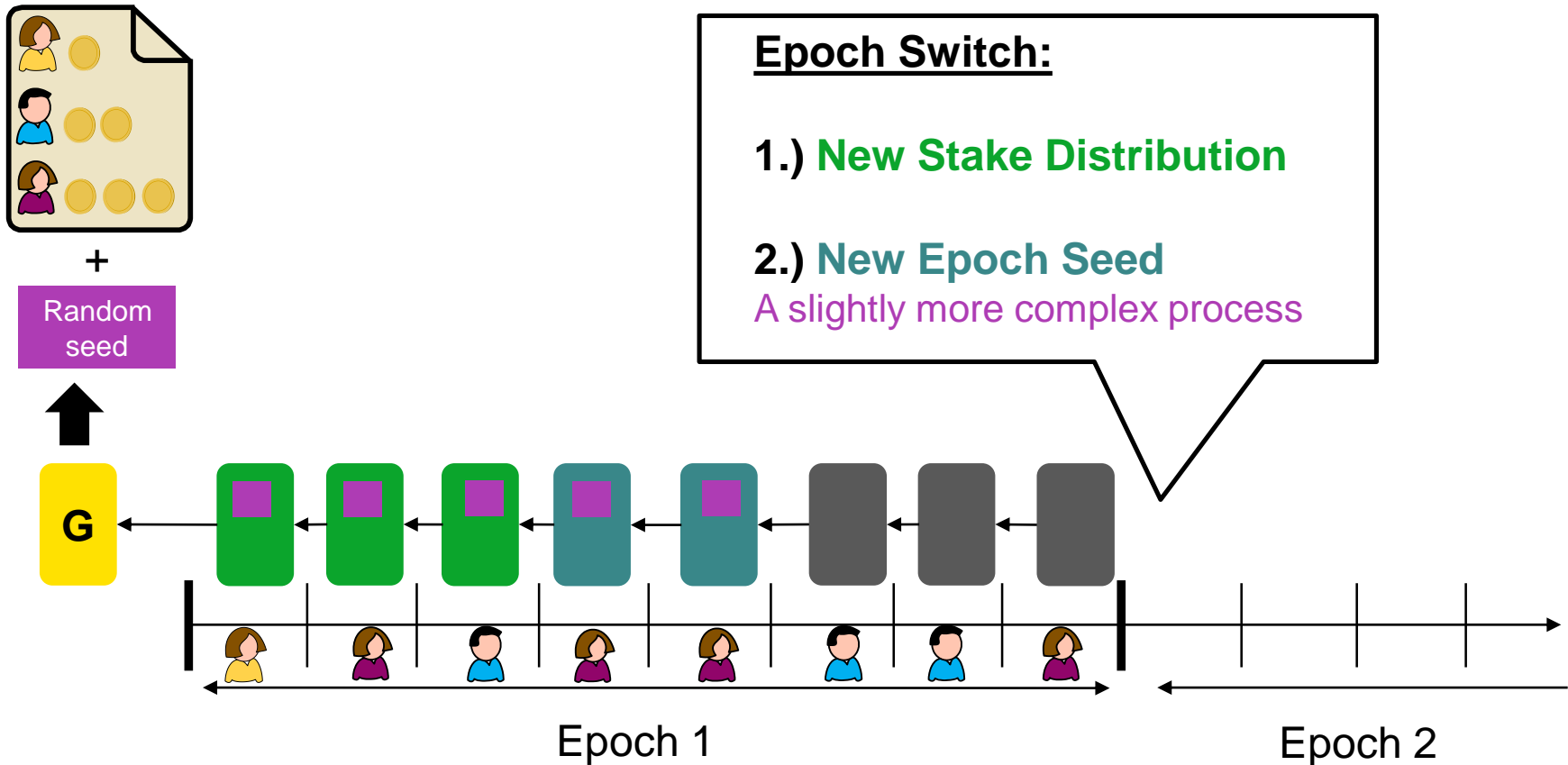
Ouroboros – Protocol Design



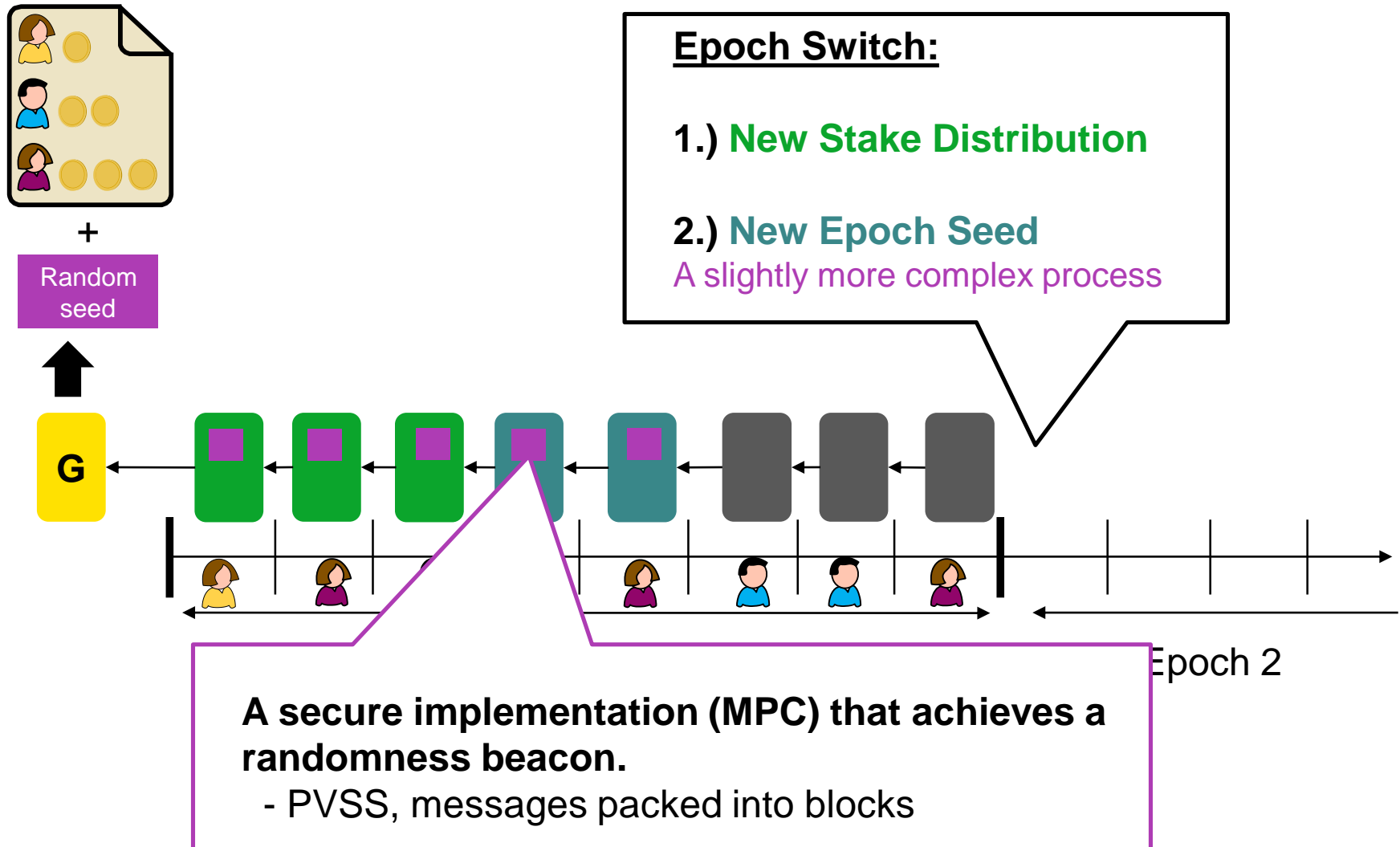
Ouroboros – Protocol Design



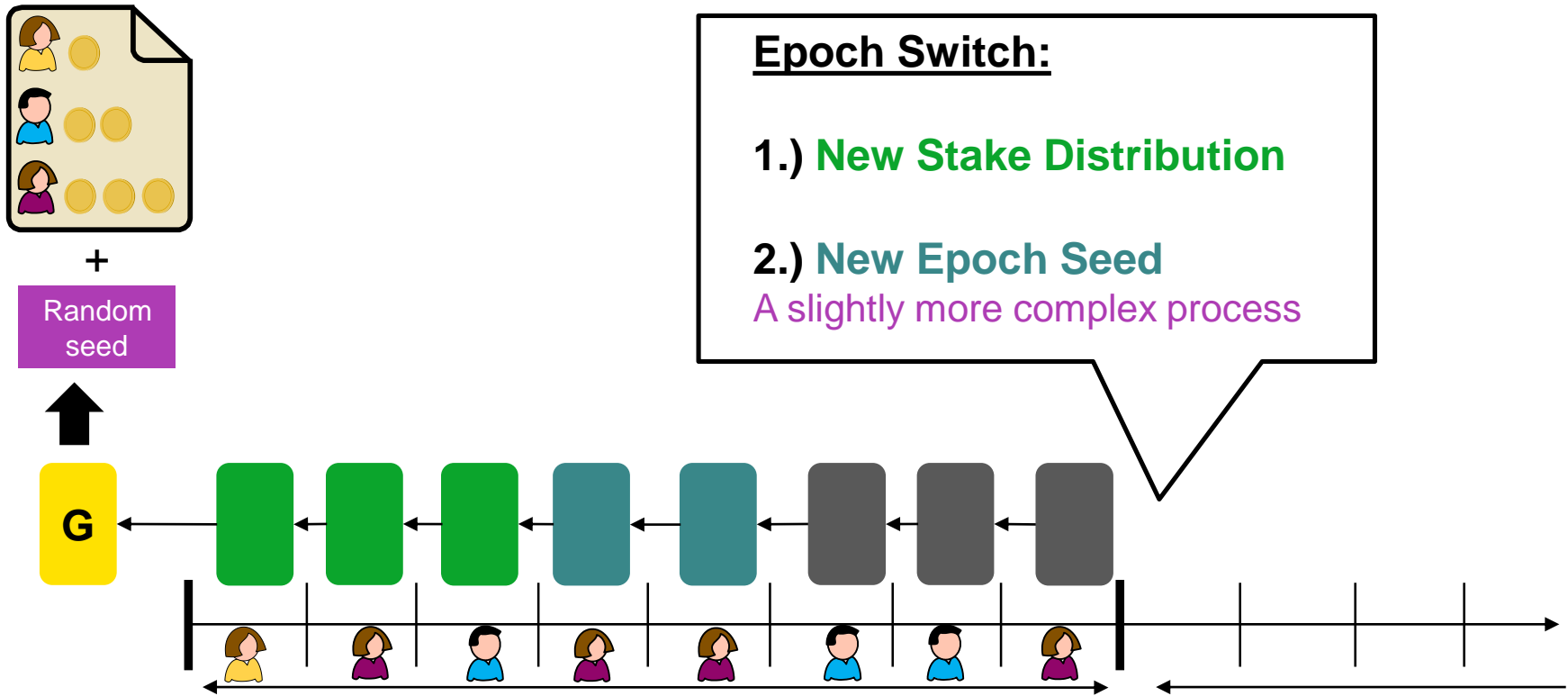
Ouroboros – Protocol Design



Ouroboros – Protocol Design



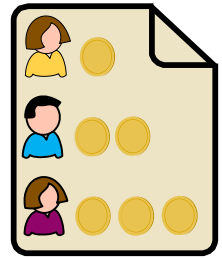
Ouroboros – Protocol Design



Randomness-Beacon Functionality:

- Emits a random value at start of epoch
- Cannot be predicted ahead of time and not tampered

Ouroboros – Protocol Design Summary

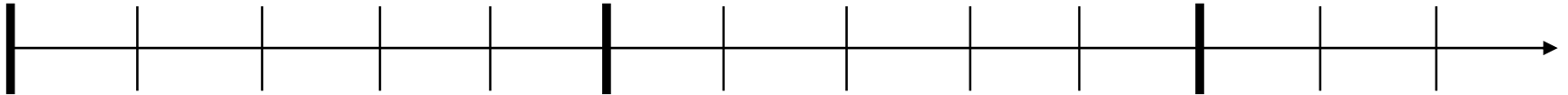


+

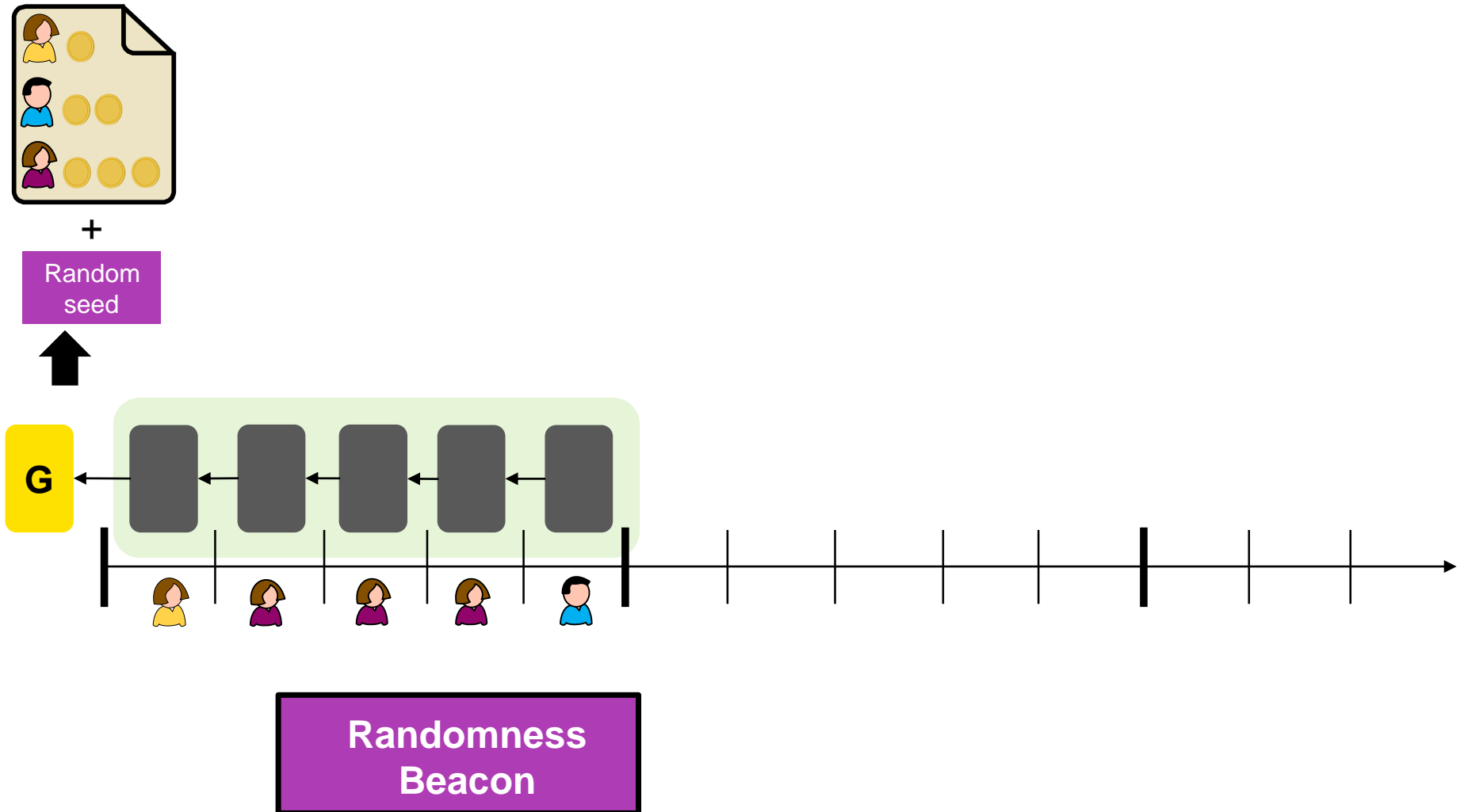
Random
seed



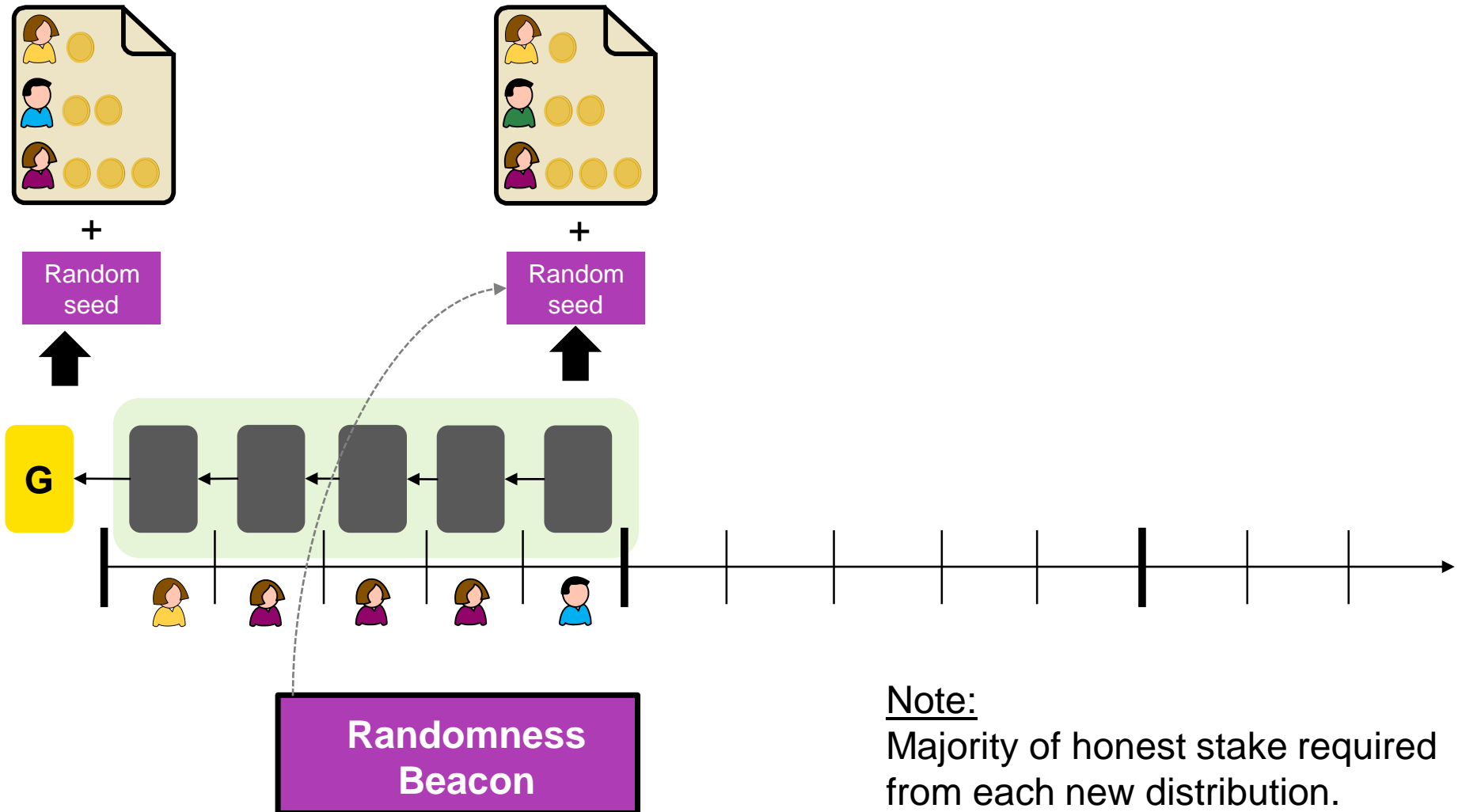
G



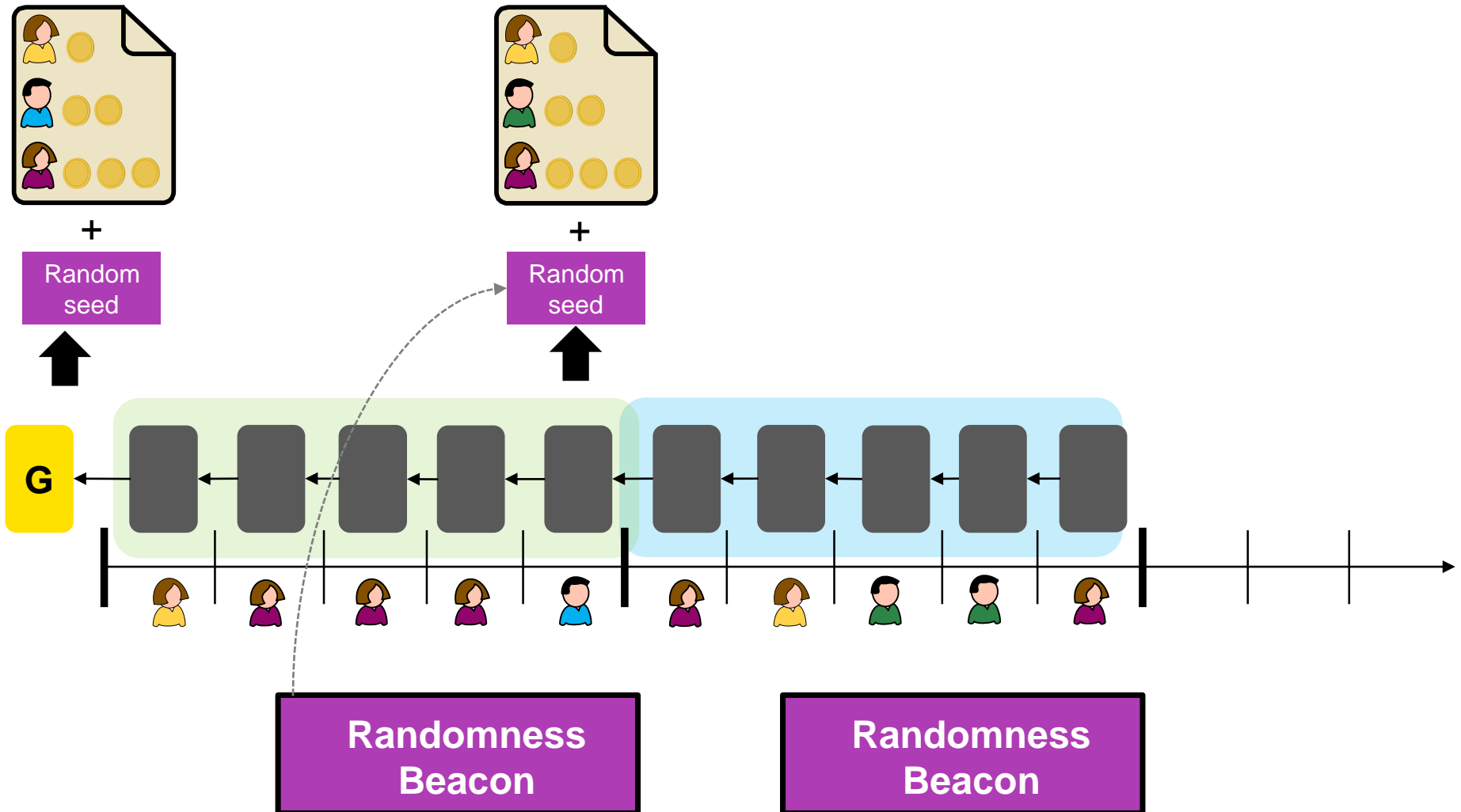
Ouroboros – Protocol Design Summary



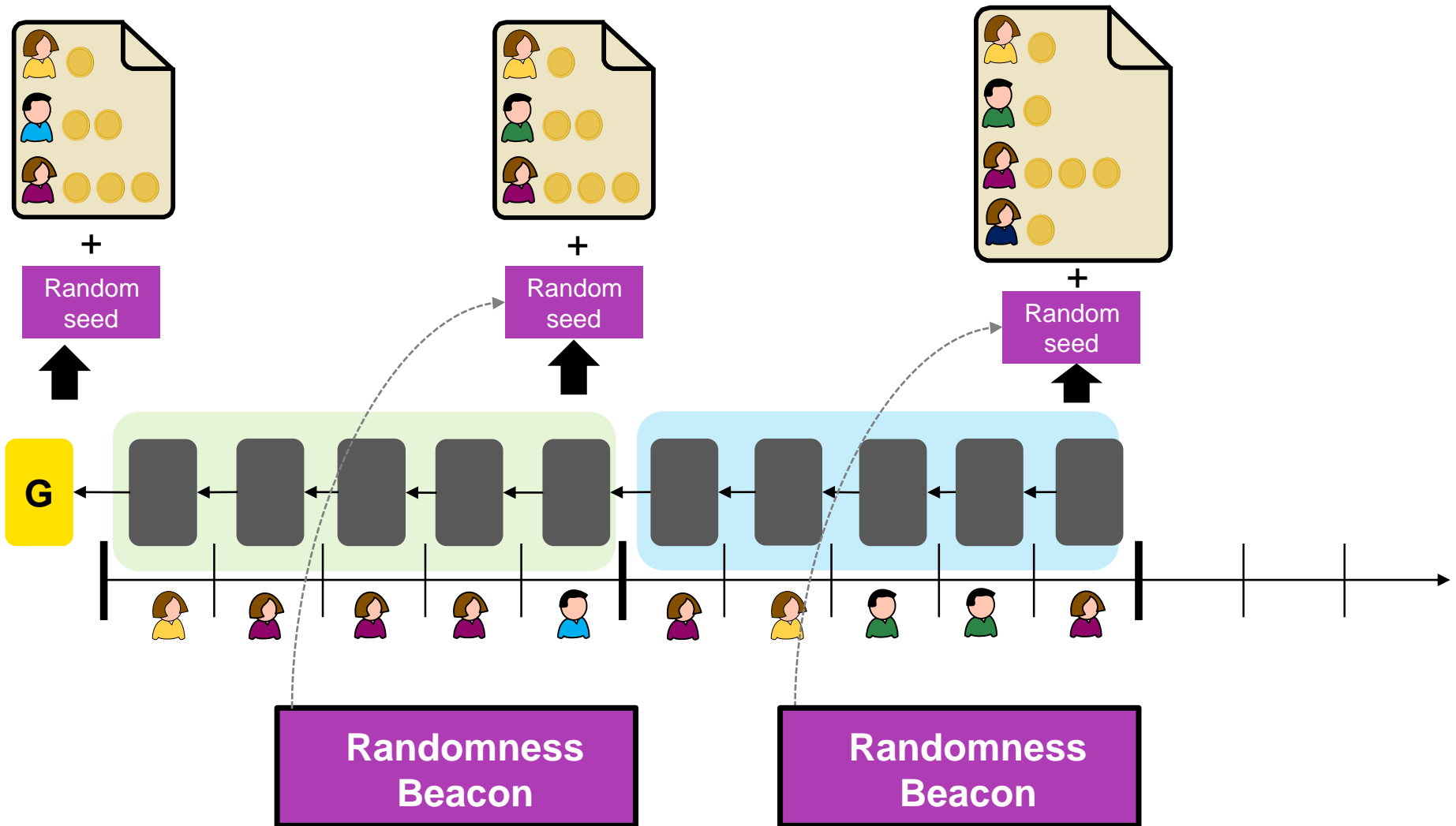
Ouroboros – Protocol Design Summary



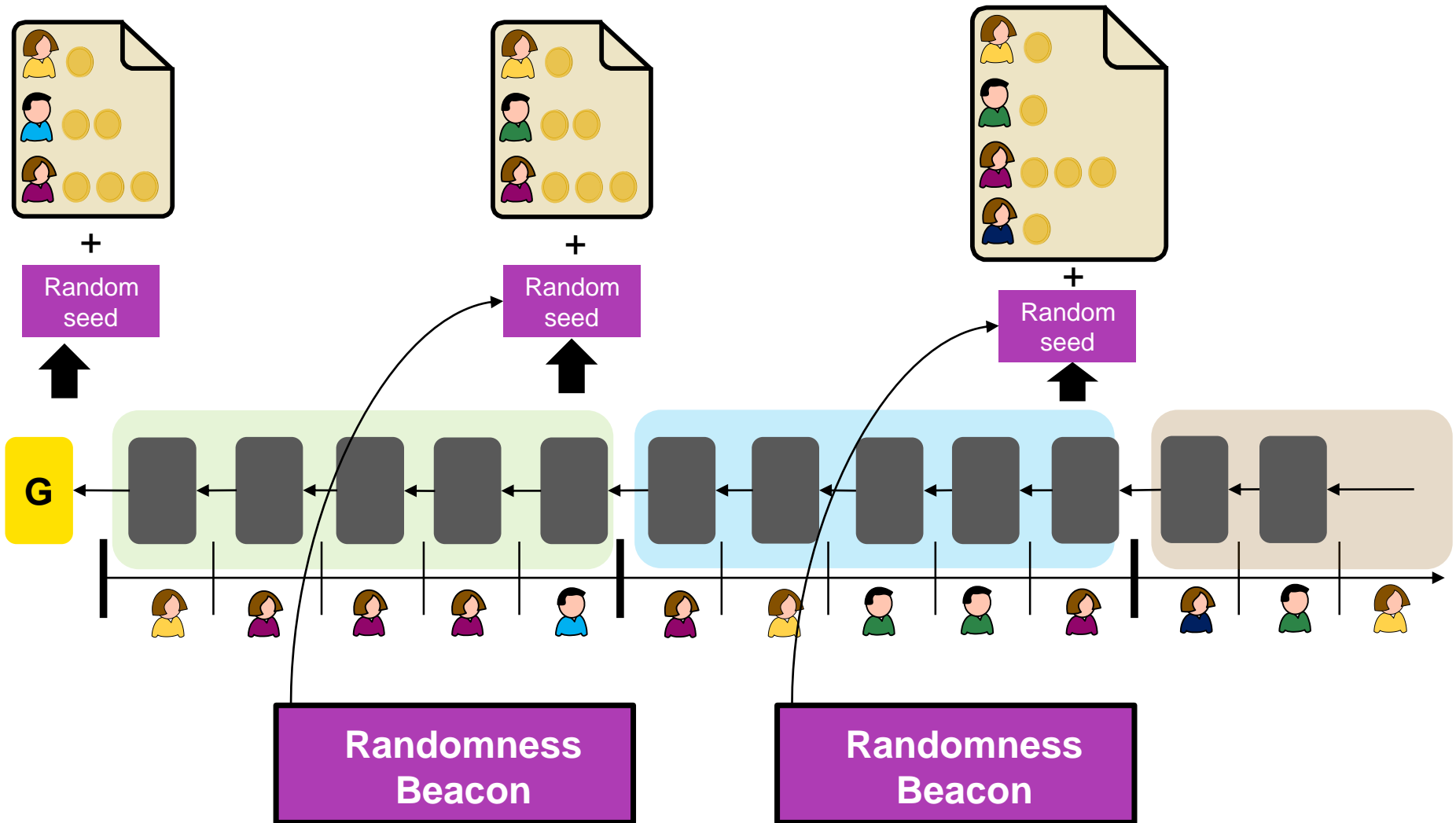
Ouroboros – Protocol Design Summary



Ouroboros – Protocol Design Summary



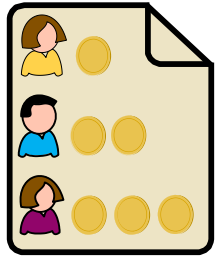
Ouroboros – Protocol Design Summary



Ouroboros – Analysis

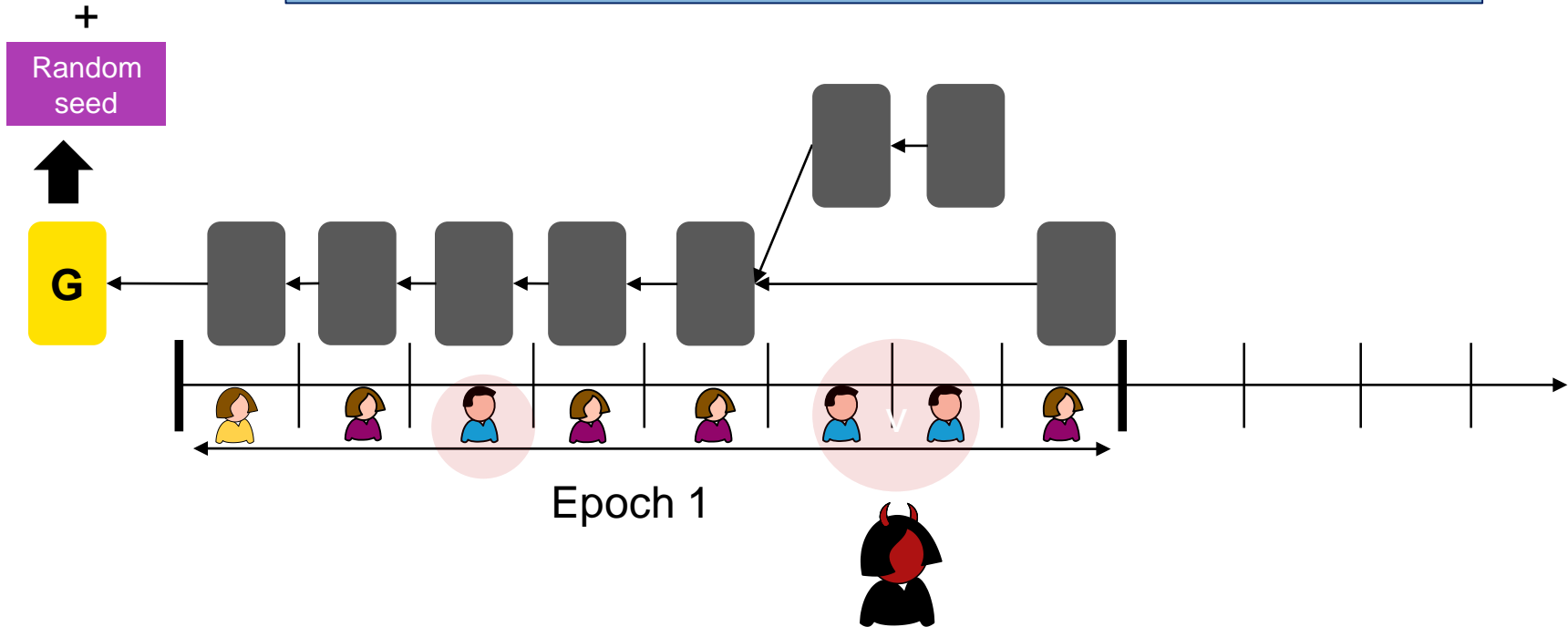
- Analysis of first epoch
- Lifting to multiple epochs (inductive argument)

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.

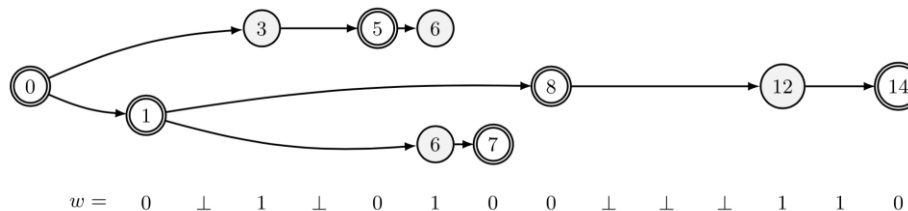
0 ⊥ 1 ⊥ 1 1 ⊥ ⊥ ⊥ ⊥ ⊥ 0 ⊥ ⊥ ⊥ ⊥ ⊥ 0 ⊥ ⊥ ⊥ 0 0 ⊥ ⊥ 0 1 0 ⊥ ⊥ ⊥ 0 1 ⊥

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.

0⊥1⊥11⊥⊥⊥⊥⊥0⊥⊥⊥⊥⊥0⊥⊥⊥00⊥⊥010⊥⊥⊥01⊥

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

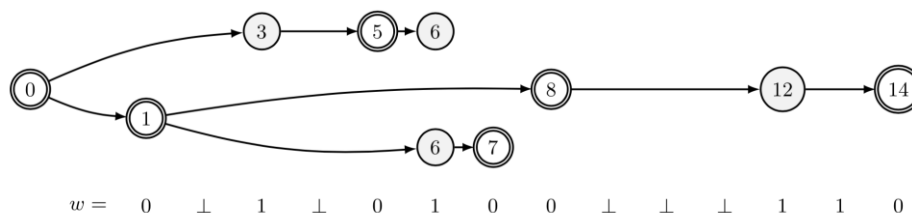


A General Analytical Approach: The Forkable String Analysis

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

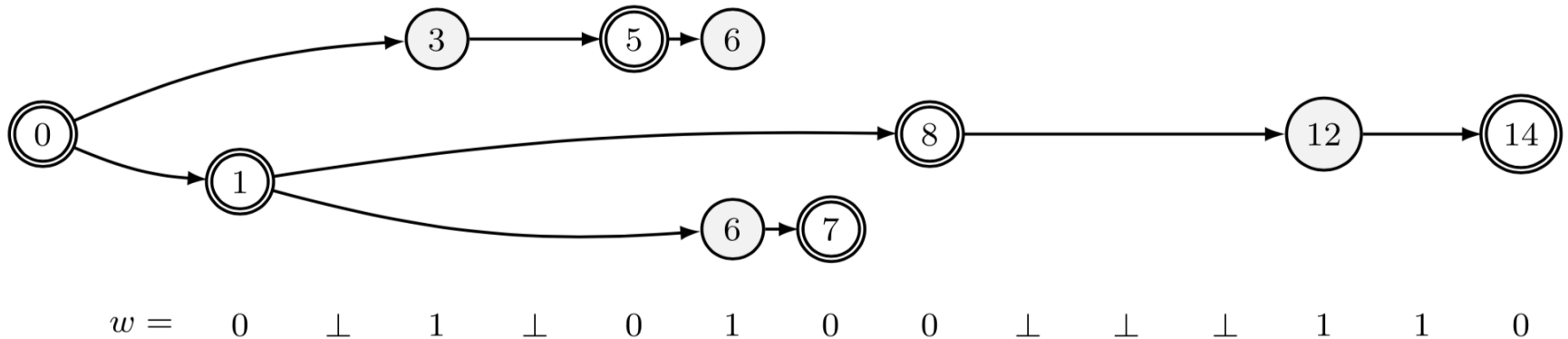
0⊥1⊥11⊥⊥⊥⊥⊥0⊥⊥⊥⊥⊥0⊥⊥⊥00⊥⊥010⊥⊥⊥01⊥

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



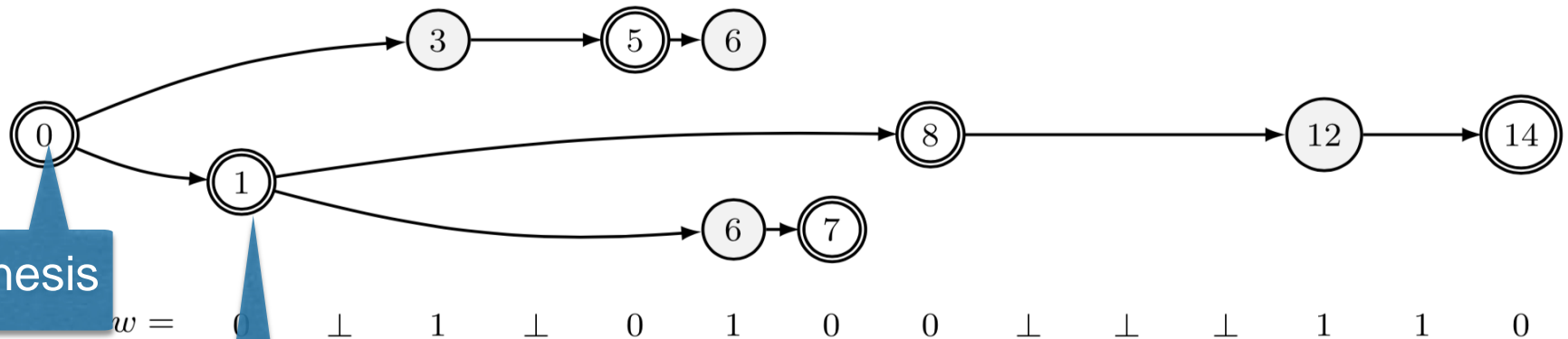
Characteristic string:

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



honest
party

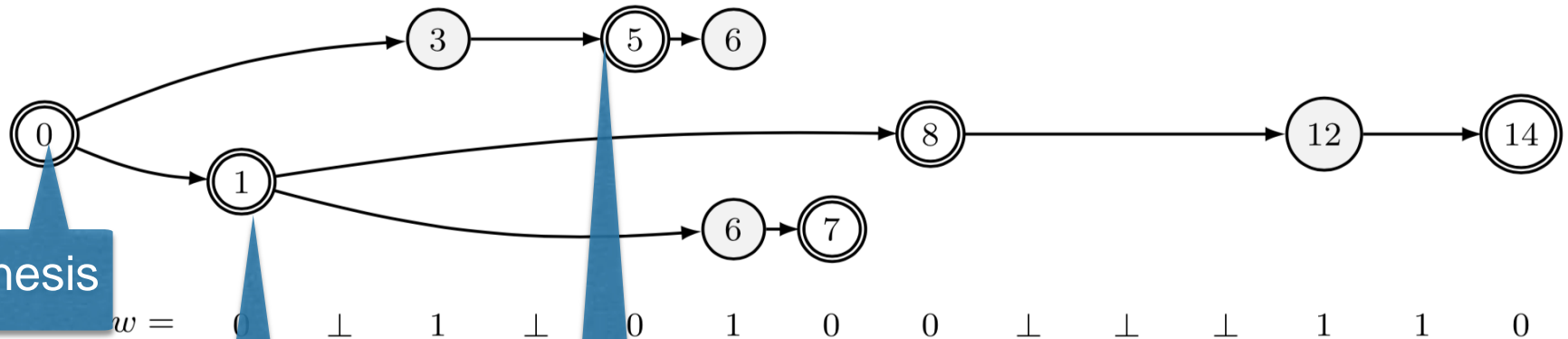
Characteristic string:

0: produces to exactly one honest party.

1: block 1 to a malicious coalition

\perp : Slot cannot be claimed (e.g. if election process would assign no leader)

Forks: Abstracting Protocol Executions



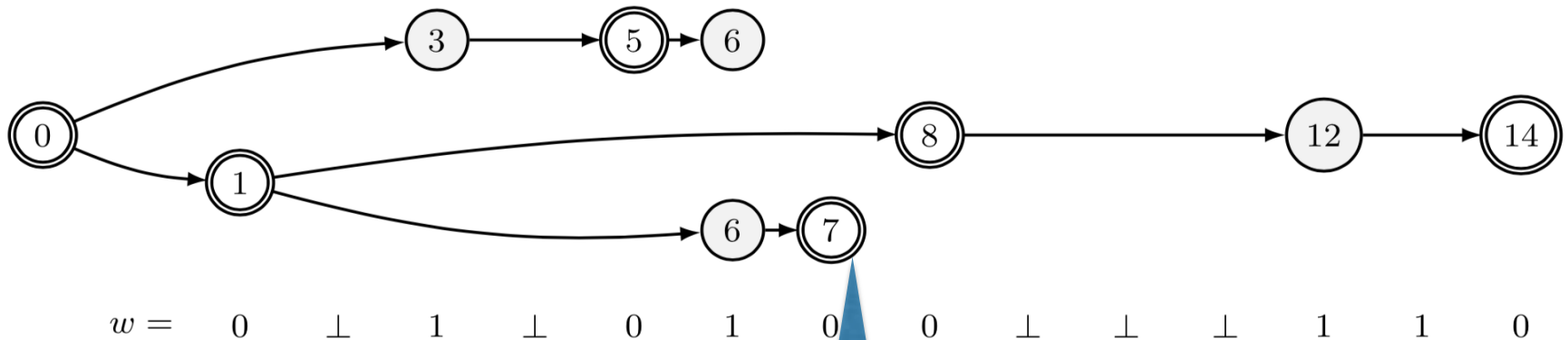
Characteristic string:

0: Slot assigned to exactly one honest party.

1: Slot assigned to a malicious coalition

\perp : Slot cannot be claimed (e.g. if election process would assign no leader)

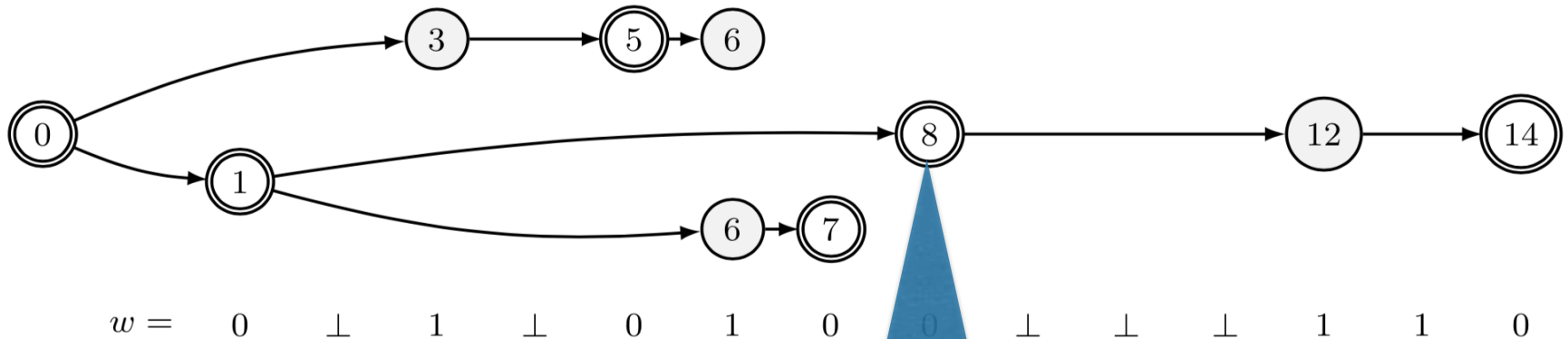
Forks: Abstracting Protocol Executions



Characteristic string:

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

Forks: Abstracting Protocol Executions



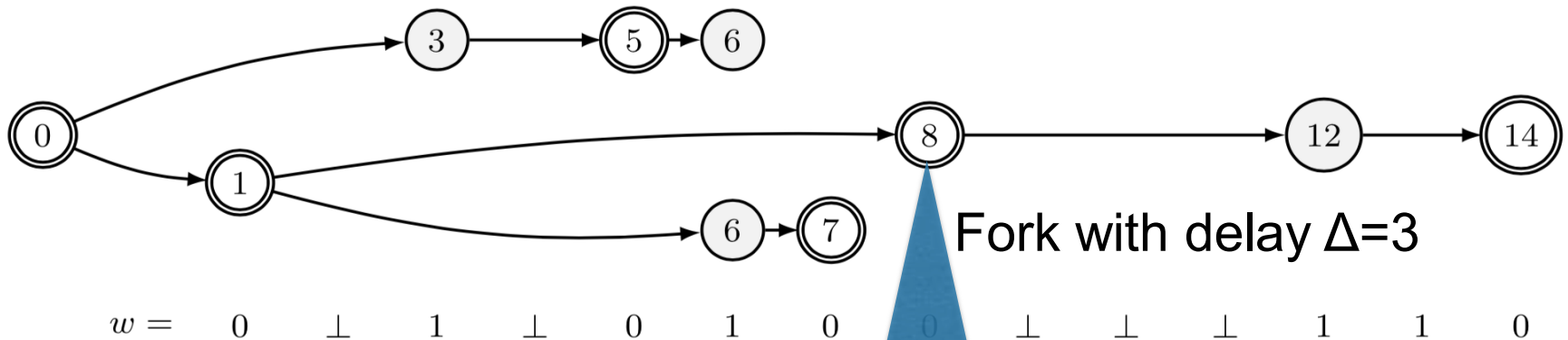
Characteristic string:

0: Slot belongs to exactly one honest party.

1: Slot belongs to a malicious coalition

\perp : Slot cannot be claimed (e.g. if process would assign no leader)

Forks: Abstracting Protocol Executions



Characteristic string:

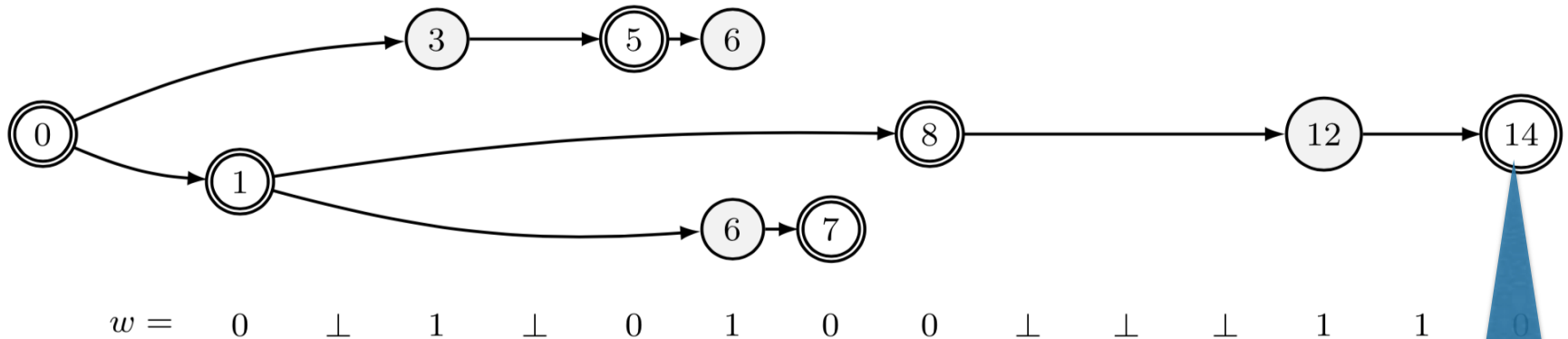
0: Slot belongs to exactly one honest party.

1: Slot belongs to a malicious coalition

\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:

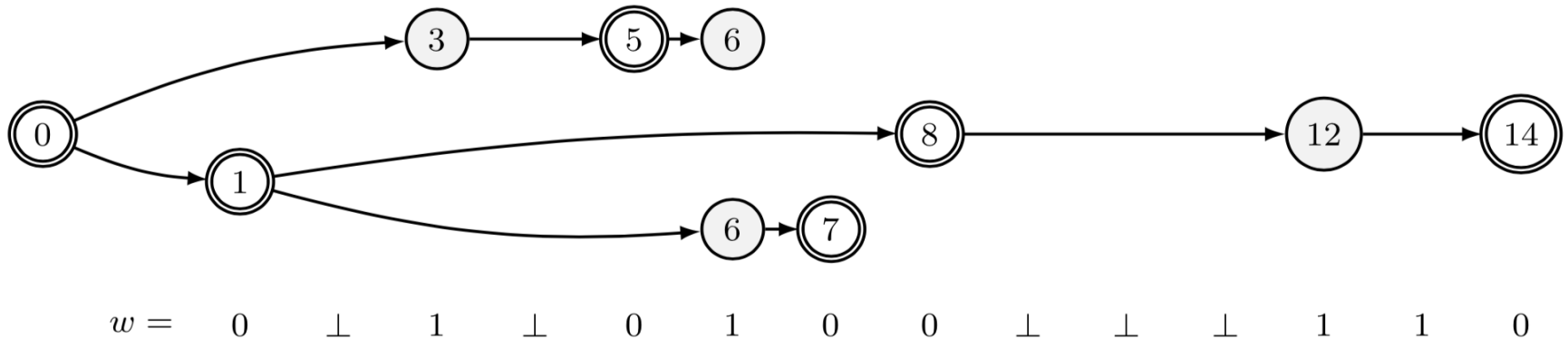
0: Slot belongs to exactly one honest party.

1: Slot belongs to a malicious coalition

\perp : Slot cannot be claimed (e.g. if process would assign no leader)

adversary
serves block 12
to honest party

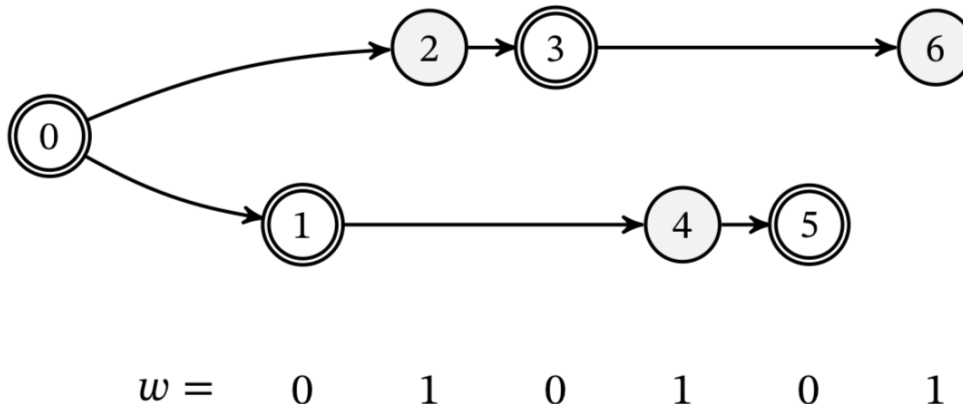
Forks: Abstracting Protocol Executions



Important Property:
Depth of **honest nodes increases** (from left to right) if more than Δ slots apart.
(Lower bound on the depth of the fork)

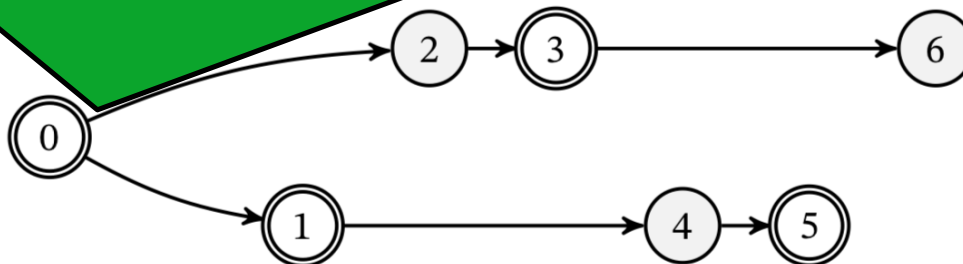
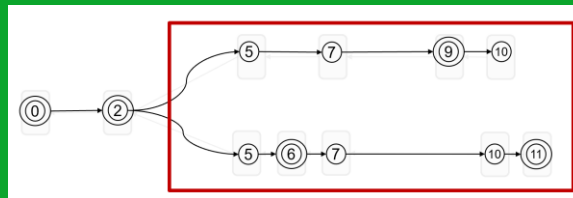
Combinatorics of Characteristic Strings

- 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.



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).



$w =$ 0 1 0 1 0 1

Drawing from Bitcoin analysis

Difference:

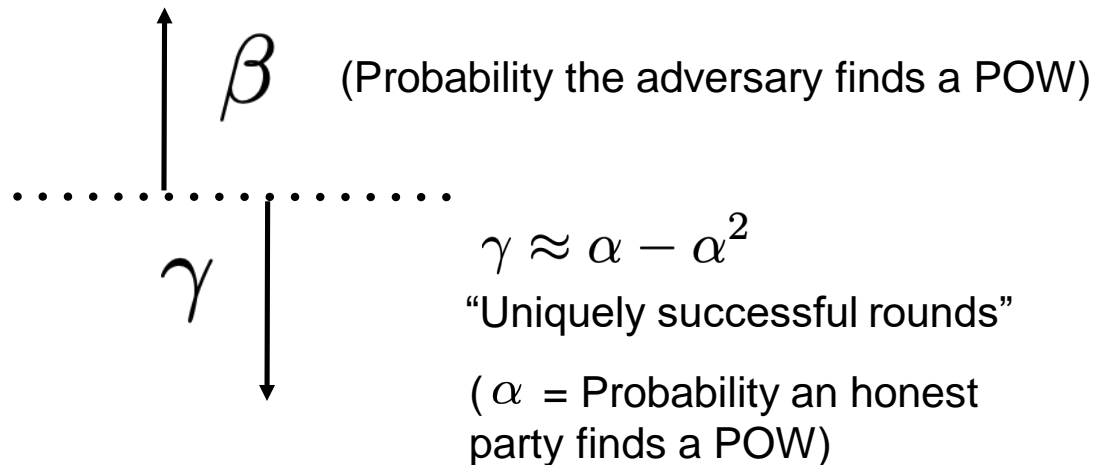
1.) #PoWs of
adversary in
time segment

-

2.) #PoWs of
honest parties
in time segment

5
4
3
2
1
0
-1
-2
-3
-4
-5

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



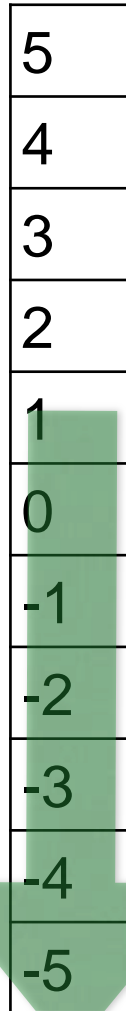
Drawing from Bitcoin analysis

Difference:

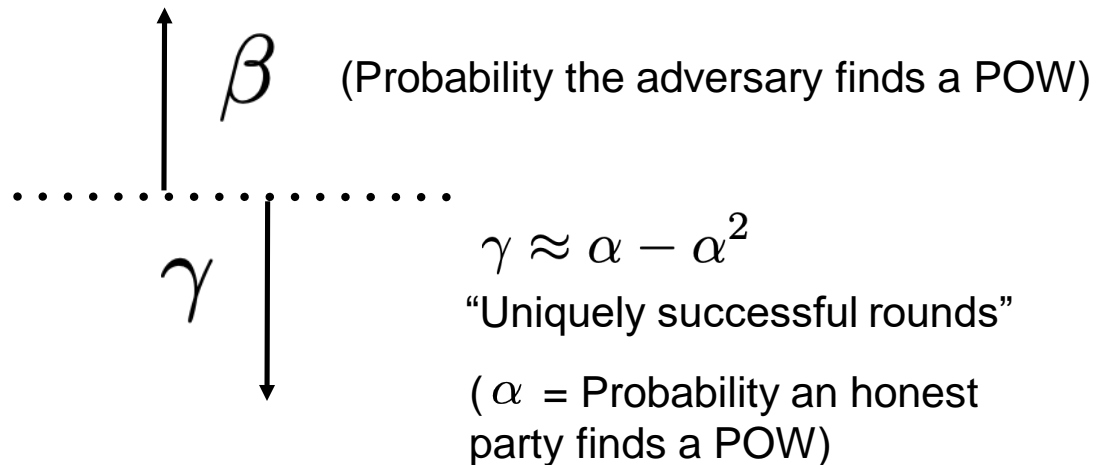
1.) #PoWs of
adversary in
time segment

-

2.) #PoWs of
honest parties
in time segment



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



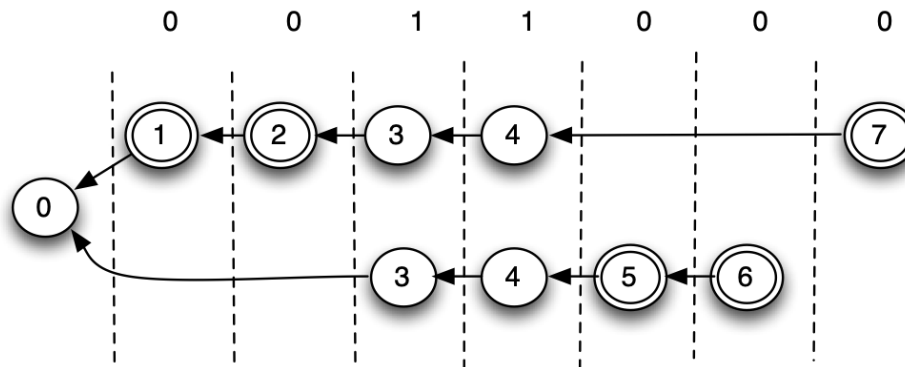
- A favorable step is downwards.
- Such a step is more likely by assumption $\gamma > \beta$.

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

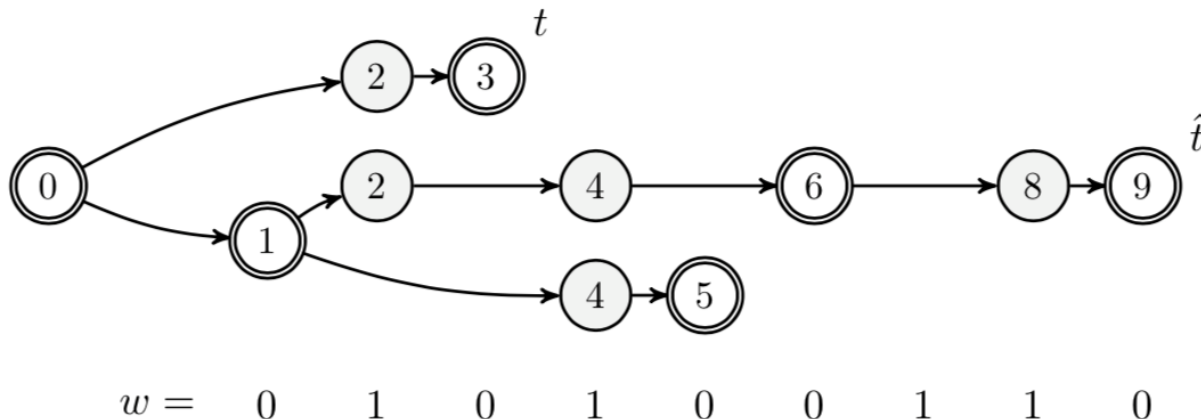


Forkable Strings

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

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

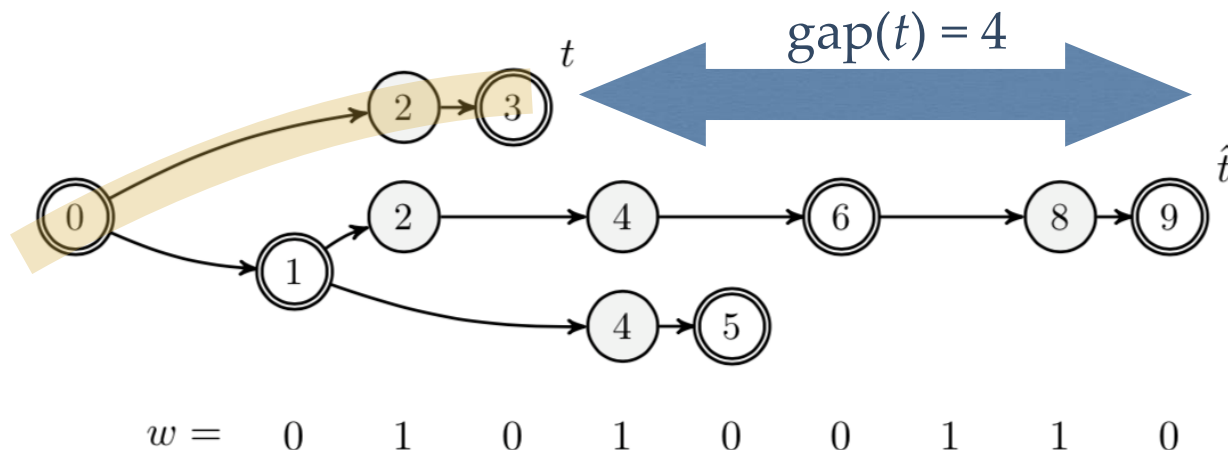
- **gap(t)**: length difference with leading honest node.
- **reserve(t)**: number of adversarial slots after end of t .
- **reach(t)** := reserve(t) - gap(t).



Forkable Strings


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

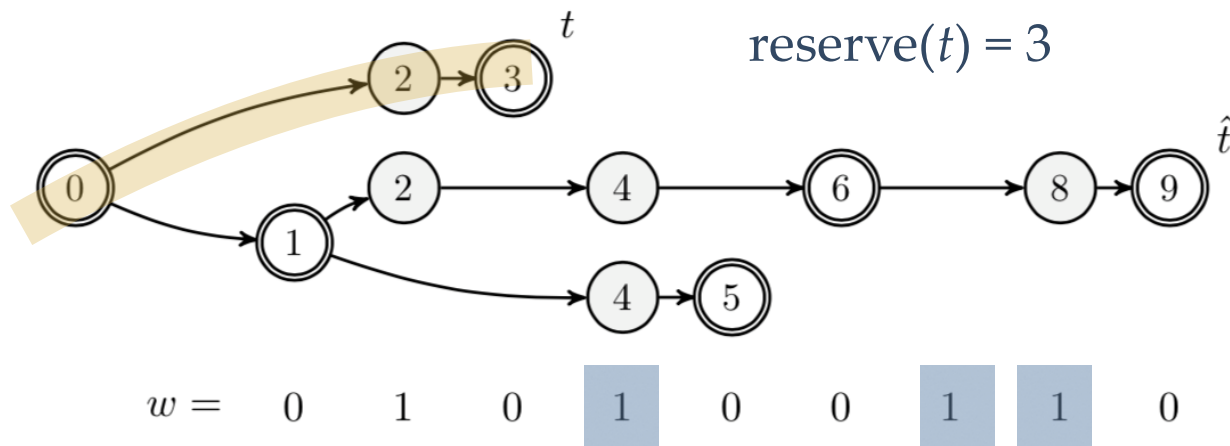
- **gap(t)**: length difference with leading honest node.
- **reserve(t)**: number of adversarial slots after end of t .
- **reach(t) := reserve(t) - gap(t)**.



Forkable Strings

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

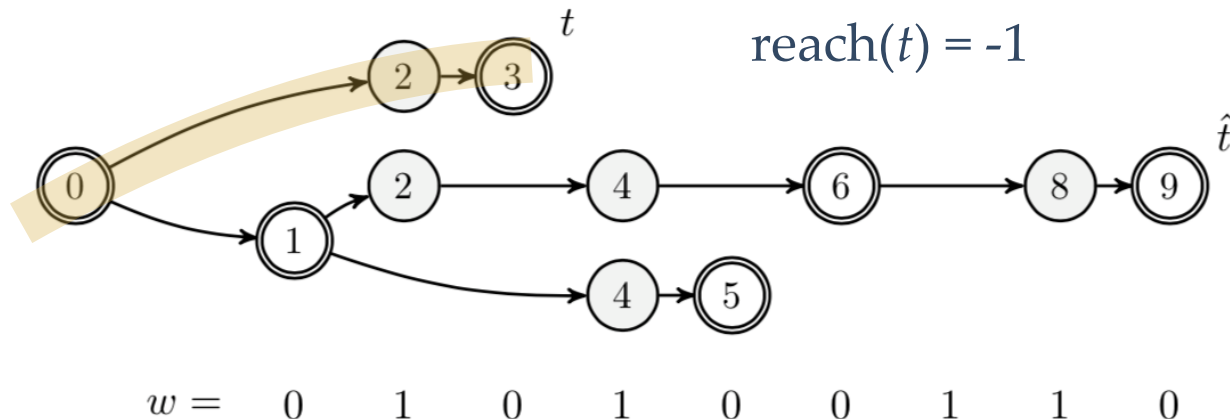
- 
- **gap(t)**: length difference with leading honest node.
 - **reserve(t)**: number of adversarial slots after end of t .
 - **reach(t) := reserve(t) - gap(t)**.



Forkable Strings

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

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



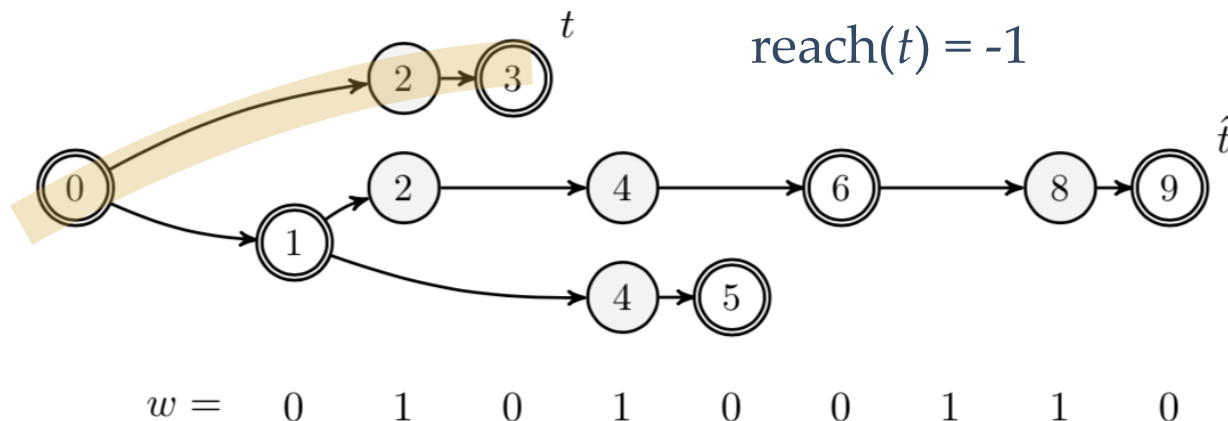
Forkable Strings

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

of interest to

honest node.

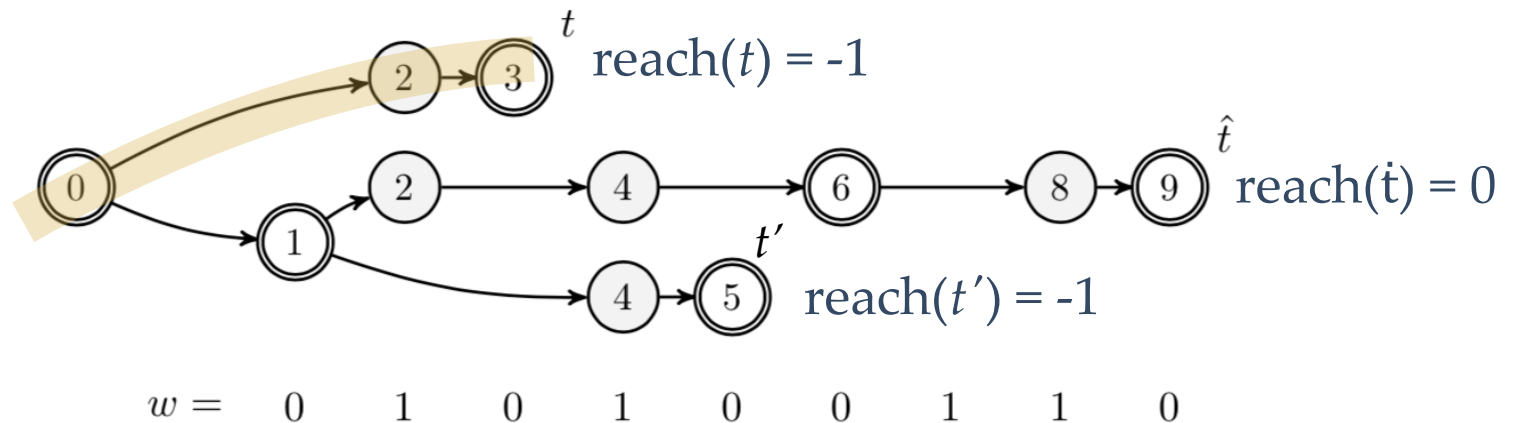
- $reserve(t)$: number of adversarial slots after end of t .
- $reach(t) := reserve(t) - gap(t)$.



Forkable Strings

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

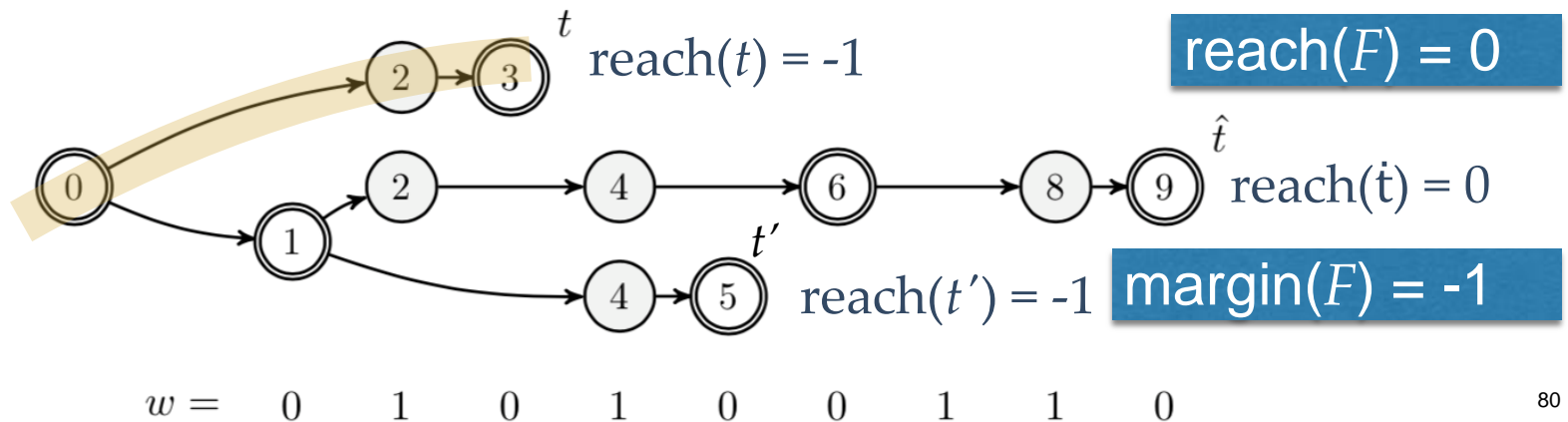
- **reach(F):** $\max \text{reach}(t)$
- **margin(F):** second highest & disjoint $\text{reach}(t')$



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')

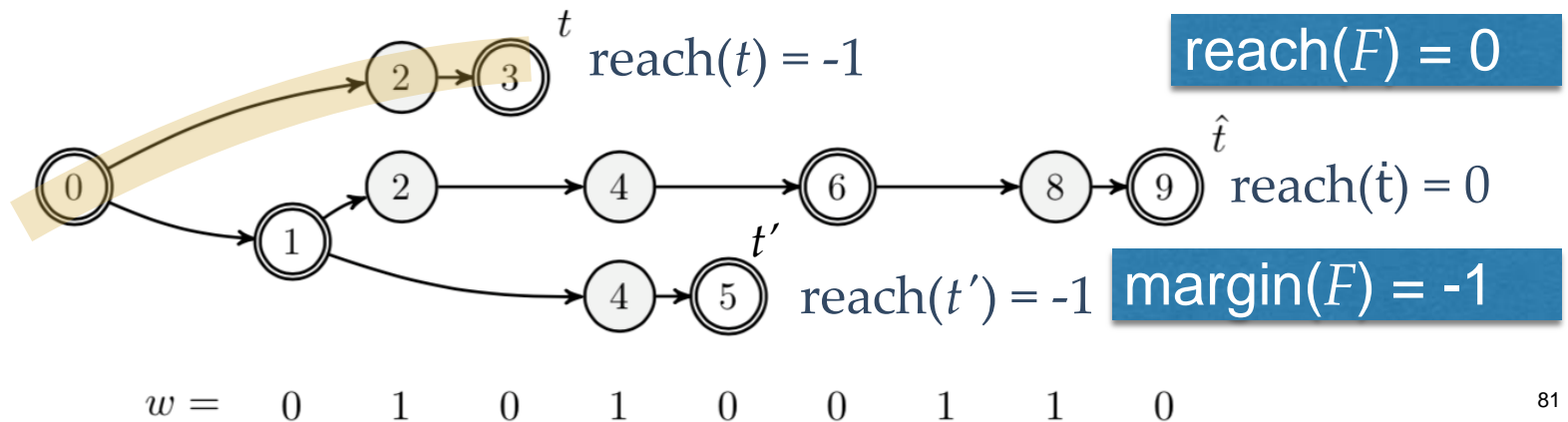


Forkable Strings

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

Looking

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



Reach & Margin

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

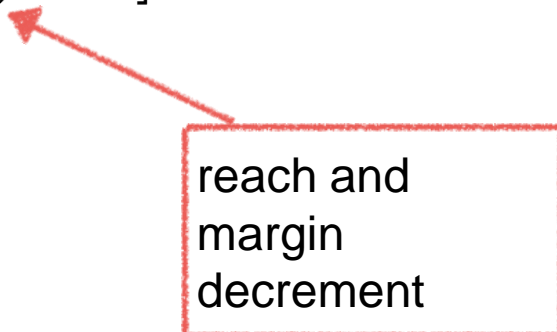
$$[\rho(w1), \mu(w1)] = [\rho(w) + 1, \mu(w) + 1]$$

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

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

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

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

reach never
drops below 0

reach and
margin
decrement

Recursive Formula for Reach & Margin

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

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

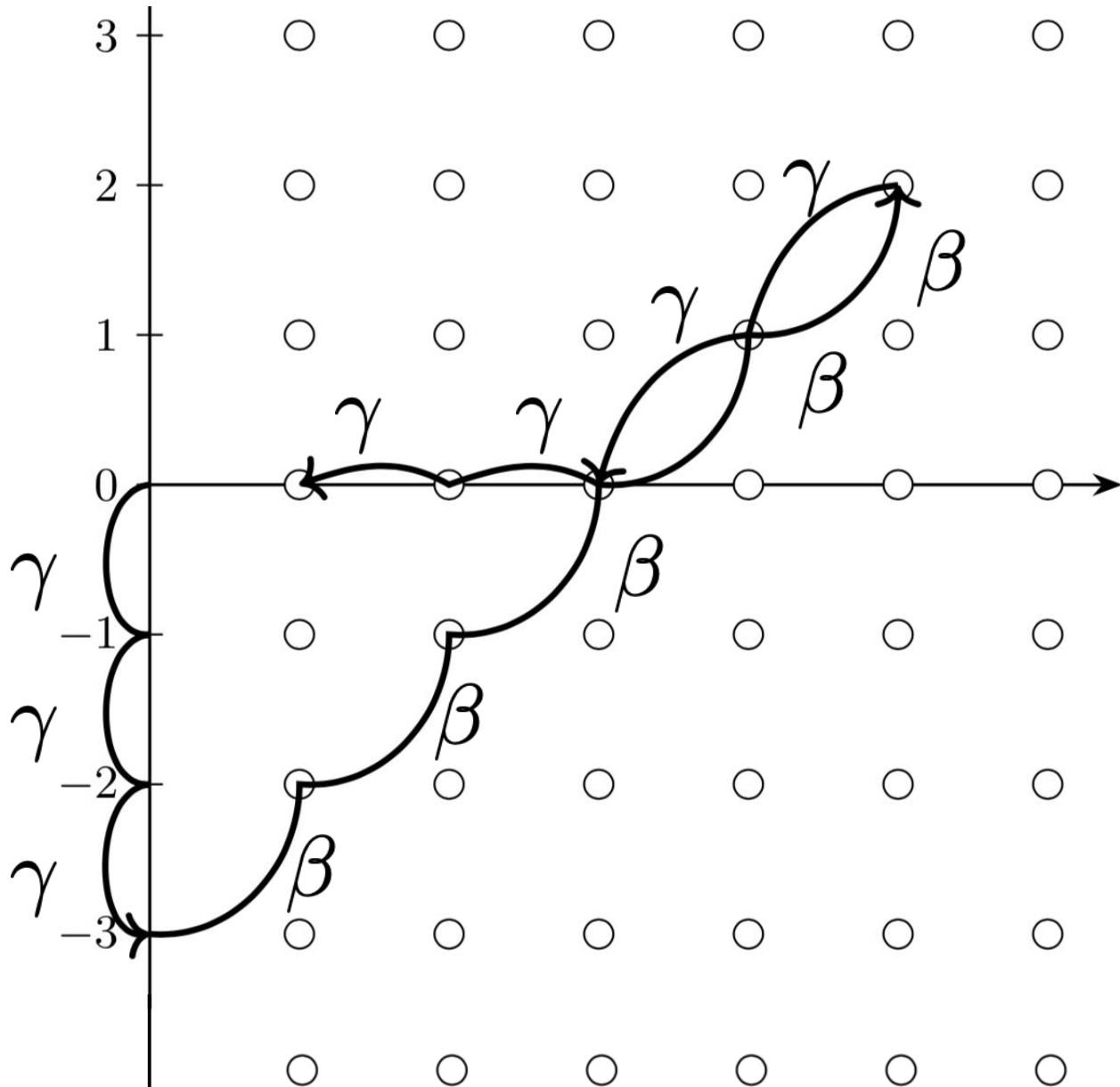
it is possible for the adversary to compensate for the margin, by sacrificing reach

reach never drops below 0

reach and margin decrement

2D Random Walk

margin



reach

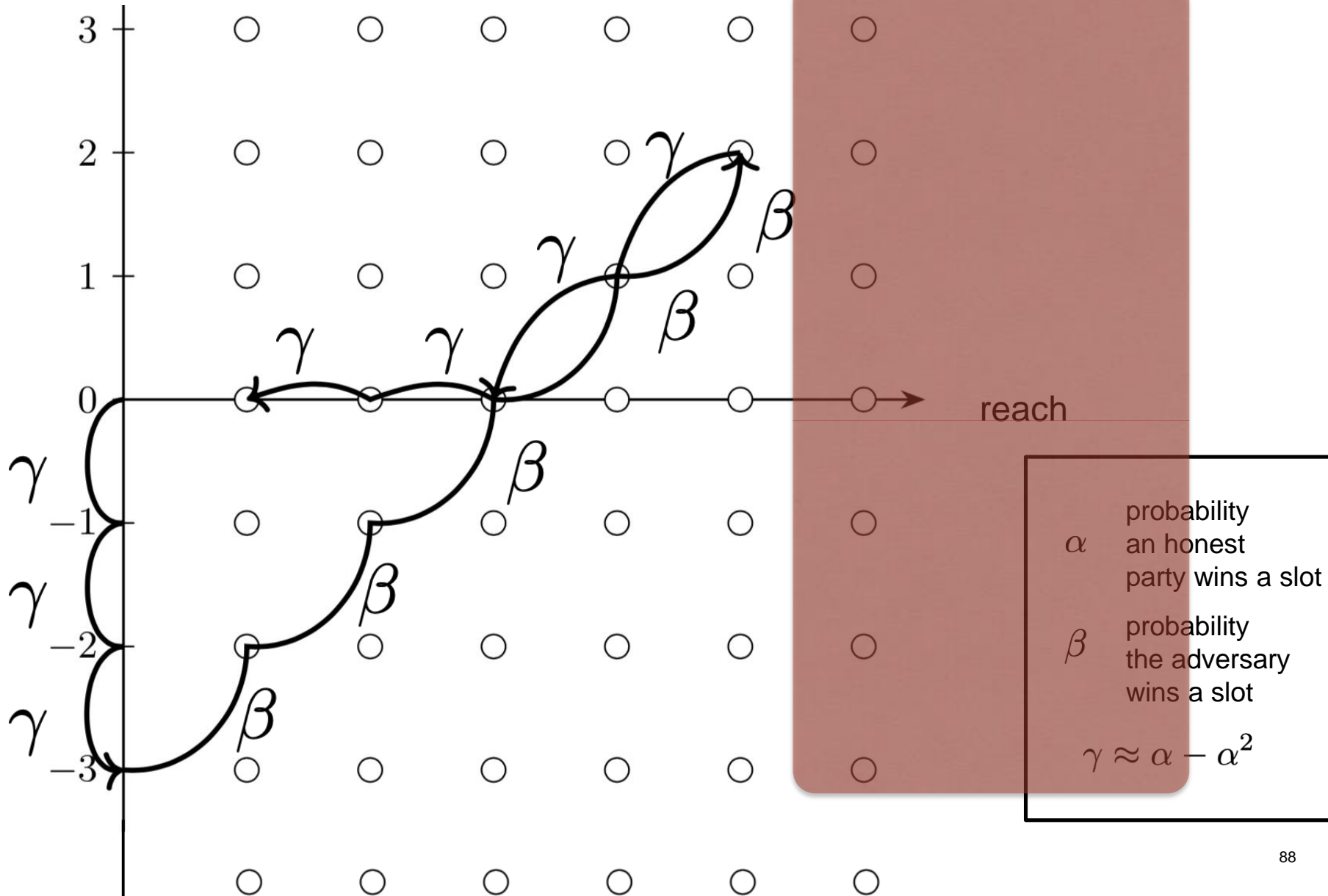
α probability
an honest
party wins a slot

β probability
the adversary
wins a slot

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

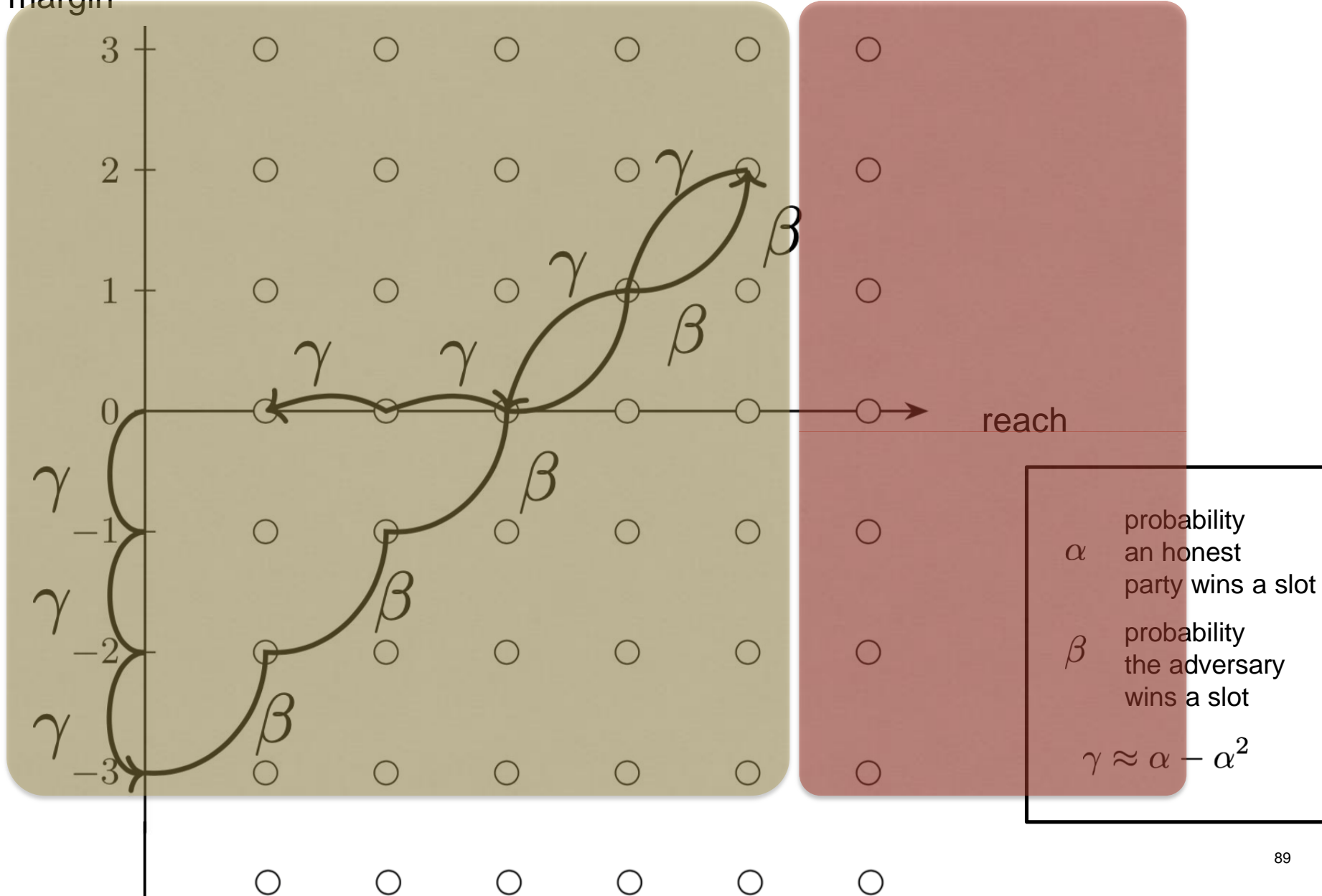
2D Random Walk

margin



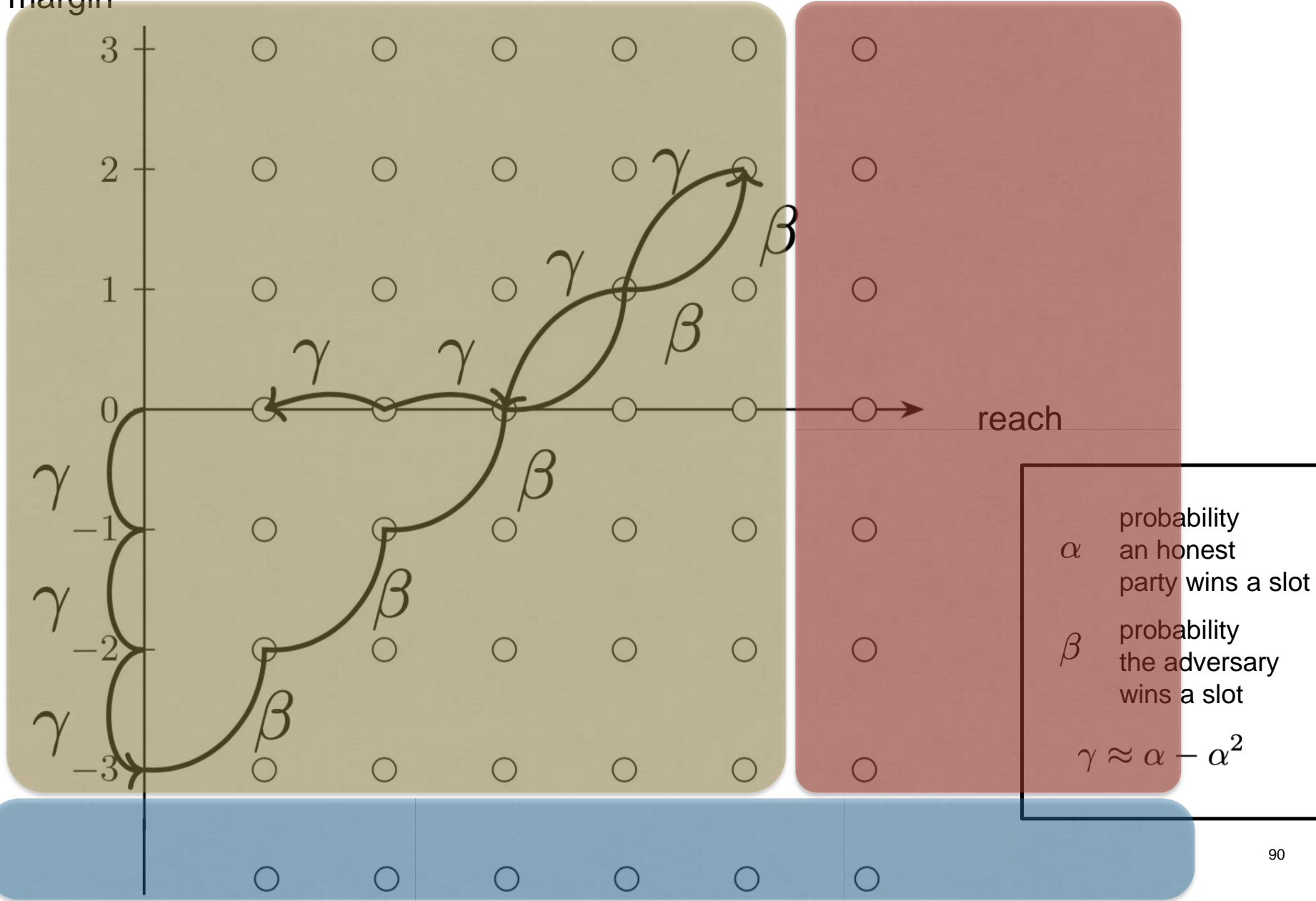
2D Random Walk

margin



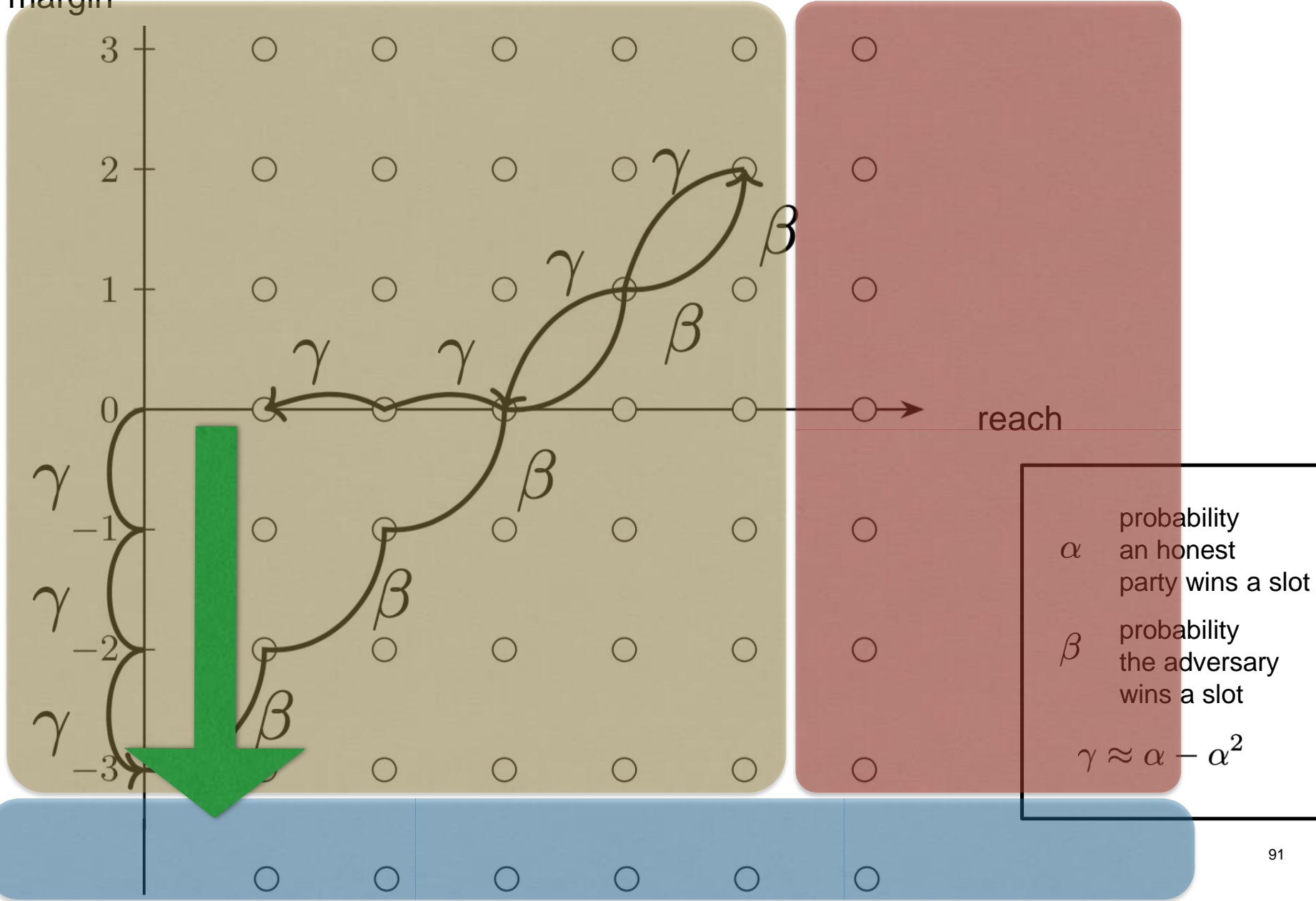
2D Random Walk

margin



2D Random Walk

margin



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volatile We let Vol_t denote the event that $-\delta\sqrt{n} \leq M_{(t)} \leq R_{(t)} < \delta\sqrt{n}$.

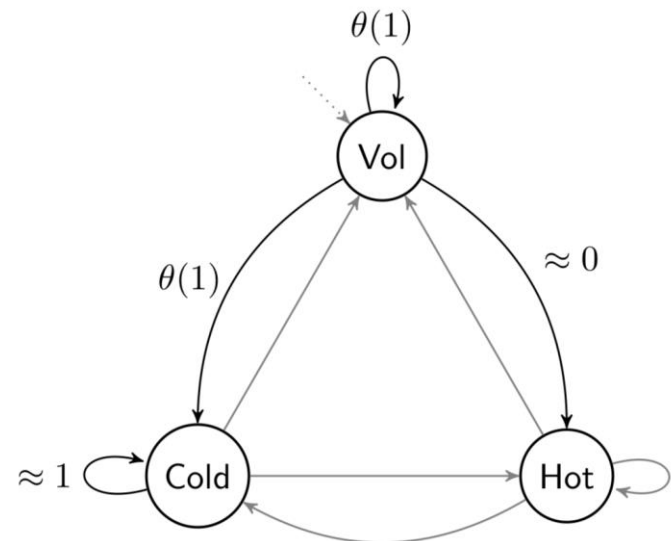
Cold We let Cold_t denote the event that $M_{(t)} < -\delta\sqrt{n}$.

Analysis shows:

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volatile We let Vol_t denote the event that $-\delta\sqrt{n} \leq M_{(t)} < R_{(t)} < \delta\sqrt{n}$.

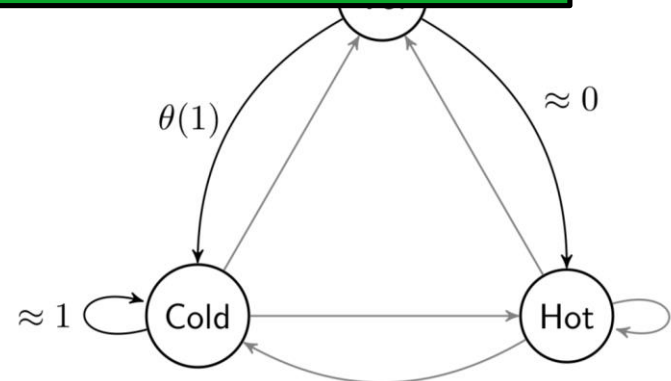
Cold We let Cold_t denote the event that $R_{(t)} < -\delta\sqrt{n}$ and $M_{(t)} < \delta\sqrt{n}$.

$R_{(t)}, M_{(t)}$:
Reach resp. margin after $t\sqrt{n}$ steps
of the random walk
("coarse grained steps of the walk").

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volatile We let Vol_t denote the event that $-\delta\sqrt{n} \leq M_{(t)} \leq R_{(t)} < \delta\sqrt{n}$.

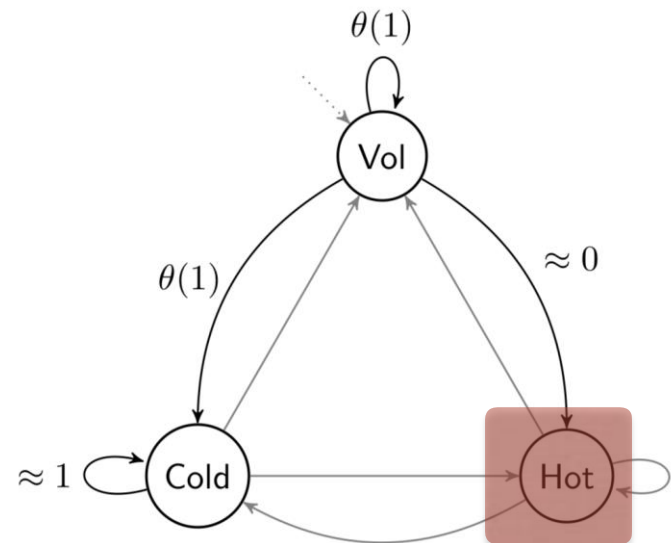
Cold We let Cold_t denote the event that $M_{(t)} < -\delta\sqrt{n}$.

Analysis shows:

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volatile We let Vol_t denote the event that $-\delta\sqrt{n} \leq M_{(t)} \leq R_{(t)} < \delta\sqrt{n}$.

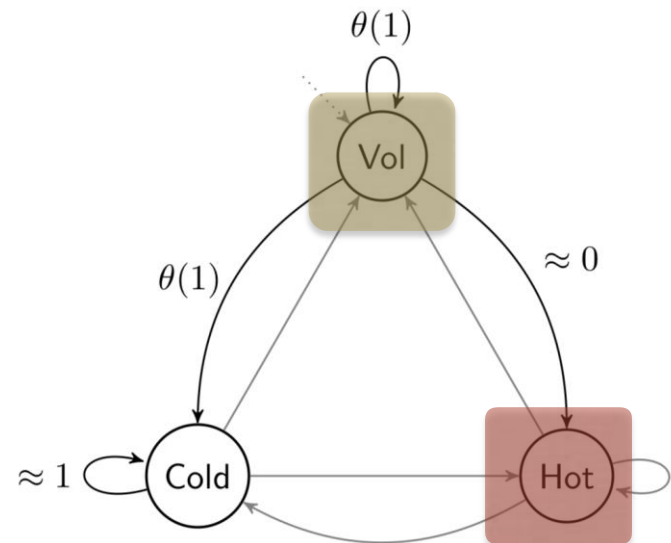
Cold We let Cold_t denote the event that $M_{(t)} < -\delta\sqrt{n}$.

Analysis shows:

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volatile We let Vol_t denote the event that $-\delta\sqrt{n} \leq M_{(t)} \leq R_{(t)} < \delta\sqrt{n}$.

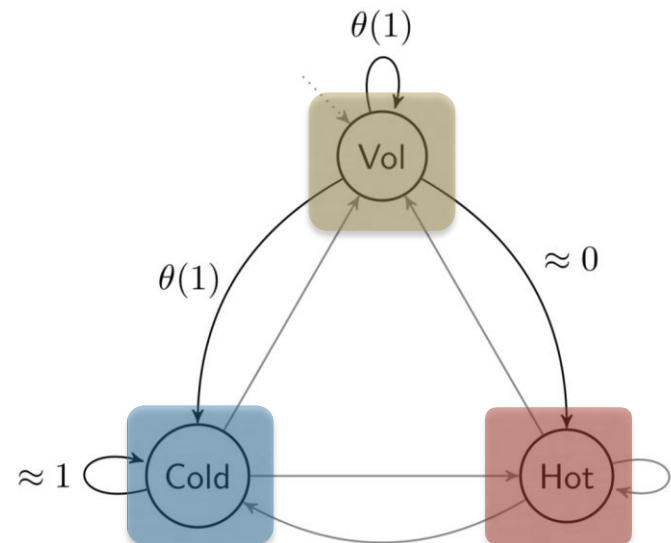
Cold We let Cold_t denote the event that $M_{(t)} < -\delta\sqrt{n}$.

Analysis shows:

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot We let Hot_t denote the event that $R_{(t)} \geq \delta\sqrt{n}$ and $M_{(t)} \geq -\delta\sqrt{n}$.

Volat An improved analysis shows an error bound of $\sqrt{n} \leq M_{(t)} \leq R_{(t)} < \delta\sqrt{n}$.

Cold $< -\delta\sqrt{n}$.

An improved analysis shows an error bound of

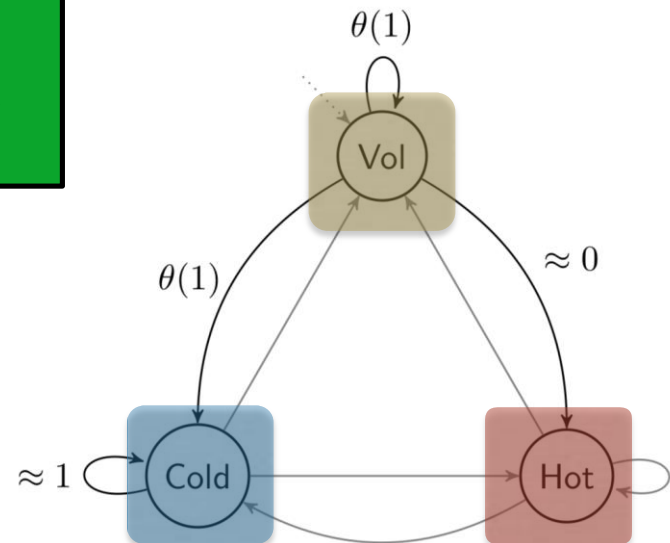
$$2^{-\Omega(n)}$$

***“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.

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(n)},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Analysis

$$R_t = \rho(w_1 \dots w_t) \quad \text{and} \quad M_t = \mu(w_1 \dots w_t).$$

Hot

Volat

Cold

Conclusion:

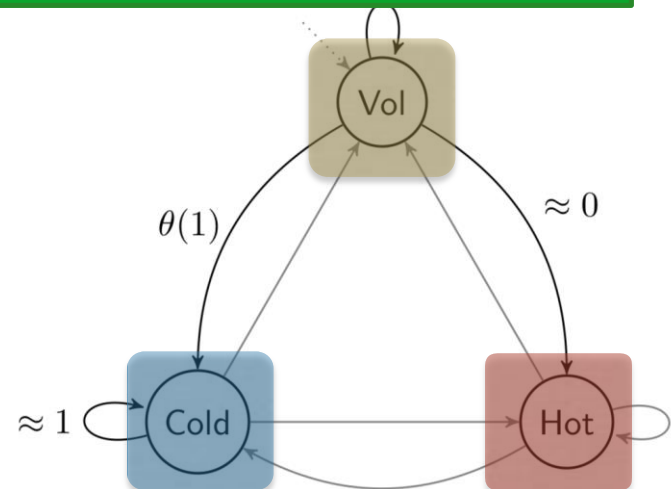
- Characteristic string not forkable (w.h.p.)
→ No long diverging paths
- Common prefix achieved (w.h.p.)

Analysis shows:

$$\Pr[\text{Cold}_{t+1} \mid \text{Cold}_t] \geq 1 - 2^{-\Omega(\sqrt{n})},$$

$$\Pr[\text{Cold}_{t+1} \mid \text{Vol}_t] \geq \Omega(\epsilon),$$

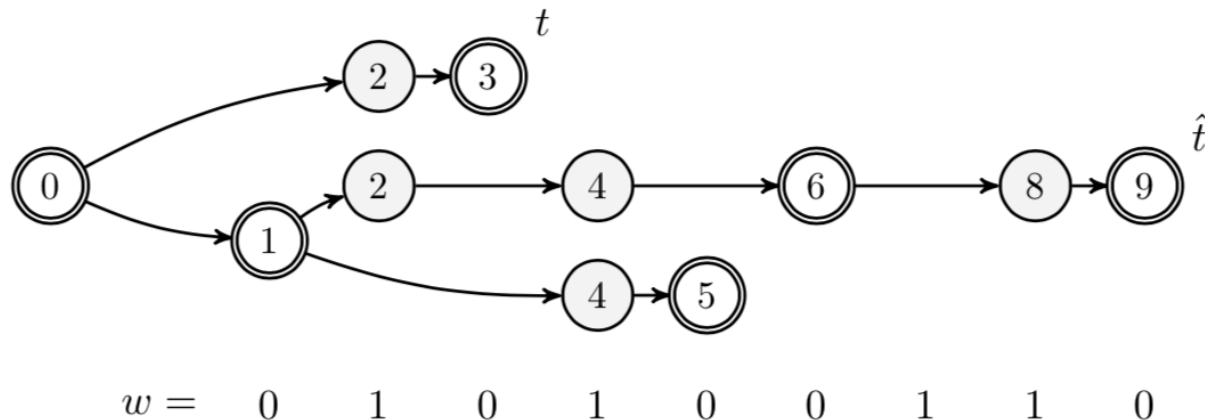
$$\Pr[\text{Hot}_{t+1} \mid \text{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Also: Chain Growth & Quality

Chain Quality: any sufficiently long section along a (viable) tine must contain an honest node with overwhelming probability.
(\rightarrow 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).

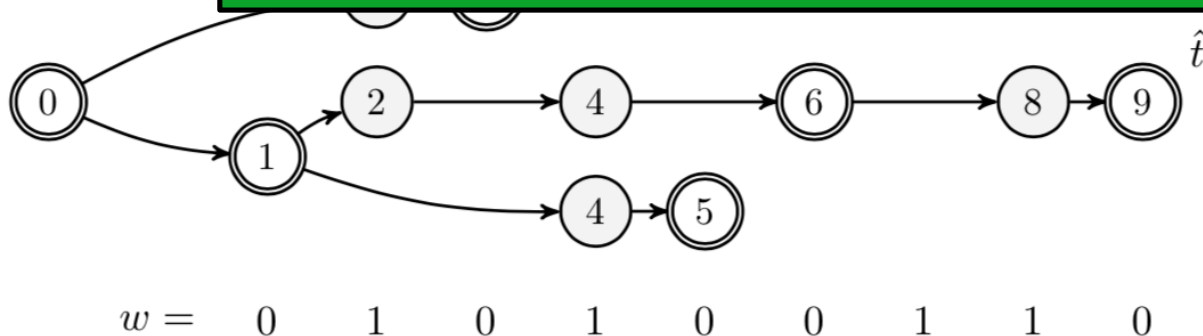


Also: Chain Growth & Quality

Chain Quality: any sufficiently long section along a (viable) line must contain an honest node with overwhelming probability
(\rightarrow Otherwise, $\#0$'s $<$ $\#1$'s)

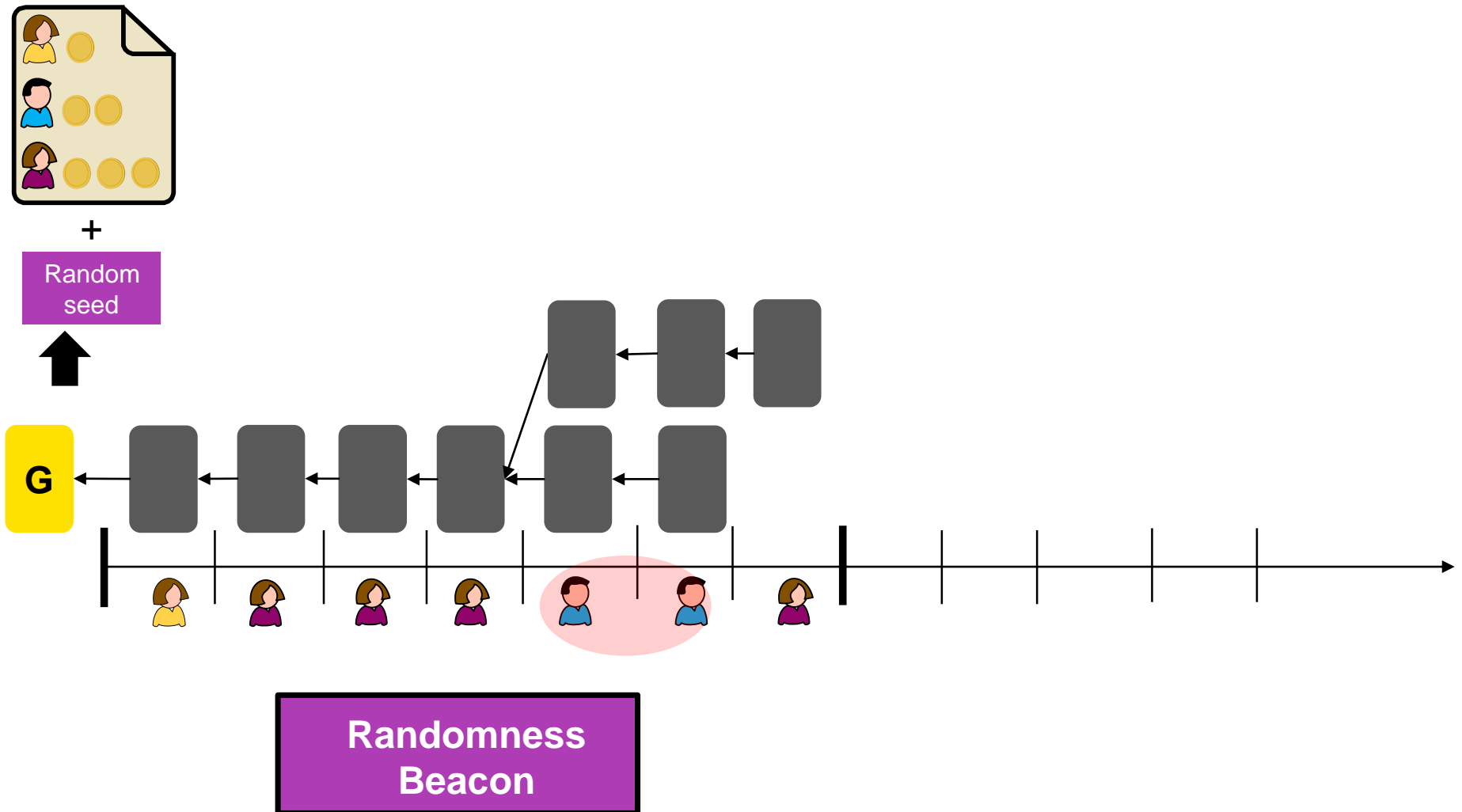
Chain Growth: growth is reflected

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

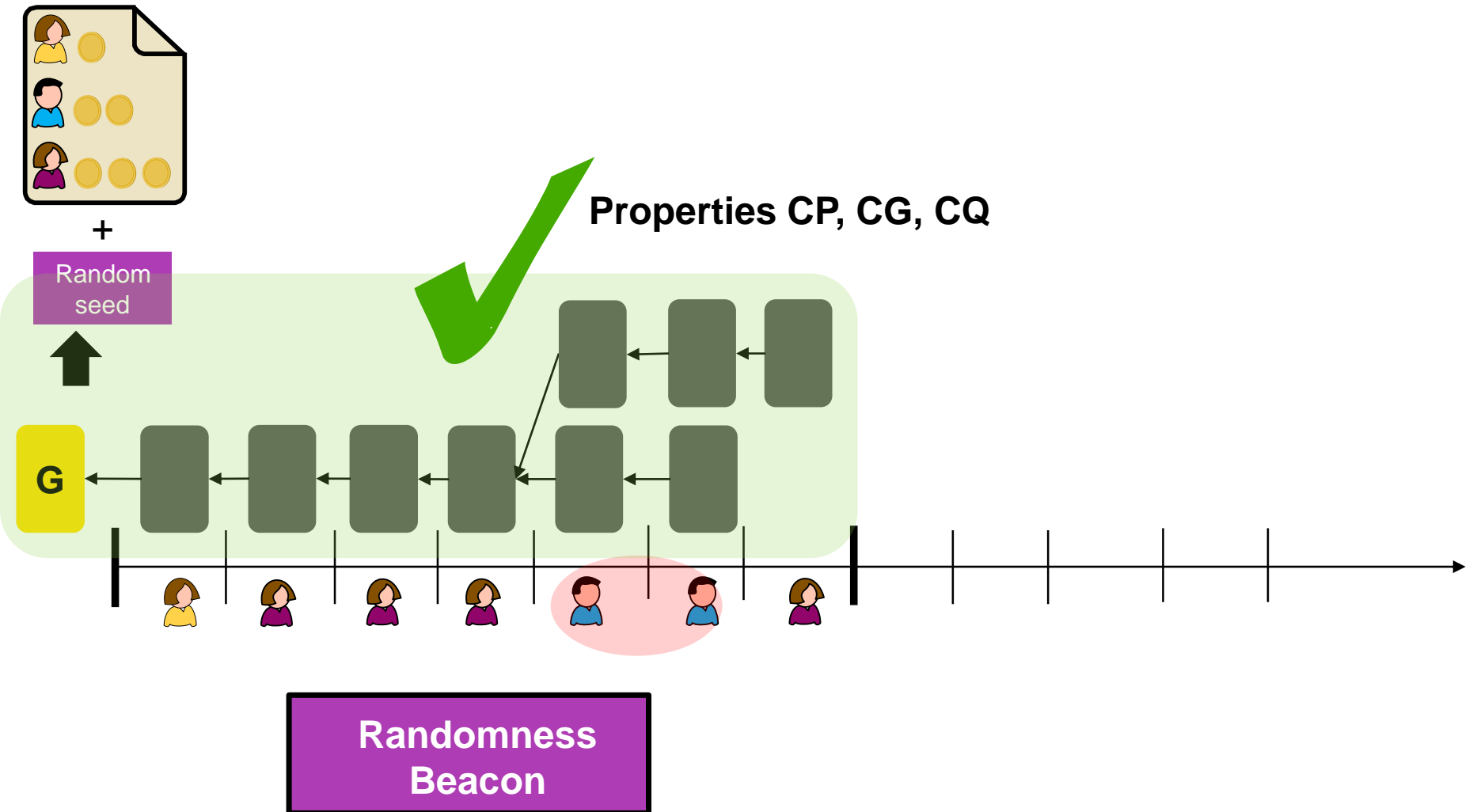


Lifting To Multiple Epochs

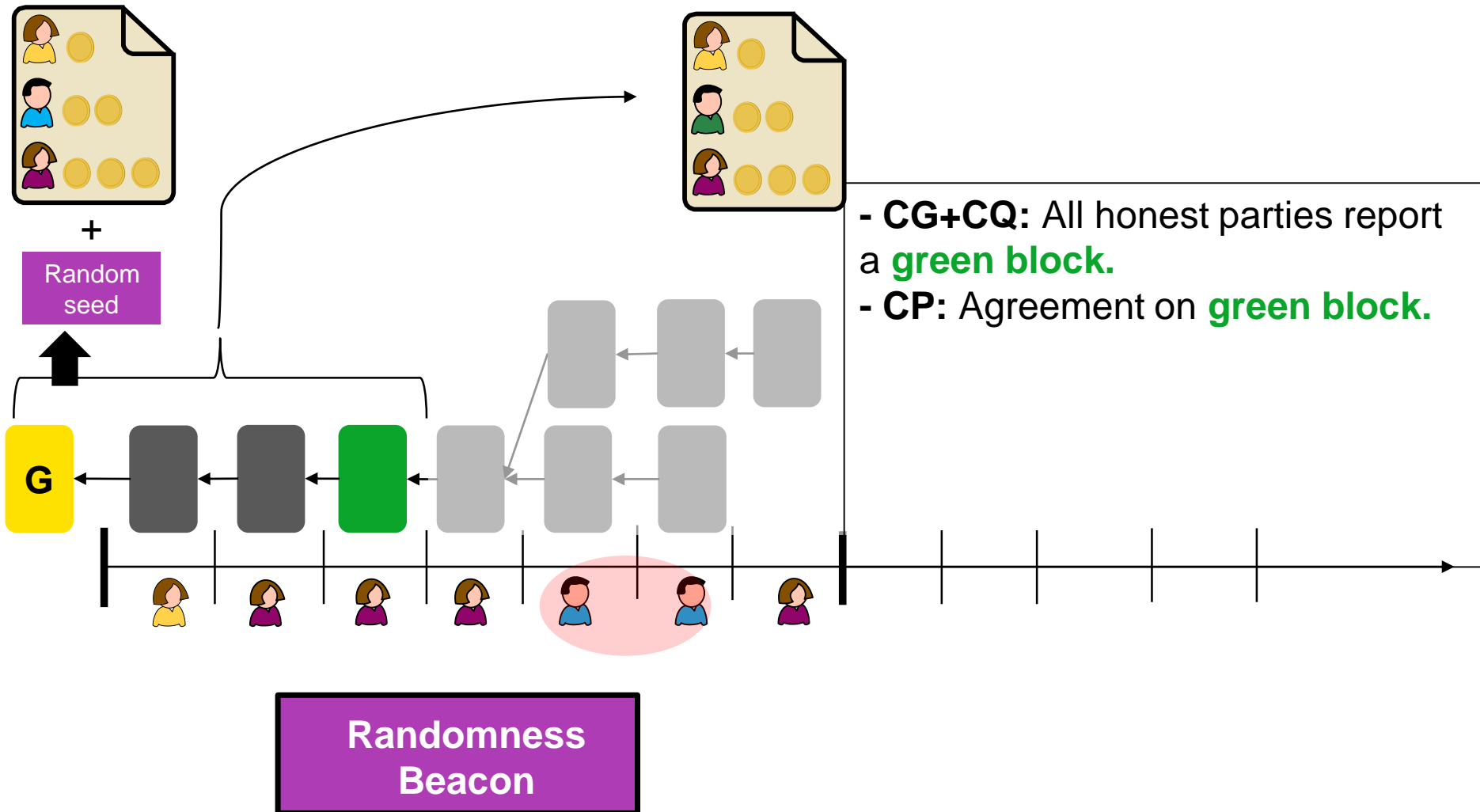
Lifting To Multiple Epochs



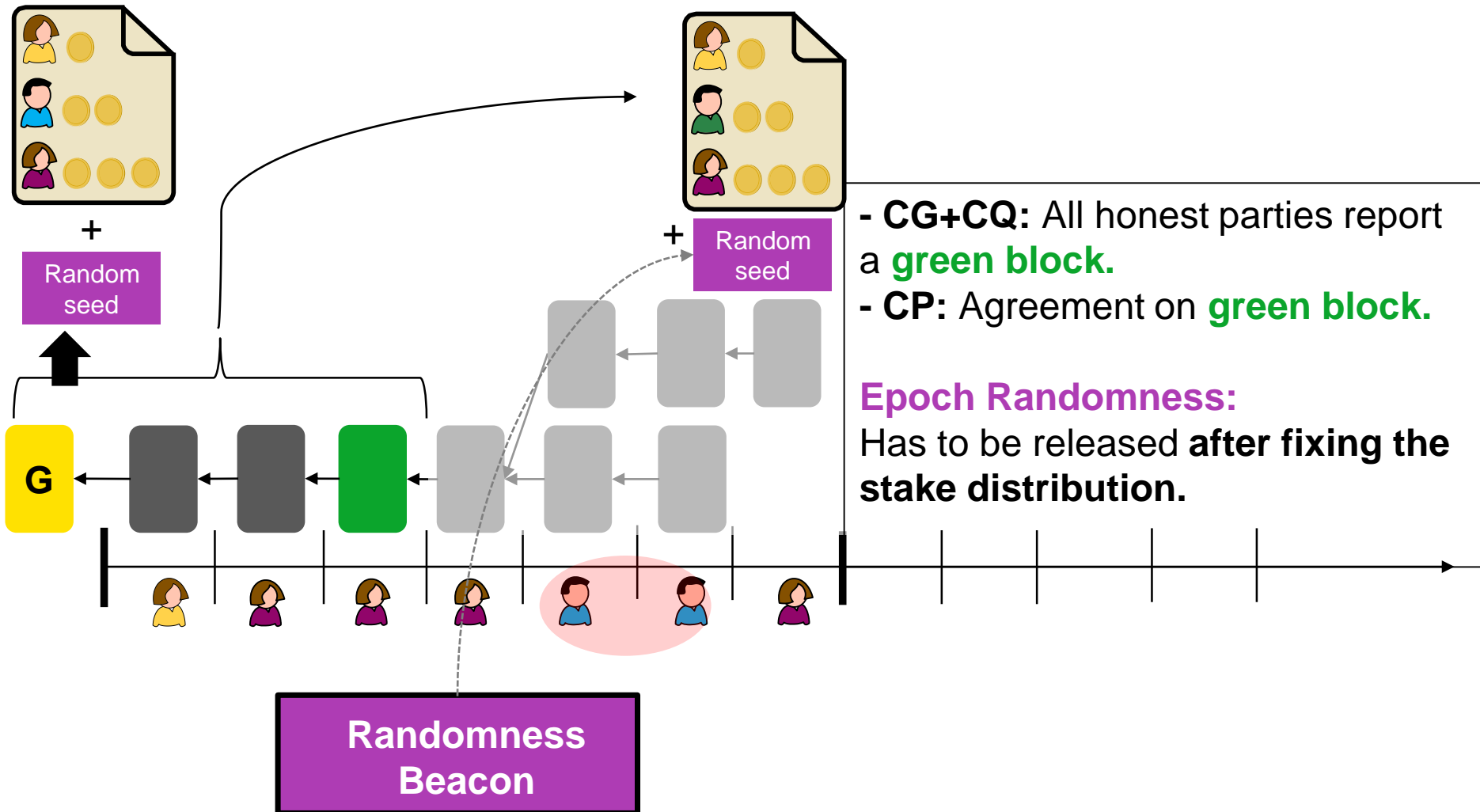
Lifting To Multiple Epochs



Lifting To Multiple Epochs



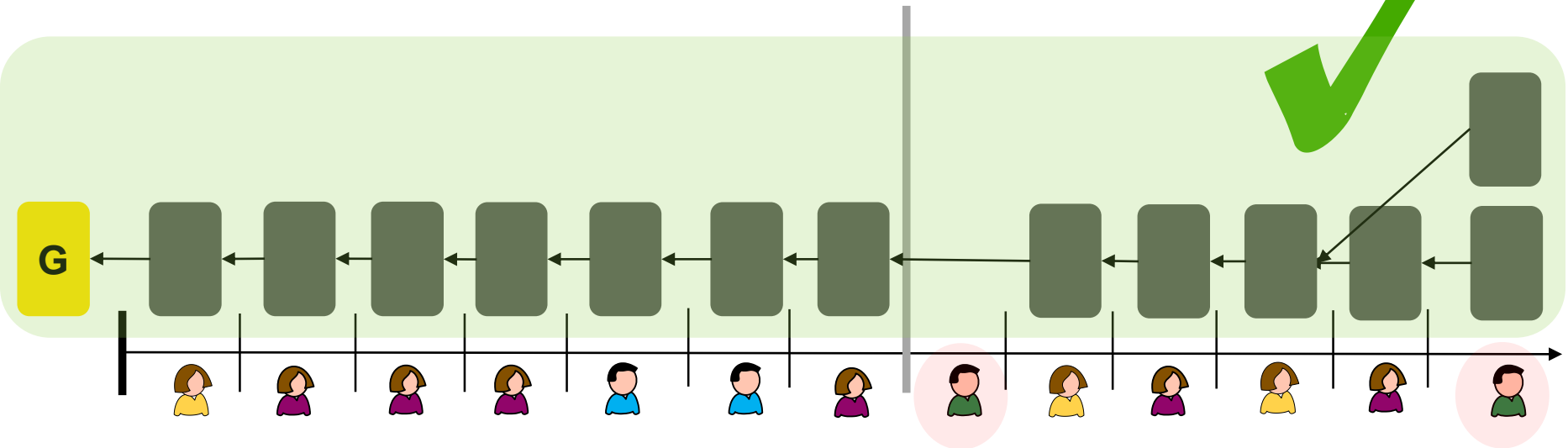
Lifting To Multiple Epochs



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

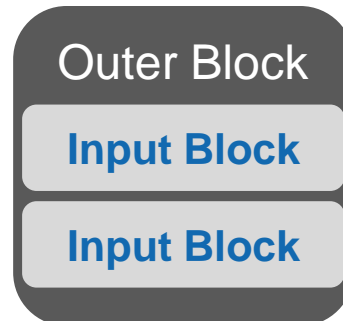
Key Idea:

Main effort is related to input contribution → Declare it to be a separate task.

Incentives

Solution: Input Endorsers

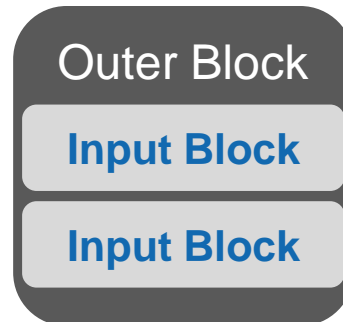
- **Each slot elects an additional stakeholder** (or a set of stakeholders) **to contribute inputs.**
 - Using a parallel lottery.
 - Like the 2-for-1 mechanism in PoW as in GKL analysis or Fruitchains.
- Endorsed inputs are permitted in the blockchain any time within a small window following and inclusive the slot that elects them.



Incentives

Solution: Input Endorsers

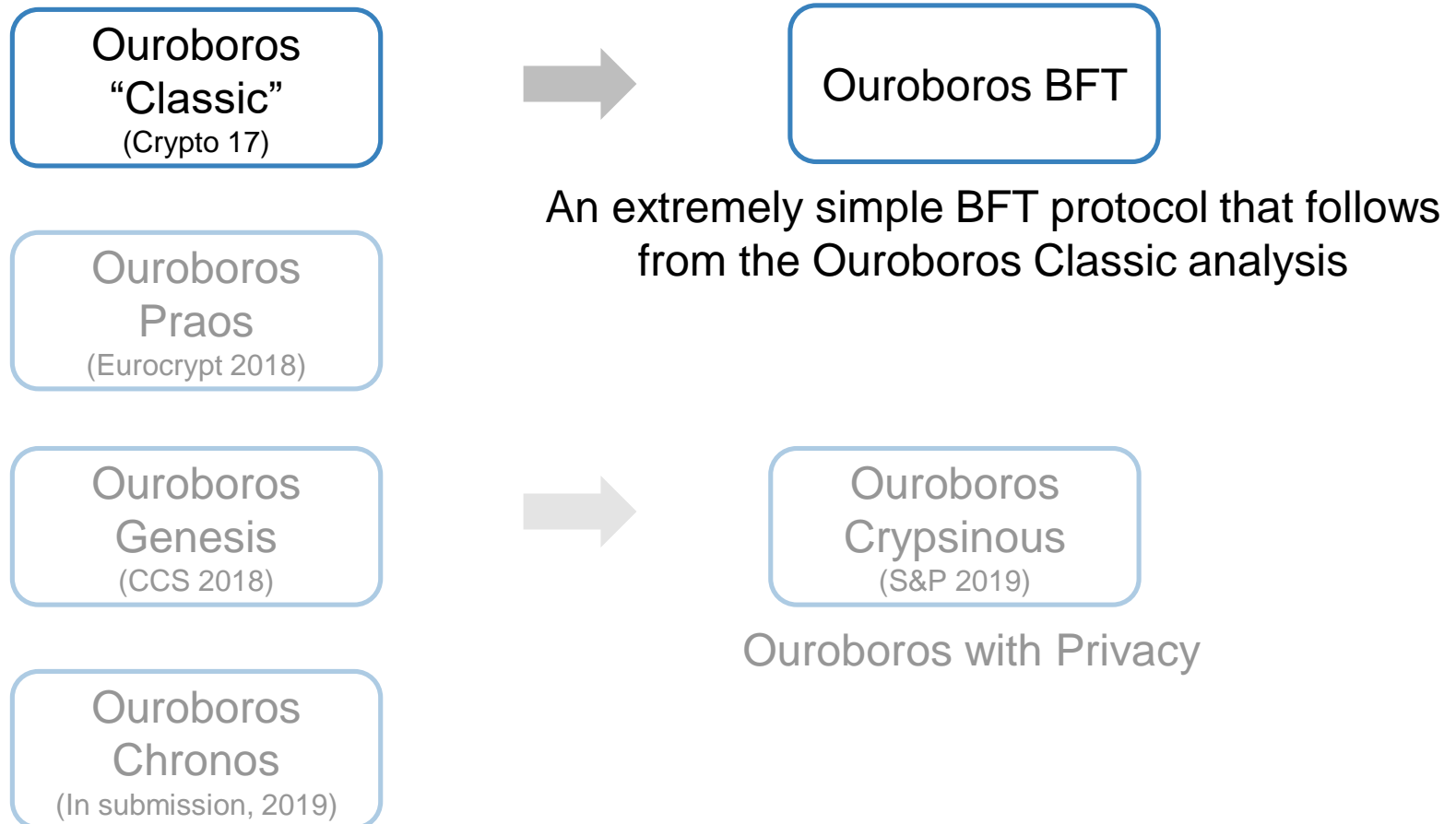
- **Each slot elects an additional stakeholder** (or a set of stakeholders) **to contribute inputs.**
 - Using a parallel lottery.
 - Like the 2-for-1 mechanism in PoW as in GKL analysis or Fruitchains.
- Endorsed inputs are permitted in the blockchain any time within a small window following and inclusive the slot that elects them.



→ Overall #Input blocks proportional to stake.

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

Ouroboros BFT



Ouroboros BFT

Ouroboros
“Classic”
(Crypto 17)



Ouroboros BFT

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

Ouroboros
Praos

Central Observation:

A characteristic string (assume binary) with Hamming-weight less than $1/3$ is not forkable.

Ouroboros with Privacy

Ouroboros
Chronos
(In submission, 2019)

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)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

Ouroboros

+ **Adaptive Adversaries**

+ Network Delay (“asynchronous”)

Stronger cryptographic primitives needed:

- To enable private lottery
- To fully mitigate adaptive corruptions

Ouroboros
Chronos

(In submission, 2019)

+ 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)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

Ouroboros
Praos
(Eurocrypt 2018)

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

Ouroboros
Genesis
(CCS 2018)

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

Newcomers should be able to join the system without the extra help of existing parties.

and assumption.
ne from genesis

A setting

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

Genesis Security Proof

+ Only based on same-speed assumption.
+ Bootstrapping state and time 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.Gen, VRF.Eval, VRF.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: **Allow private leader-election** and thereby a more realistic attacker model (mitigate adaptive attacks).

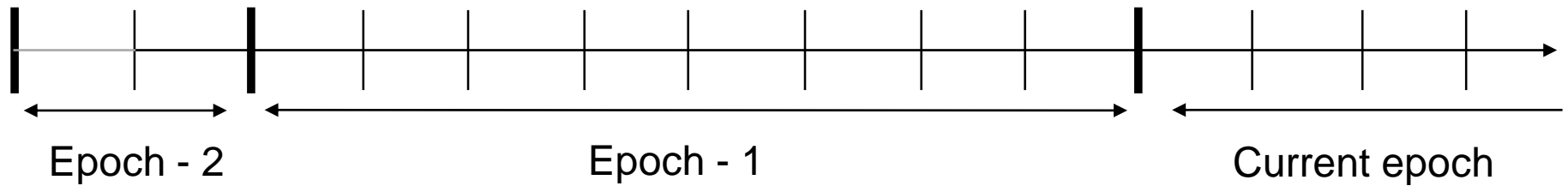
Cryptographic Tools to Protect against Adaptive Adversaries

Key evolving signature scheme (KES)

(KES.Gen, KES.Sign, KES.Update, KES.Verify)

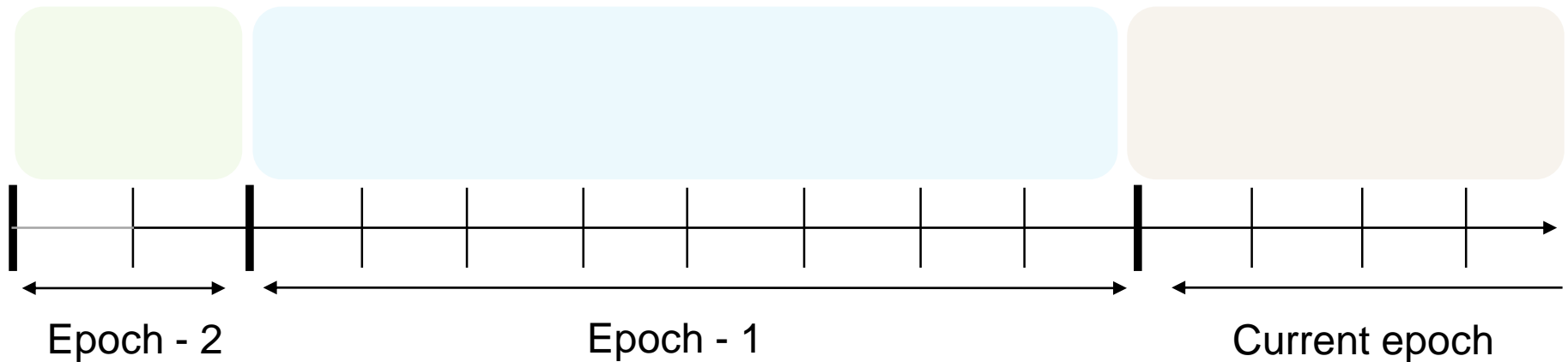
- Operates as normal signature scheme (unforgeable)
- Key updates: All values signed “in the past” remain unforgeable even if party gets corrupted after the update.
- Purpose: **Protect previous actions** and thereby allow realistic corruption model (tolerate “immediate corruptions”).

Ouroboros Praos/Genesis: Basic Operation

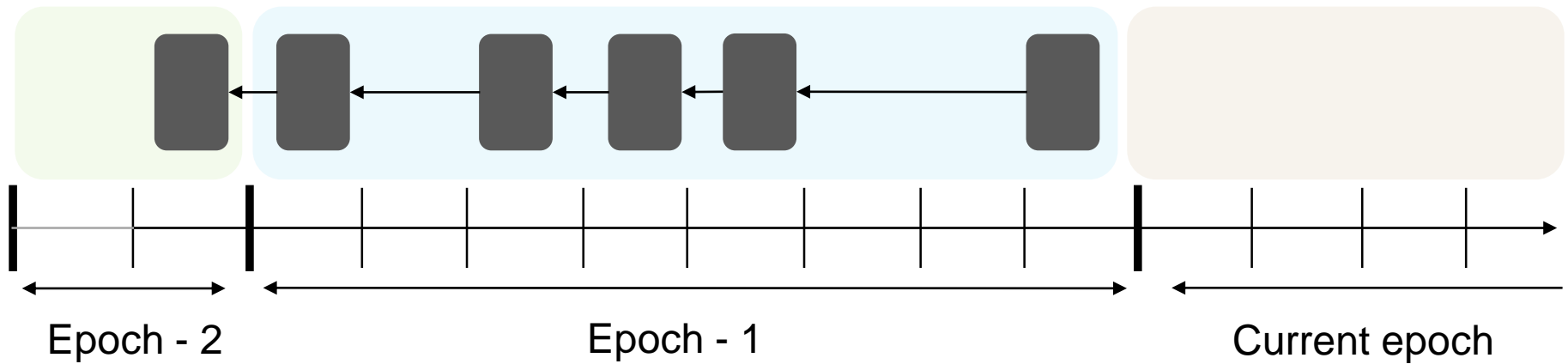


Ouroboros Praos/Genesis: Basic Operation

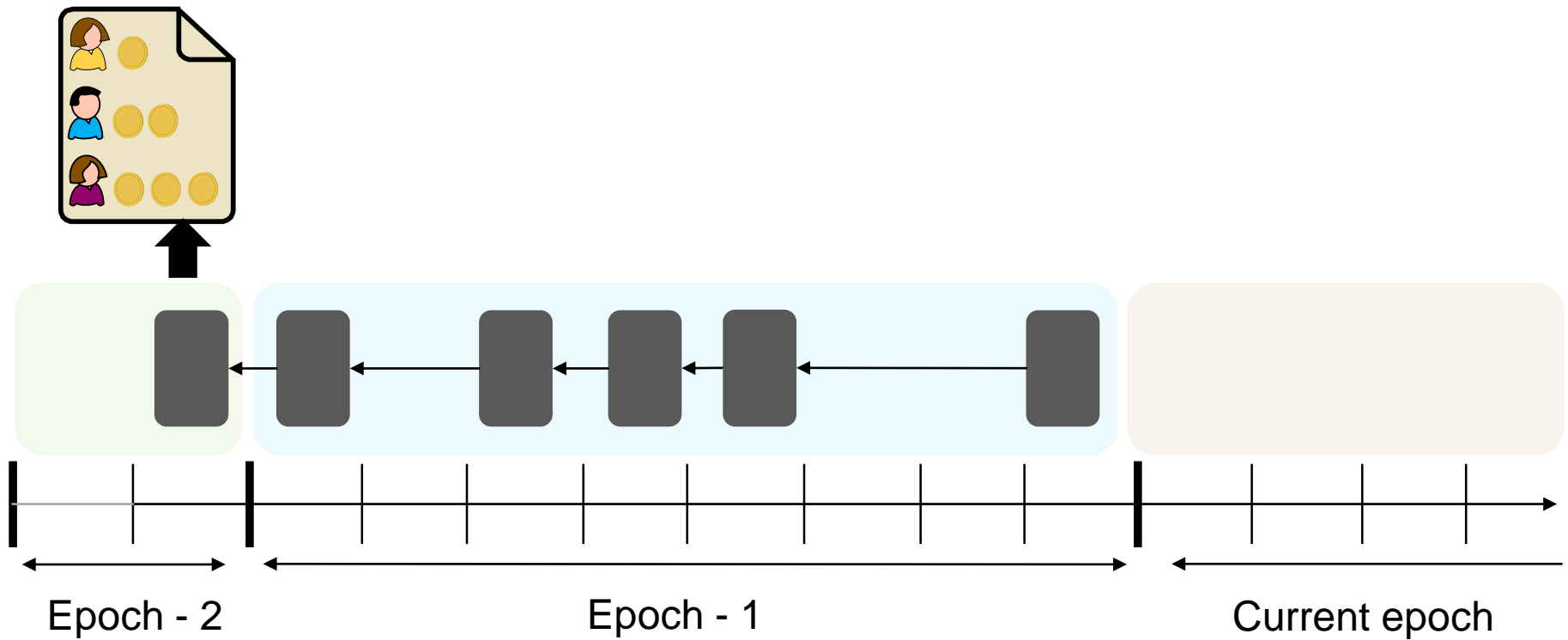
- In each slot, each party evaluates slot-leadership.
Private election, proportional to stake, including recent randomness from the chain
- A slot leader extends a chain by creating the block for this slot.



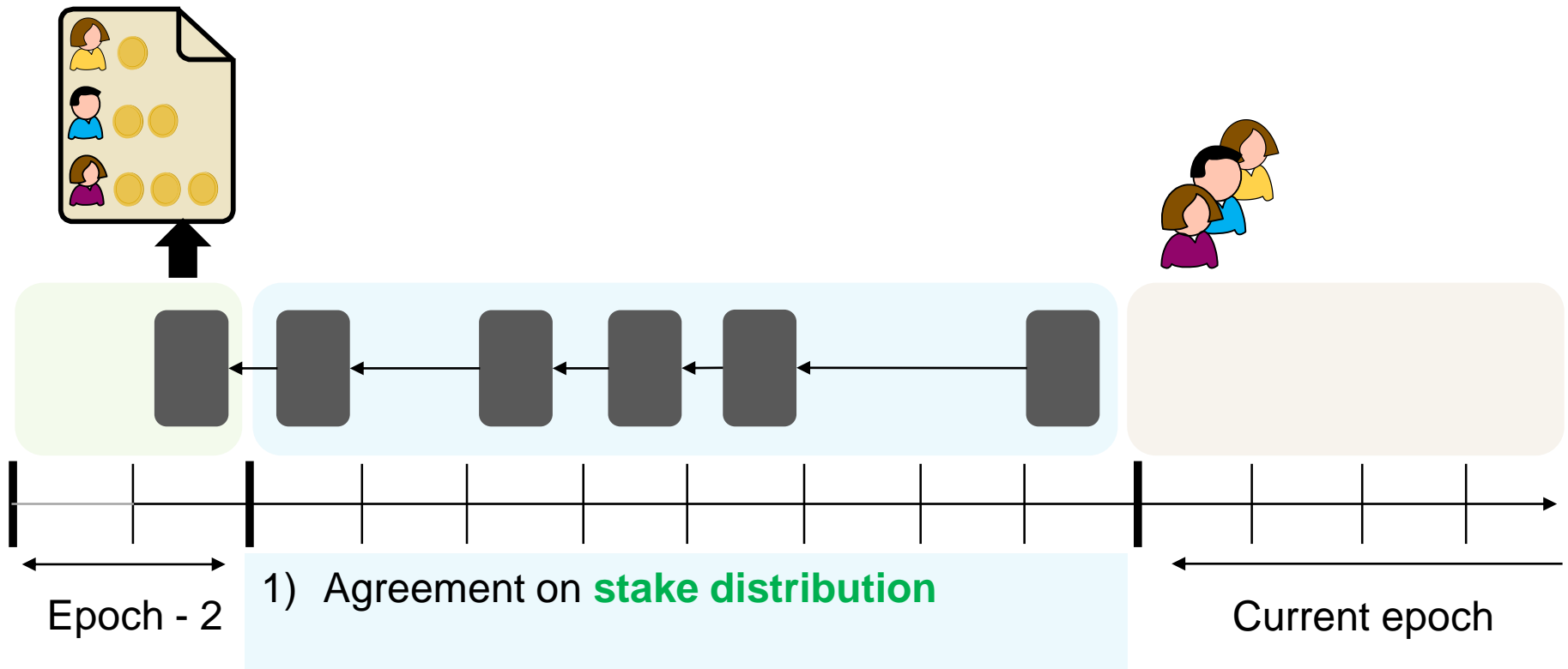
Ouroboros Praos/Genesis: Basic Operation



Ouroboros Praos/Genesis: Basic Operation



Ouroboros Praos/Genesis: Basic Operation

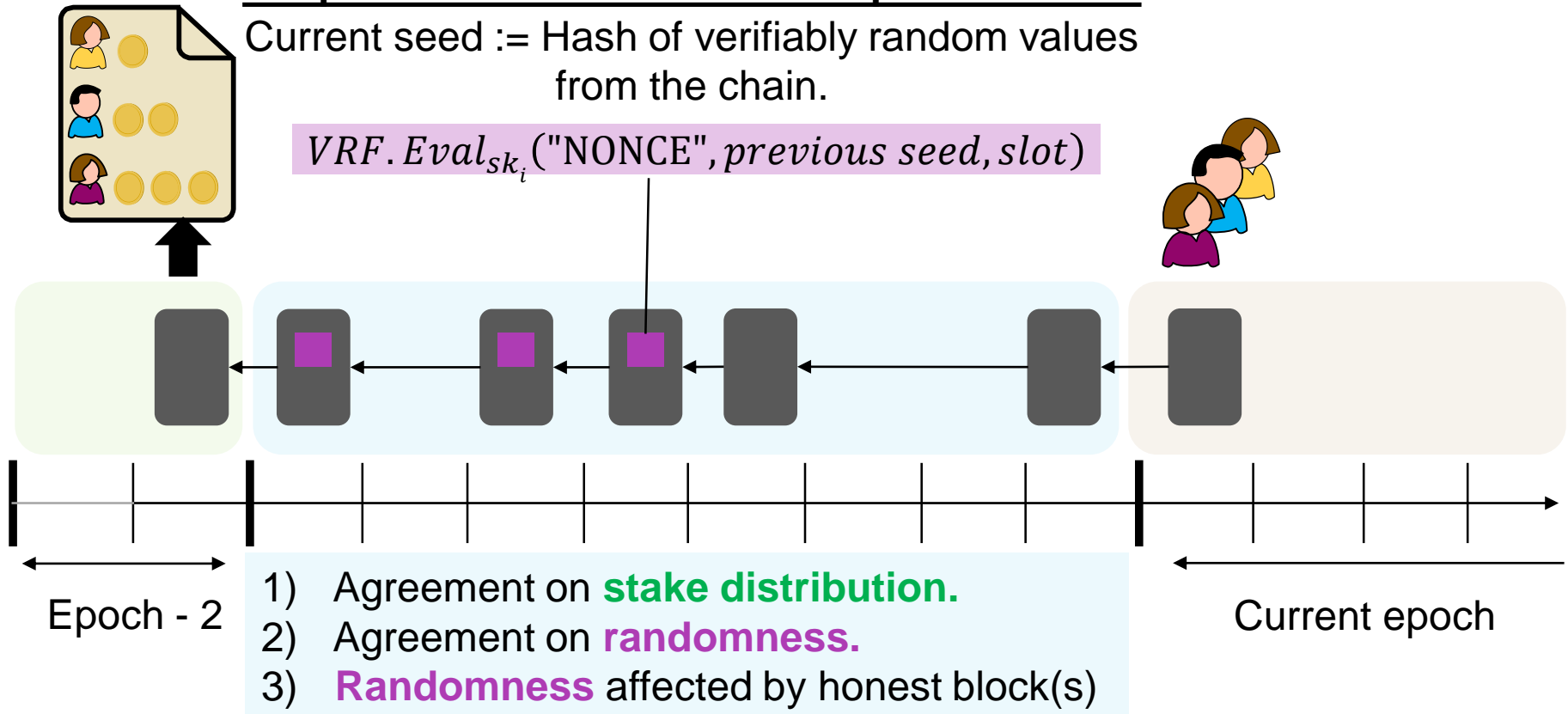


Ouroboros Praos/Genesis: Basic Operation

Simple Randomness-Beacon Implementation:

Current seed := Hash of verifiably random values from the chain.

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

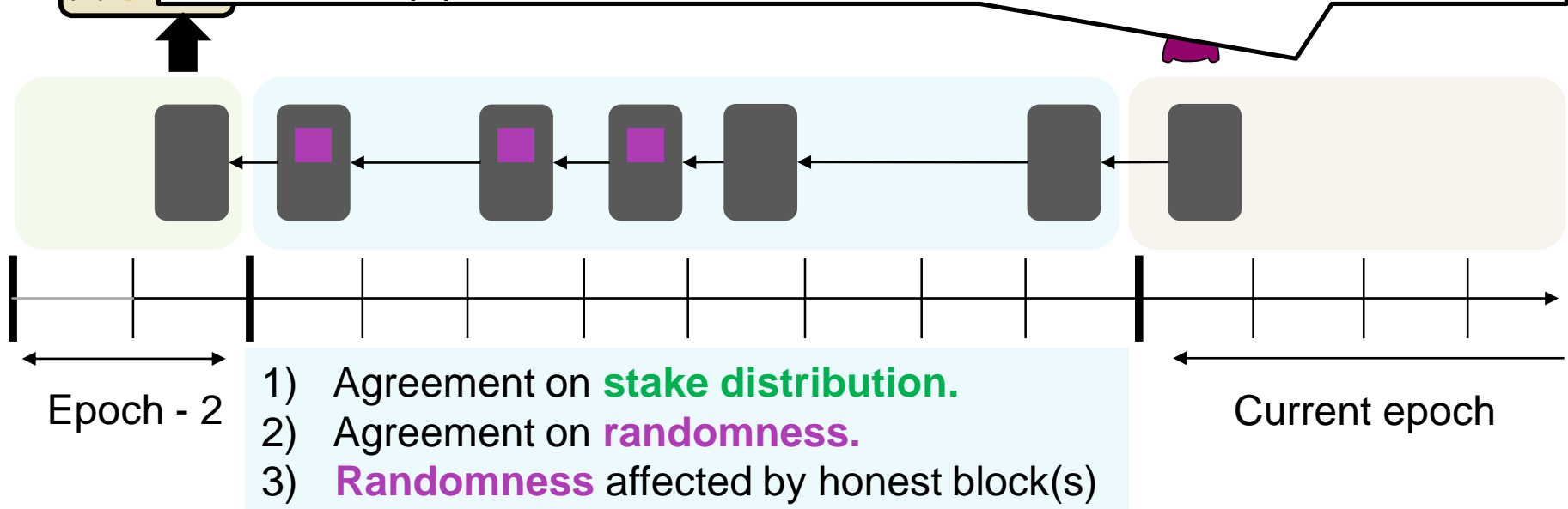


Ouroboros Praos/Genesis: Basic Operation

Lottery in each slot:

A party i is leader if and only if $VRF.Eval_{sk_i}(\text{"TEST", } \textcolor{violet}{seed}, \text{slot}) < T(stake_i)$

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

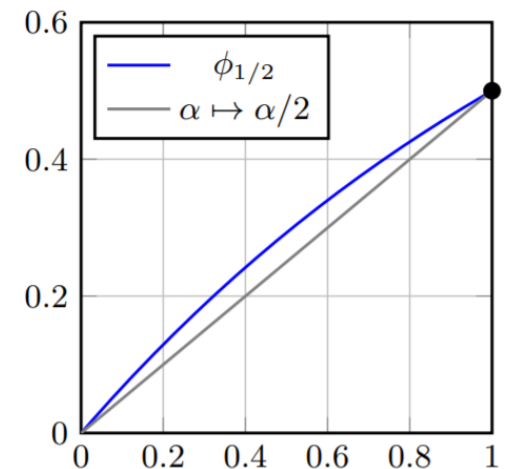


Ouroboros Praos/Genesis: Details on Leader Election

$$VRF.Eval_{sk_i}(\text{"TEST"}, \text{seed}, slot) < T(stake_i)$$

$$\rightarrow T(stake_i) = 2^{\ell_{VRF}} \varphi_f(rel.stake_i)$$

$$\rightarrow \varphi_f(x) = 1 - (1 - f)^x$$

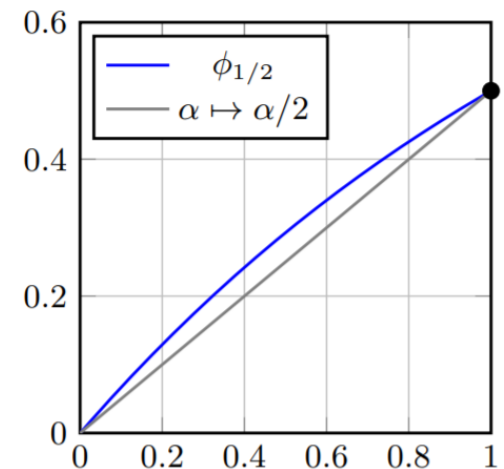


Ouroboros Praos/Genesis: Details on Leader Election

$$VRF.Eval_{sk_i}(\text{"TEST"}, \text{seed}, slot) < T(stake_i)$$

$$\rightarrow T(stake_i) = 2^{\ell_{VRF}} \varphi_f(rel.stake_i)$$

$$\rightarrow \varphi_f(x) = 1 - (1 - f)^x$$



Some remarks:

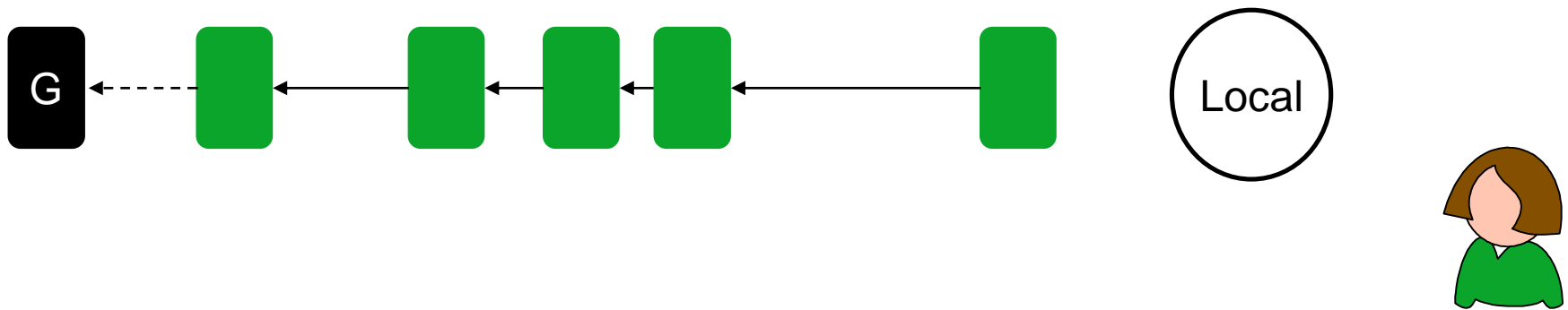
1.) Active slot coefficient: $\varphi_f(1) = f$; slot empty with prob. $1 - f$.

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

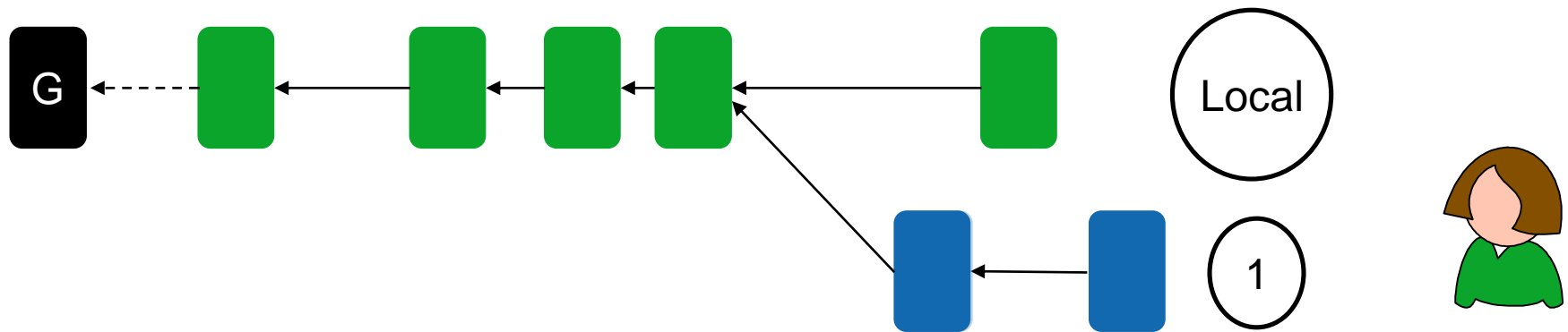
→ Probability of leadership independent of distribution to addresses.

→ The concave (and subadditive as $\varphi_f(0) = 0$) property eases the analysis.

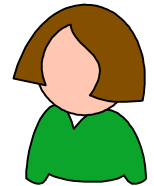
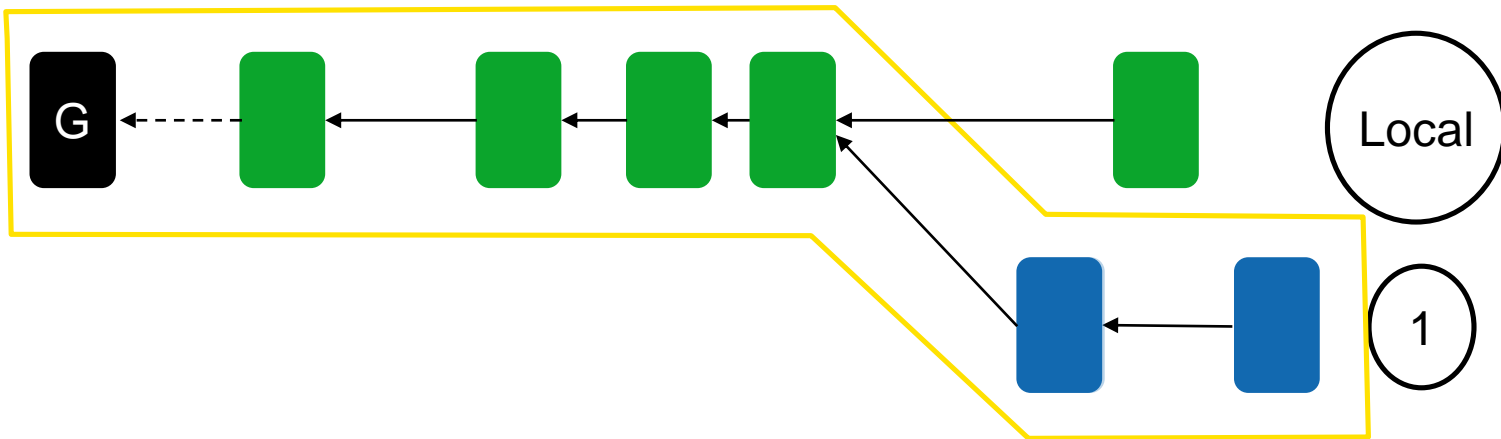
Recall: Chain-Selection Rule



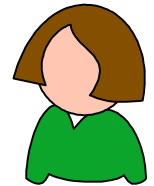
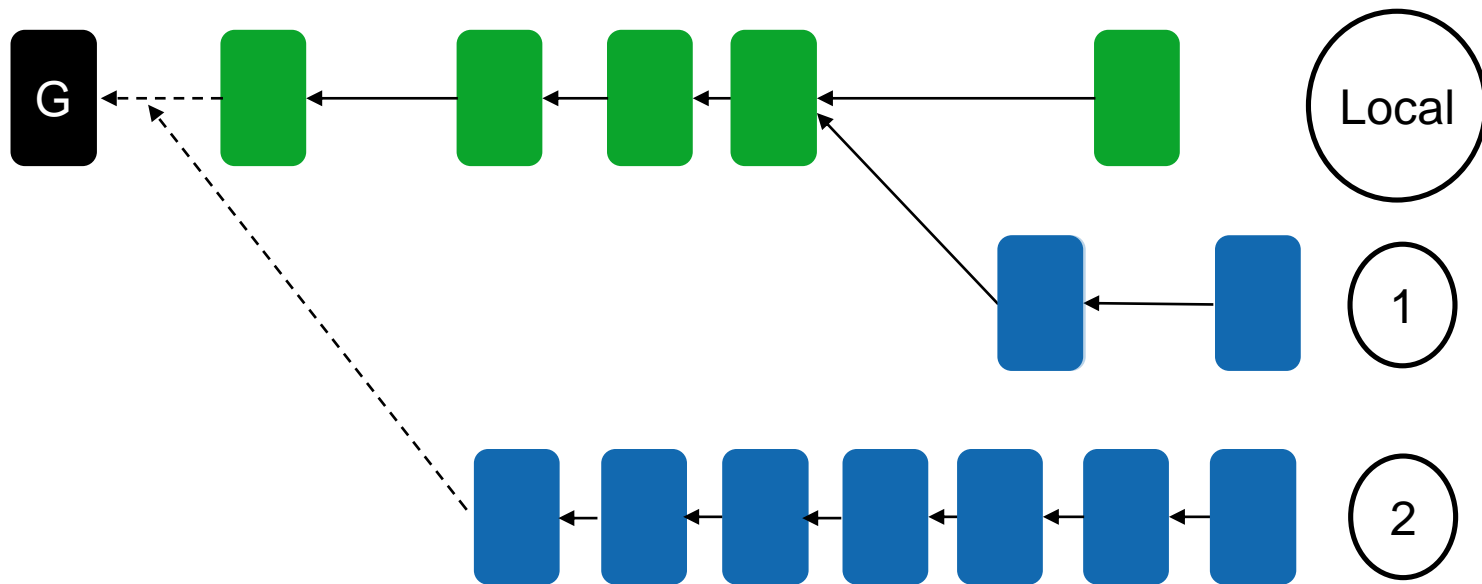
Recall: Chain-Selection Rule



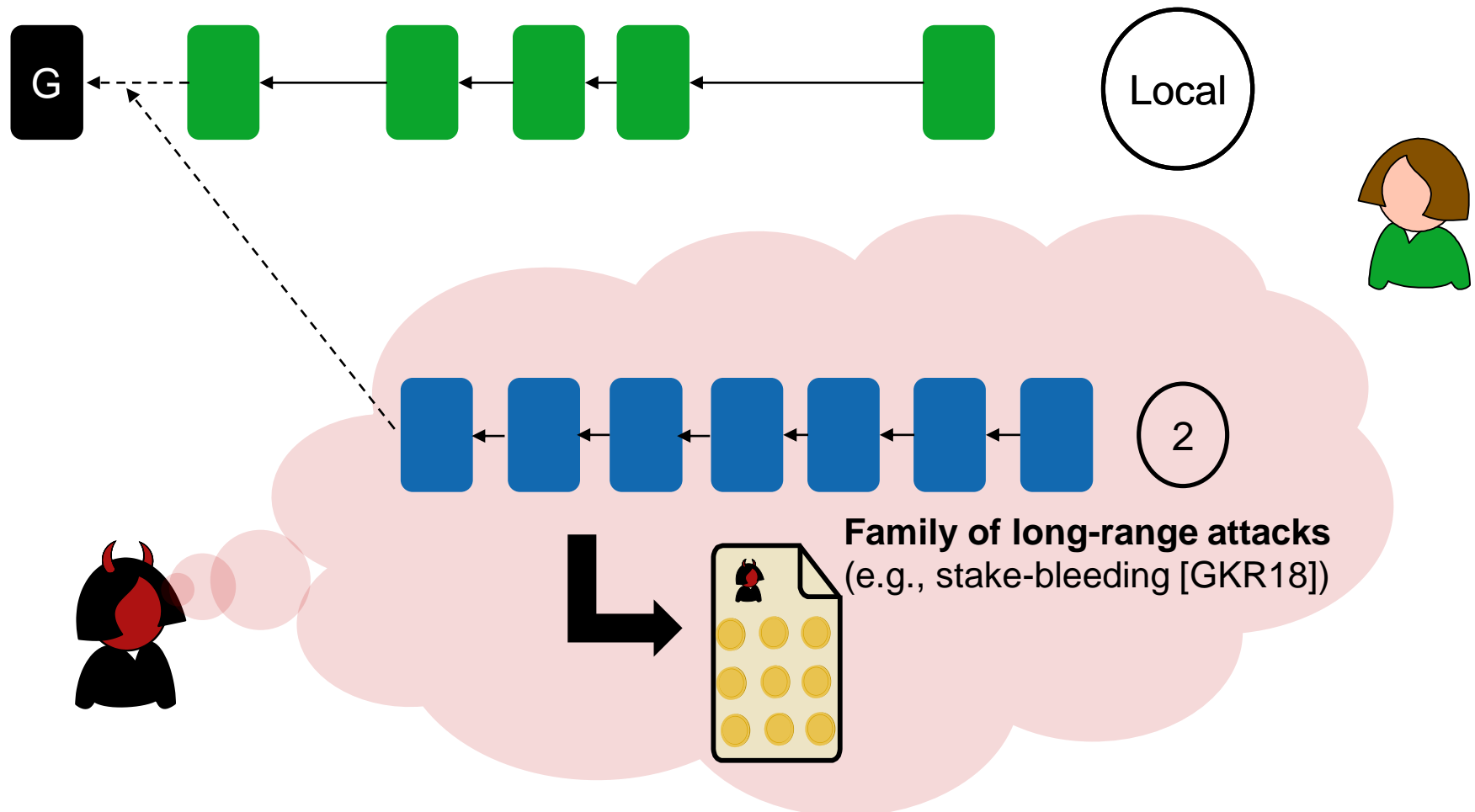
Recall: Chain-Selection Rule



Recall: Chain-Selection Rule



Attention: Longest Chain Rule Does not Work



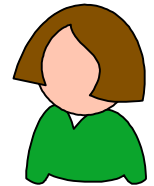
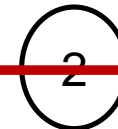
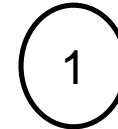
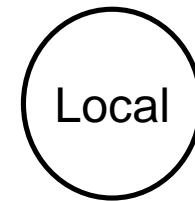
Recall: Chain-Selection Rule

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)

Semi-adaptive adversaries, synchrony
Strong mathematical framework

Ouroboros
Praos
(Eurocrypt 2018)

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

Ouroboros
Genesis
(CCS 2018)

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

Ouroboros
Chronos
(In submission, 2019)

+ 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:

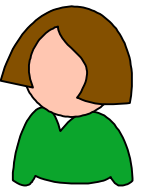
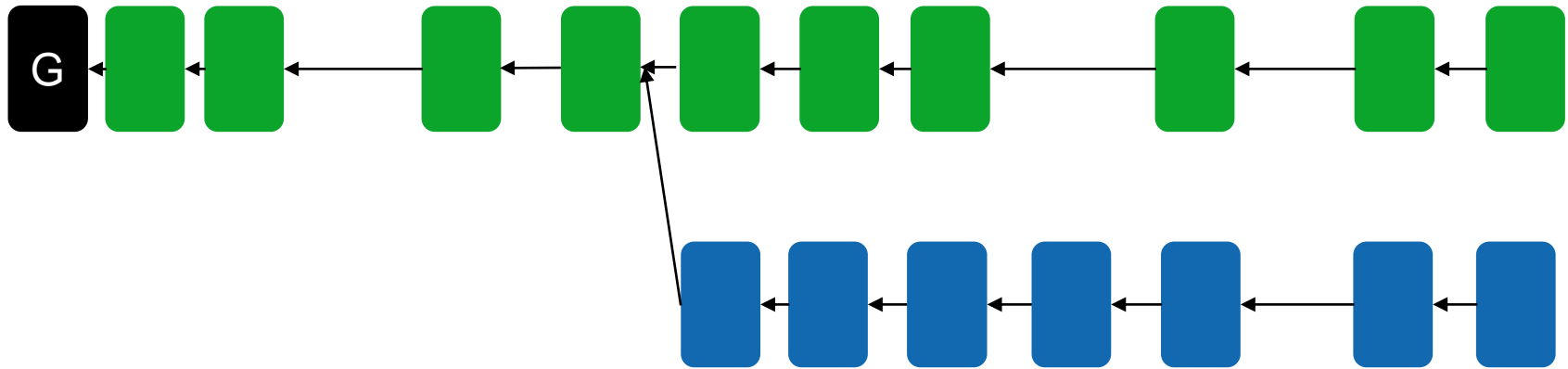
- 1.) Online parties **maintain a moving checkpoint**
→ Joining parties need advice
- 2.) A **fixed and known lower bound on participation**
→ No flexible participation, protocol might be stalled.

We do not require either of these!

Thanks to a **more involved Chain-Selection Rule**

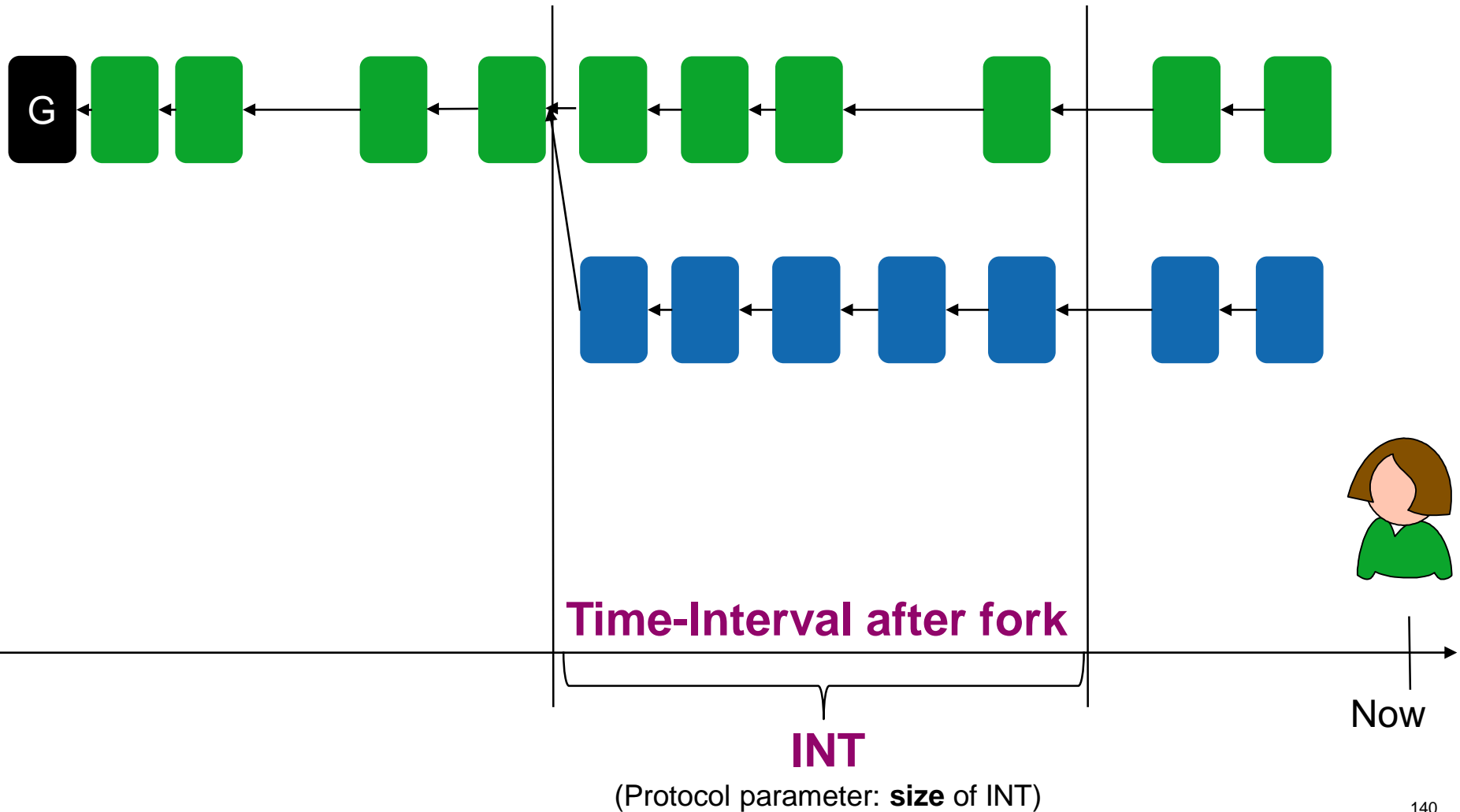
The Genesis Chain-Selection Rule

The Genesis Chain-Selection Rule



Now

The Genesis Chain-Selection Rule



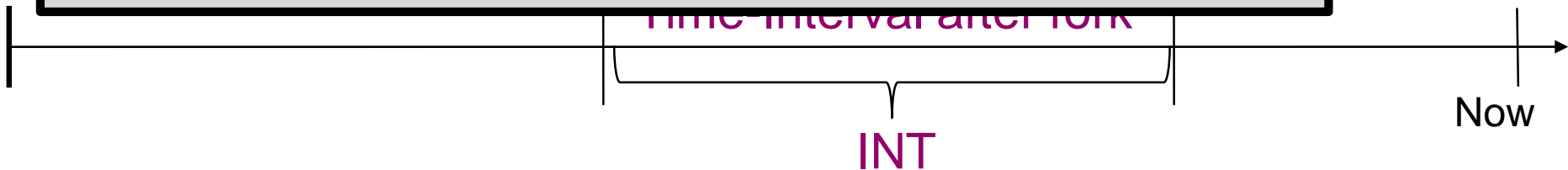
The Genesis Chain-Selection Rule

Genesis Rule:

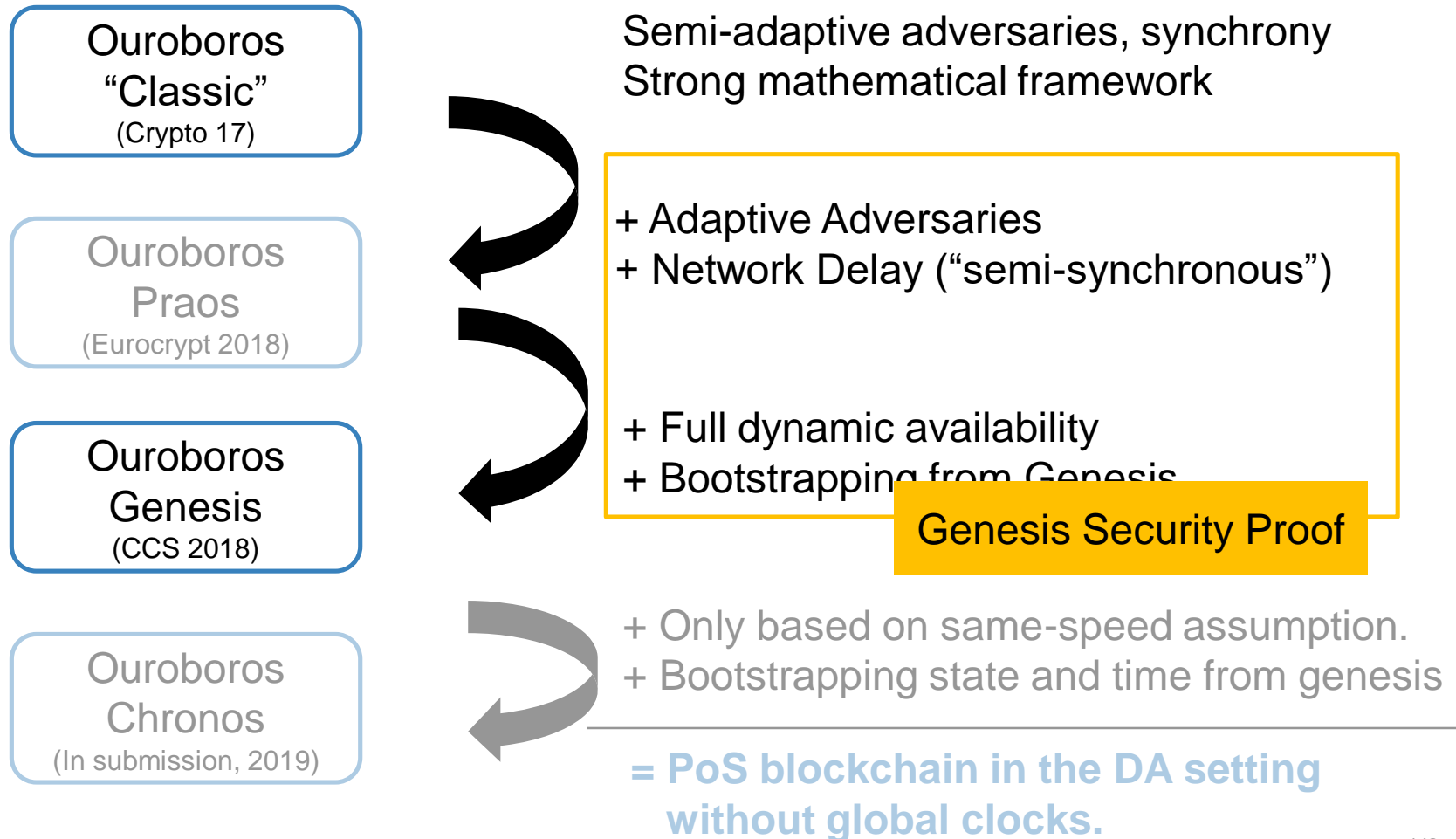
Adopt a valid **new chain**...

- 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



Roadmap of Security Proof of Genesis

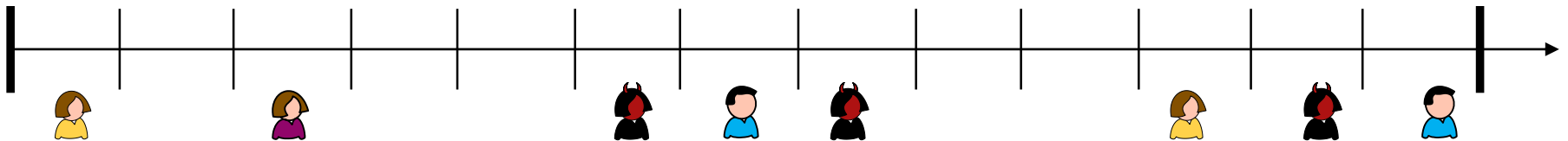
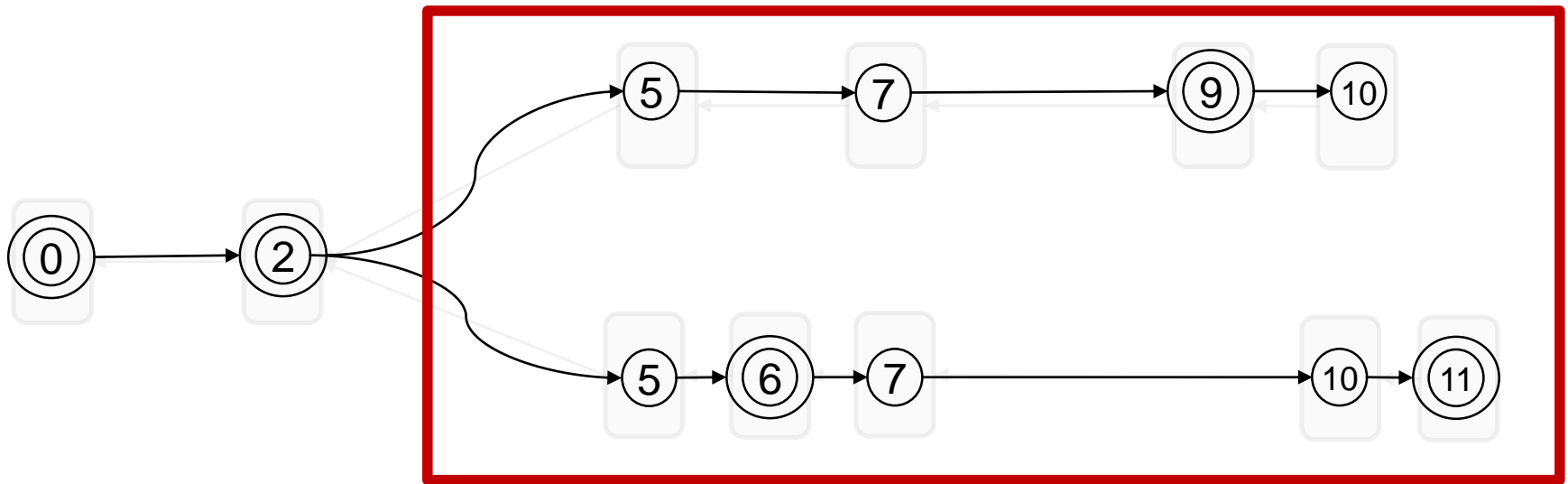
- Security of Ouroboros Genesis with the old chain selection rule (=Praos) and **dynamic participation** but no newly joining parties.
- Security for joining parties: **new Genesis rule in action.**

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:

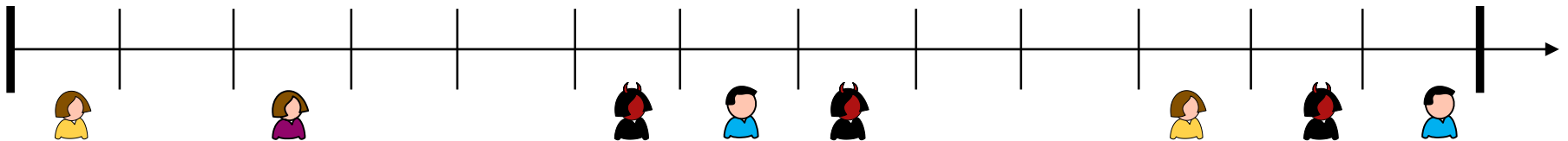
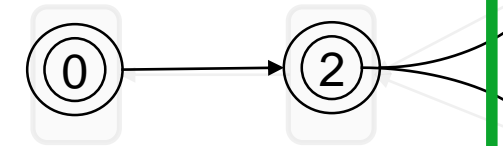


Security under dynamic Participation

- Recall the Fork abstraction:

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

Majority of **active** stake is honest.

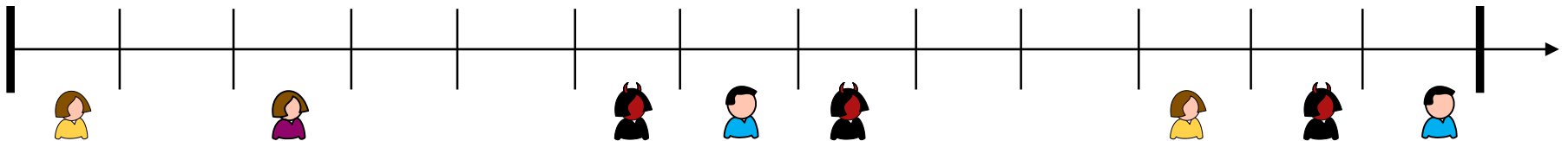
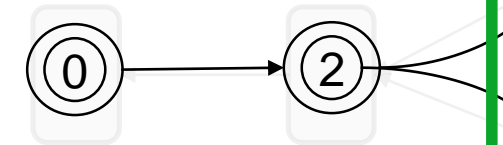


Security under dynamic Participation

- Recall the Fork abstraction:

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

$$\alpha \cdot (1 - f)^{\Delta+1} \geq (1 + \epsilon)/2$$



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

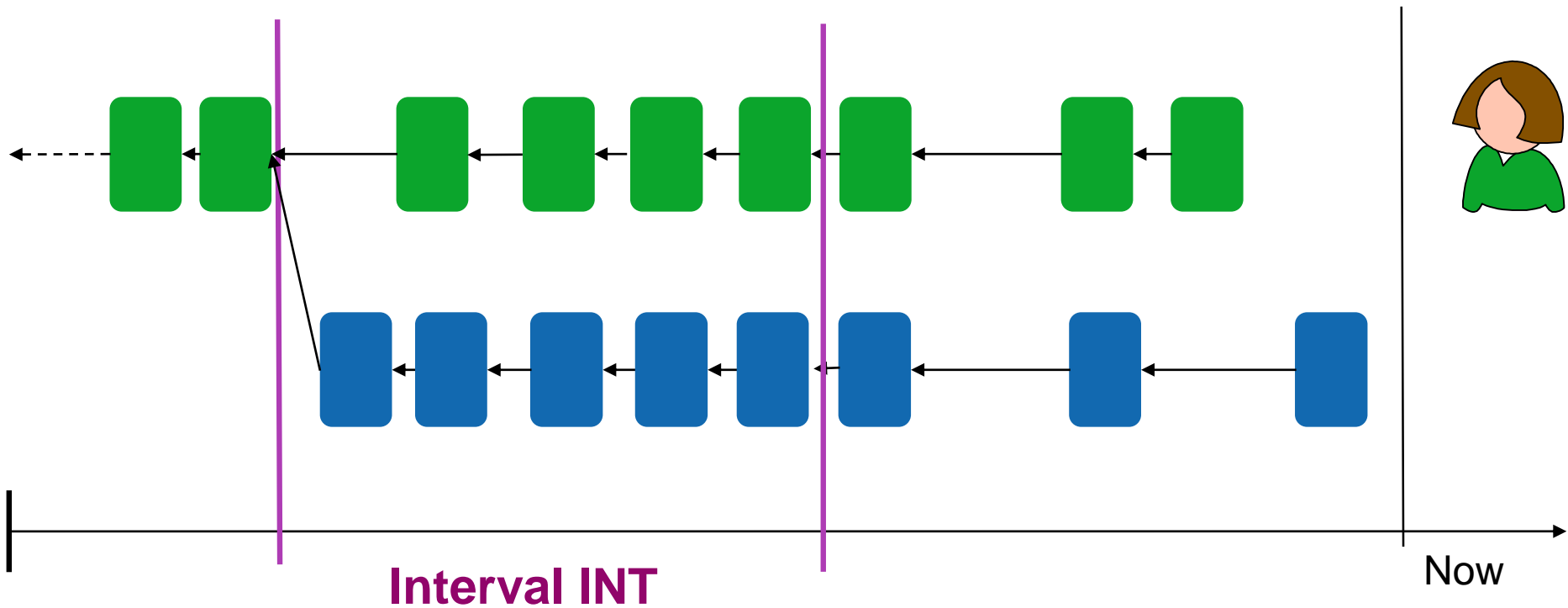
Claim 1:

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).

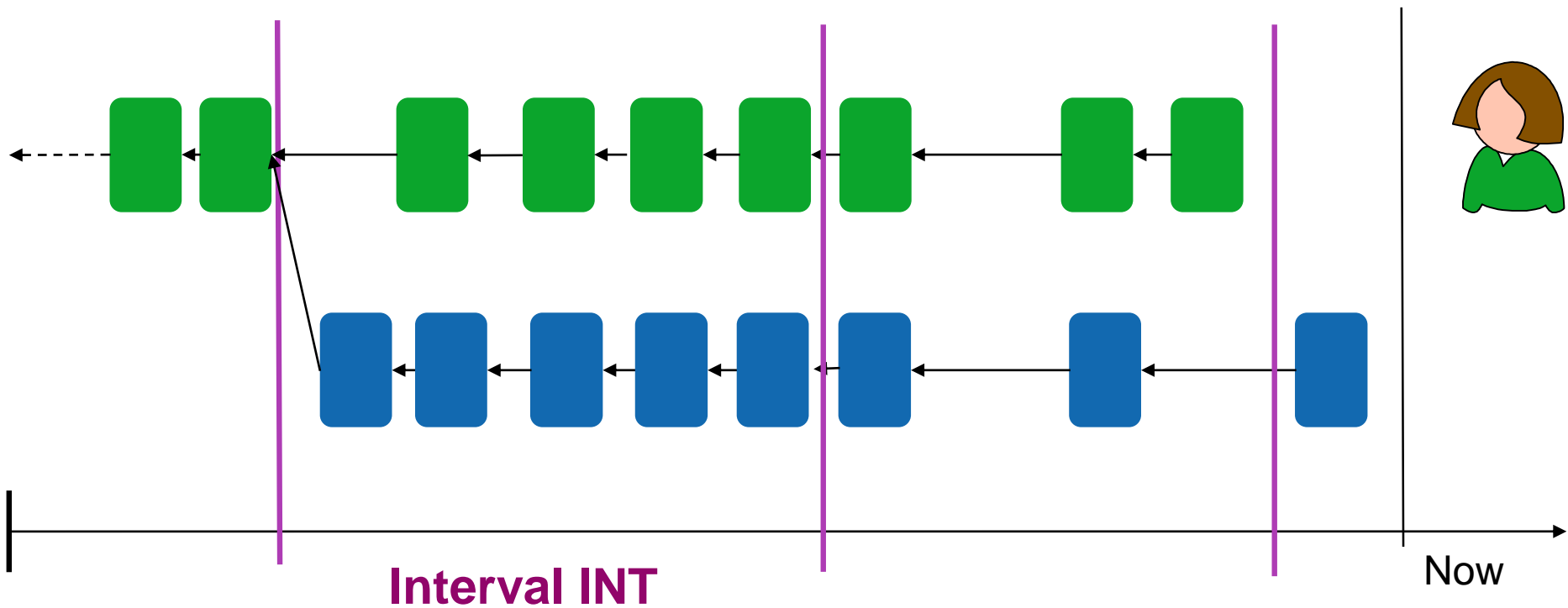
Claim 2:

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.

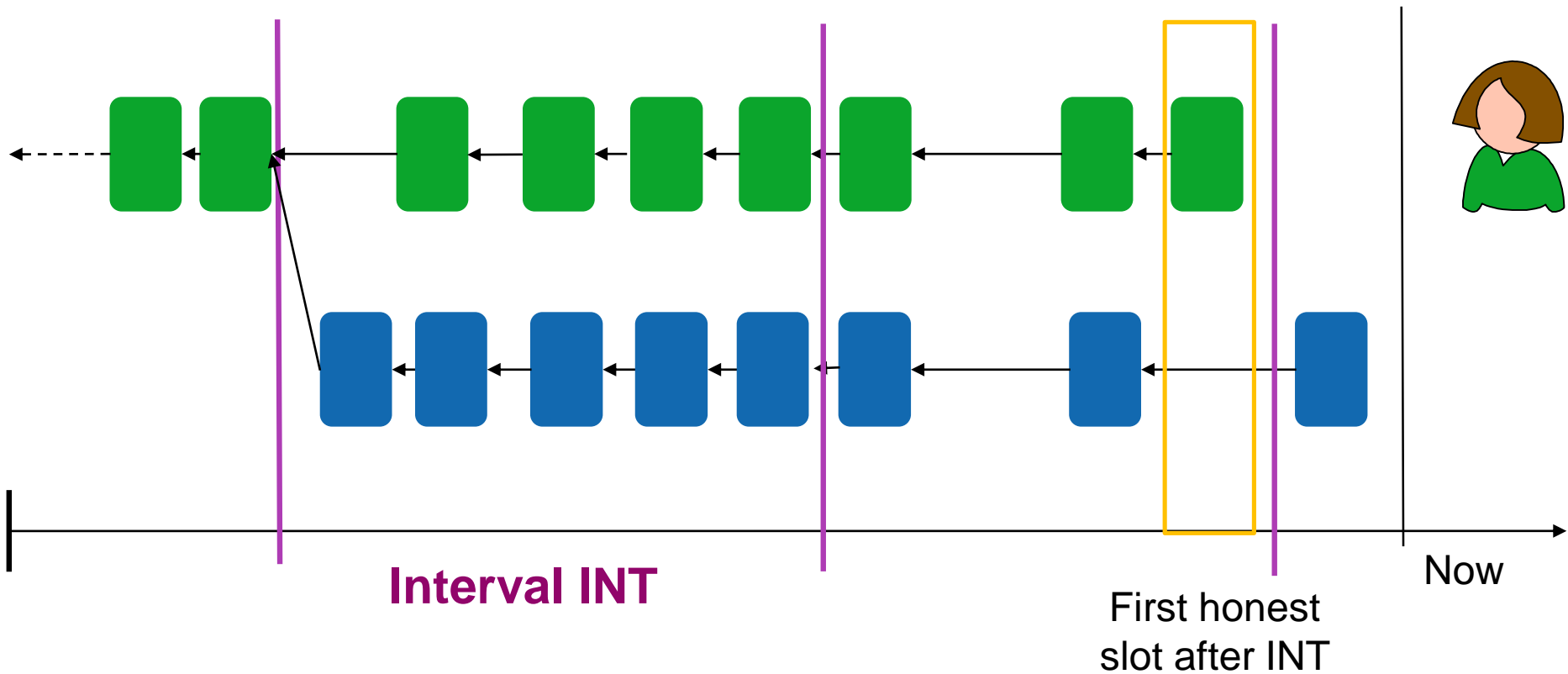
Claim 1 – Proof Idea



Claim 1 – Proof Idea

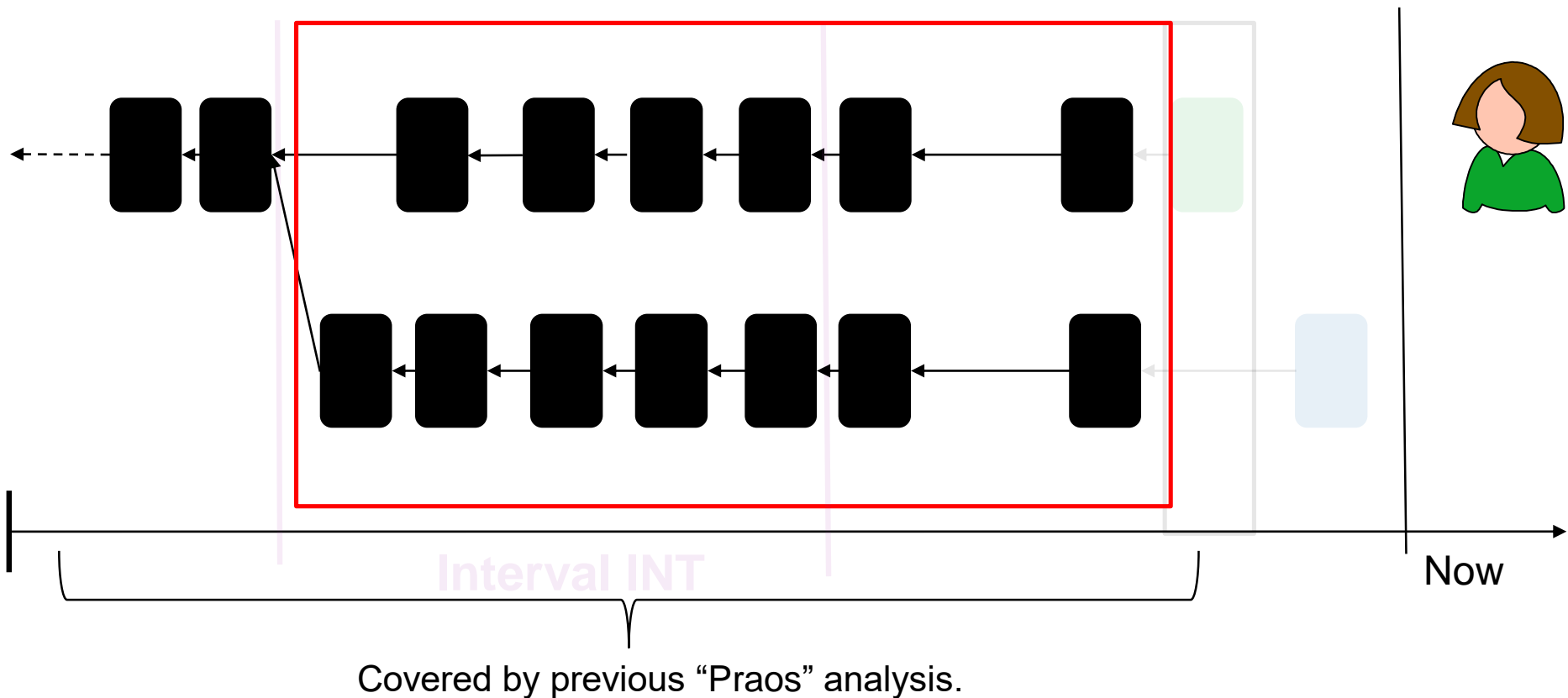


Claim 1 – Proof Idea



Claim 1 – Proof Idea

A substantial divergence! Hence, situation does not occur.



Security of the Genesis Rule

Claim 1:

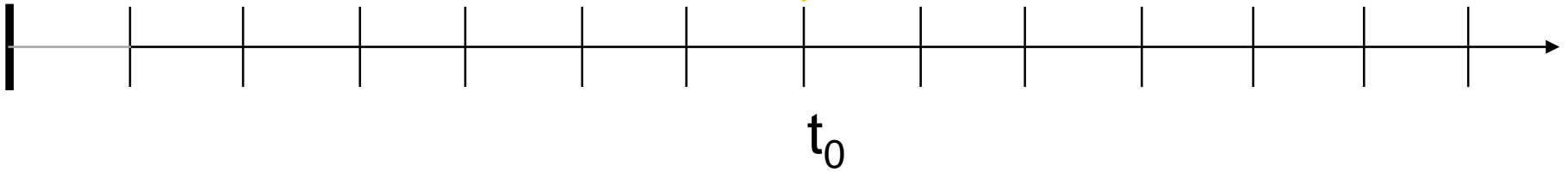
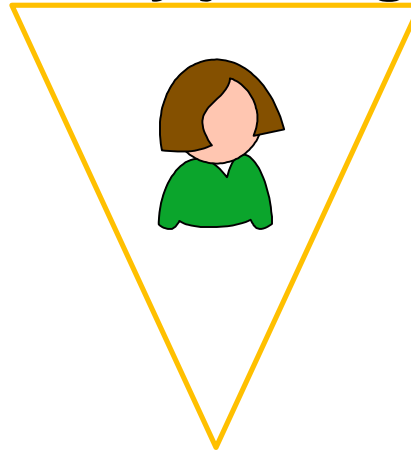
If a party is **always up-to-date** and using the Genesis chain-selection rule, she will **never adopt a chain that forks by more than k blocks** (compared to her local chain in any round).

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.

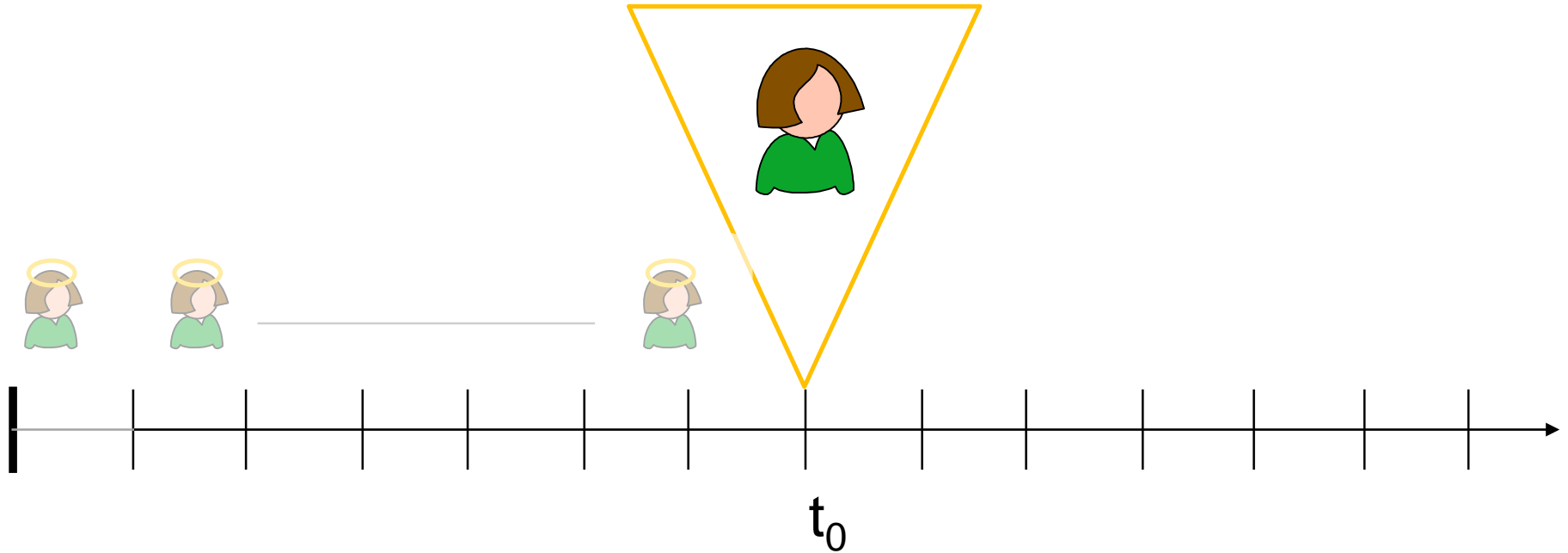
Claim 2 – Proof Idea

First newly joining party





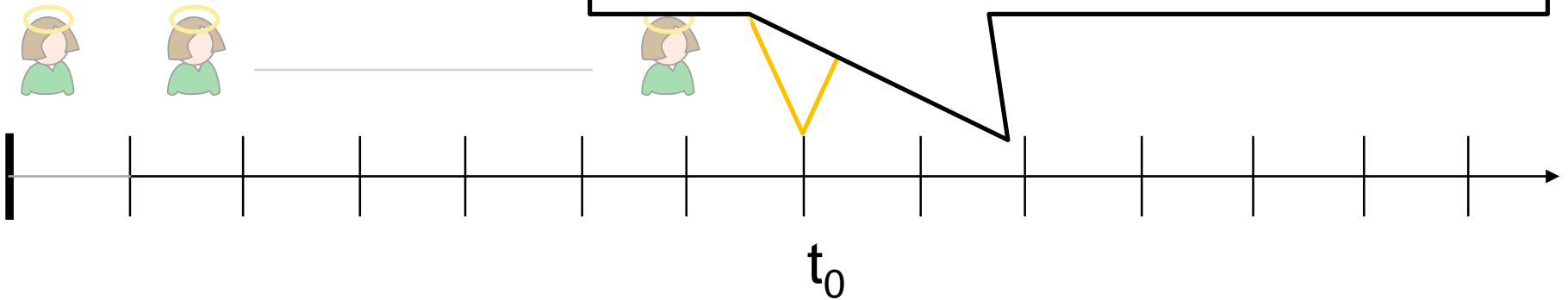
Claim 2 – Proof Idea

First newly joining party

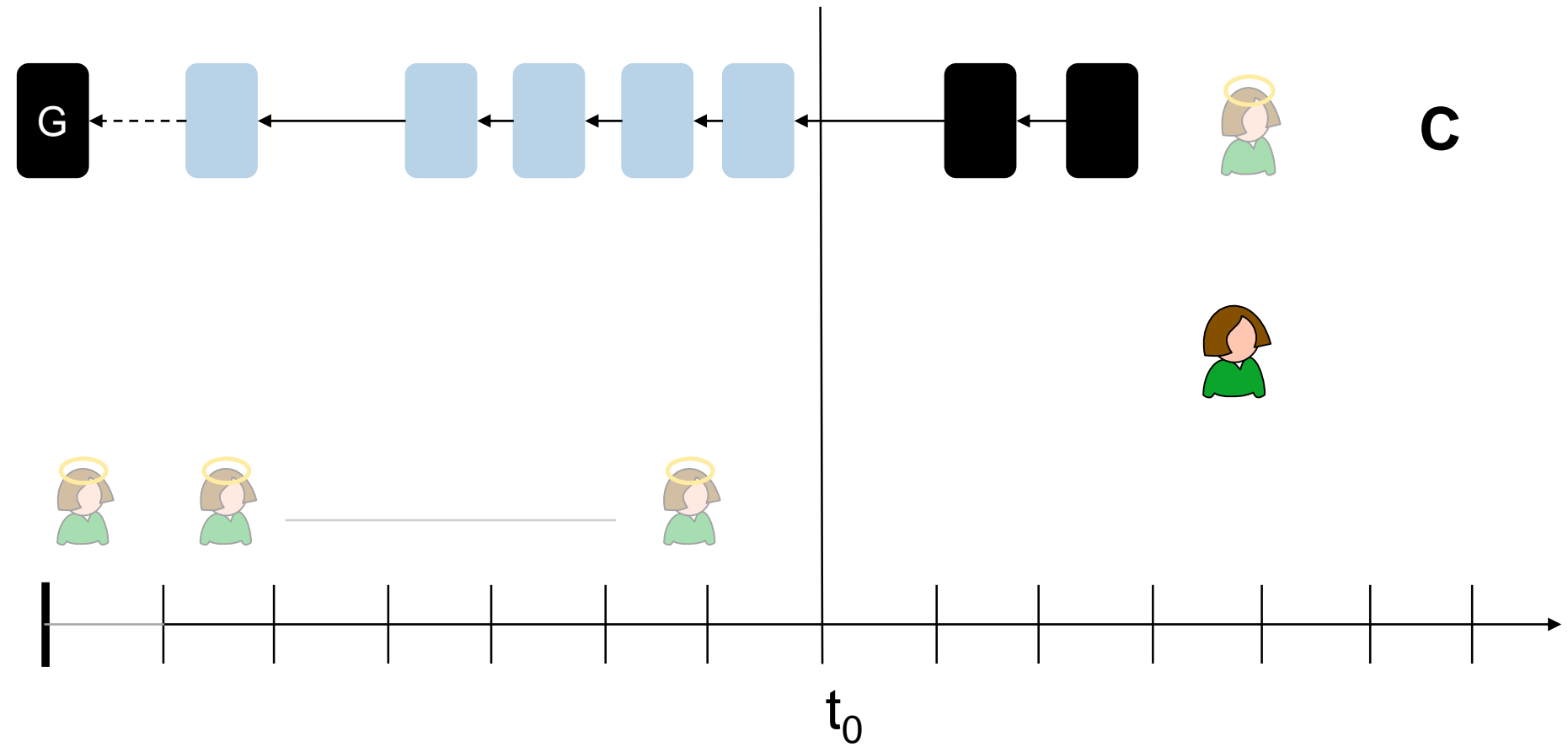


Claim 2 – Proof Idea

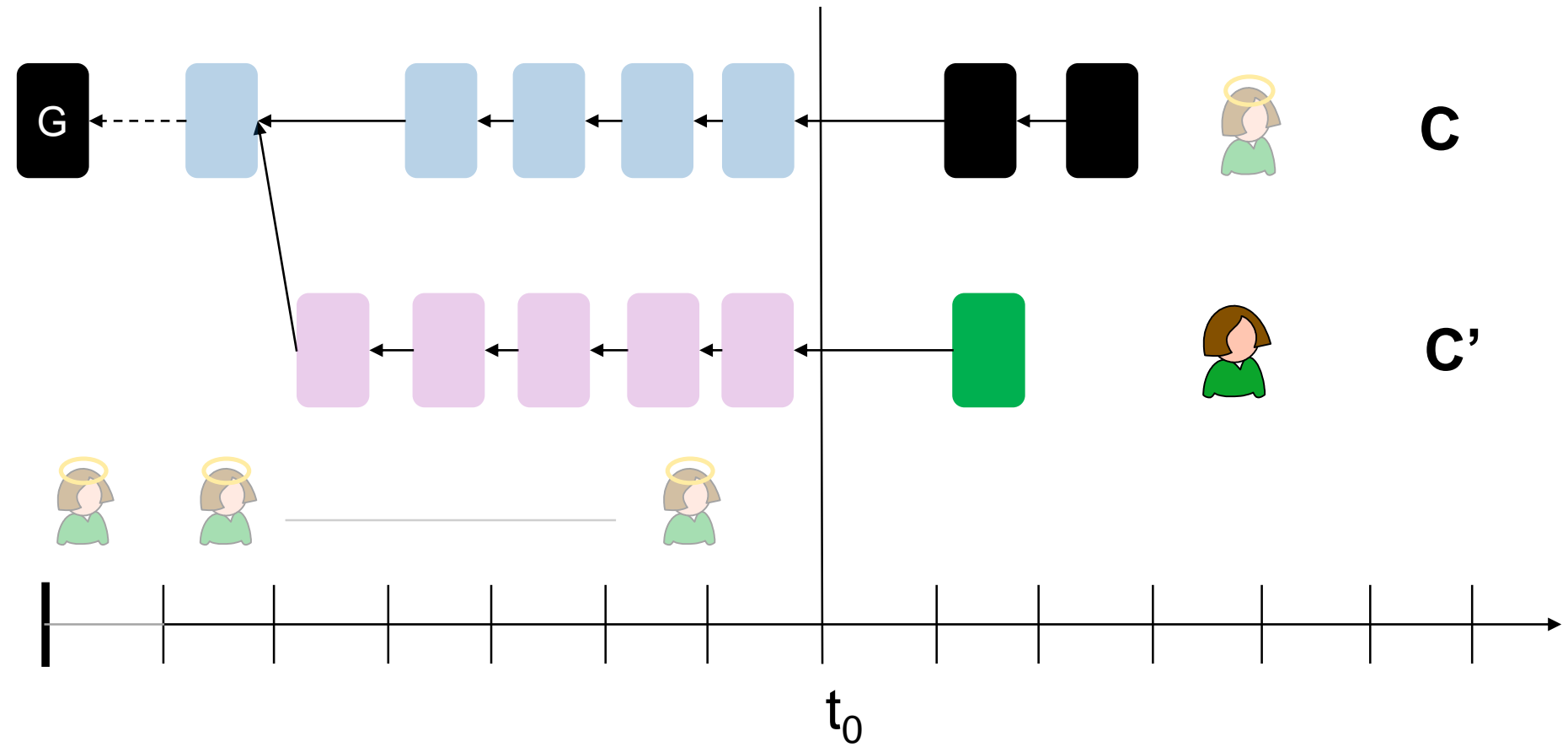
If  decides to adopt a “good” chain C, then so does 



Claim 2 – Proof Idea

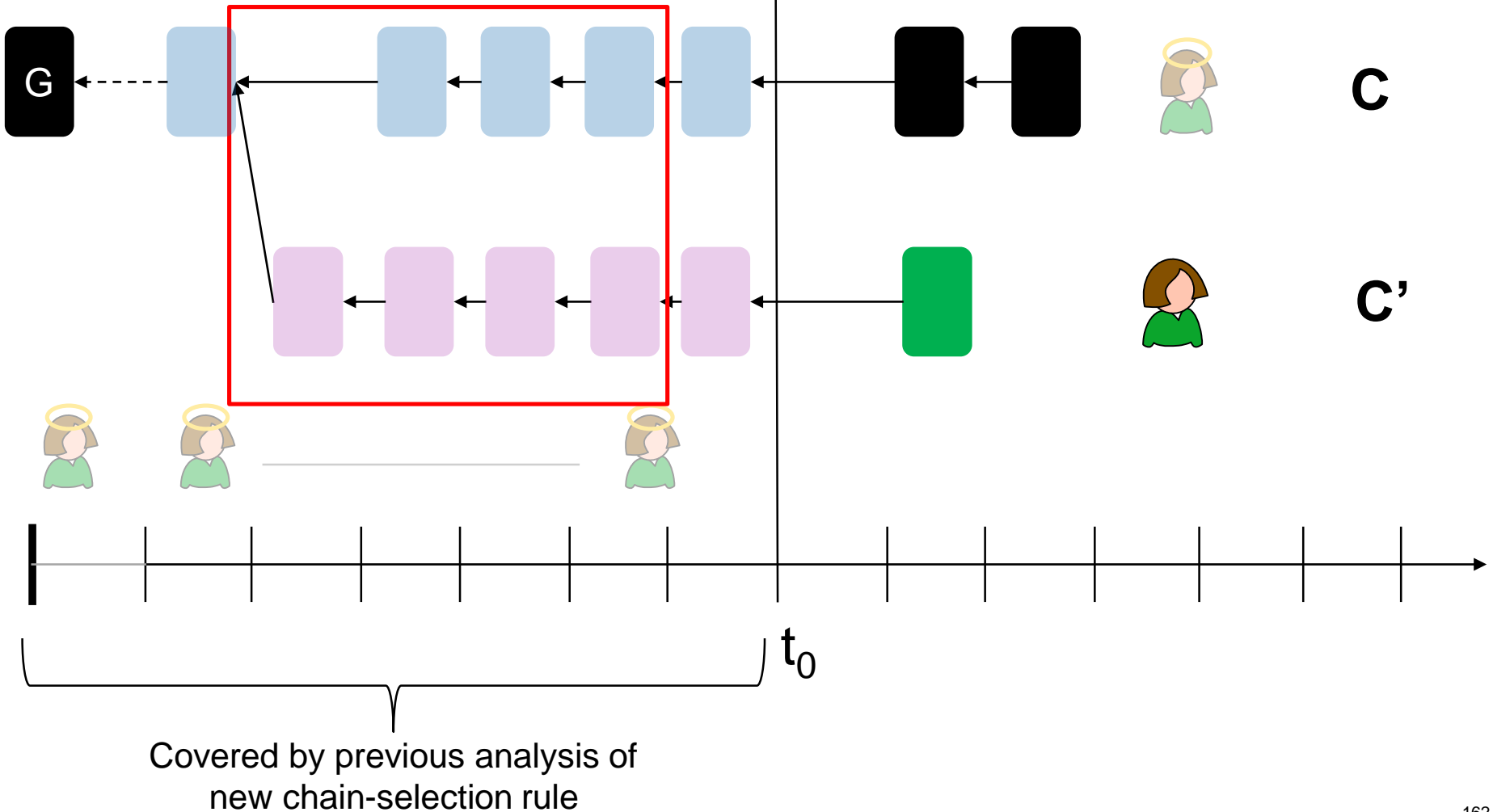


Claim 2 – Proof Idea

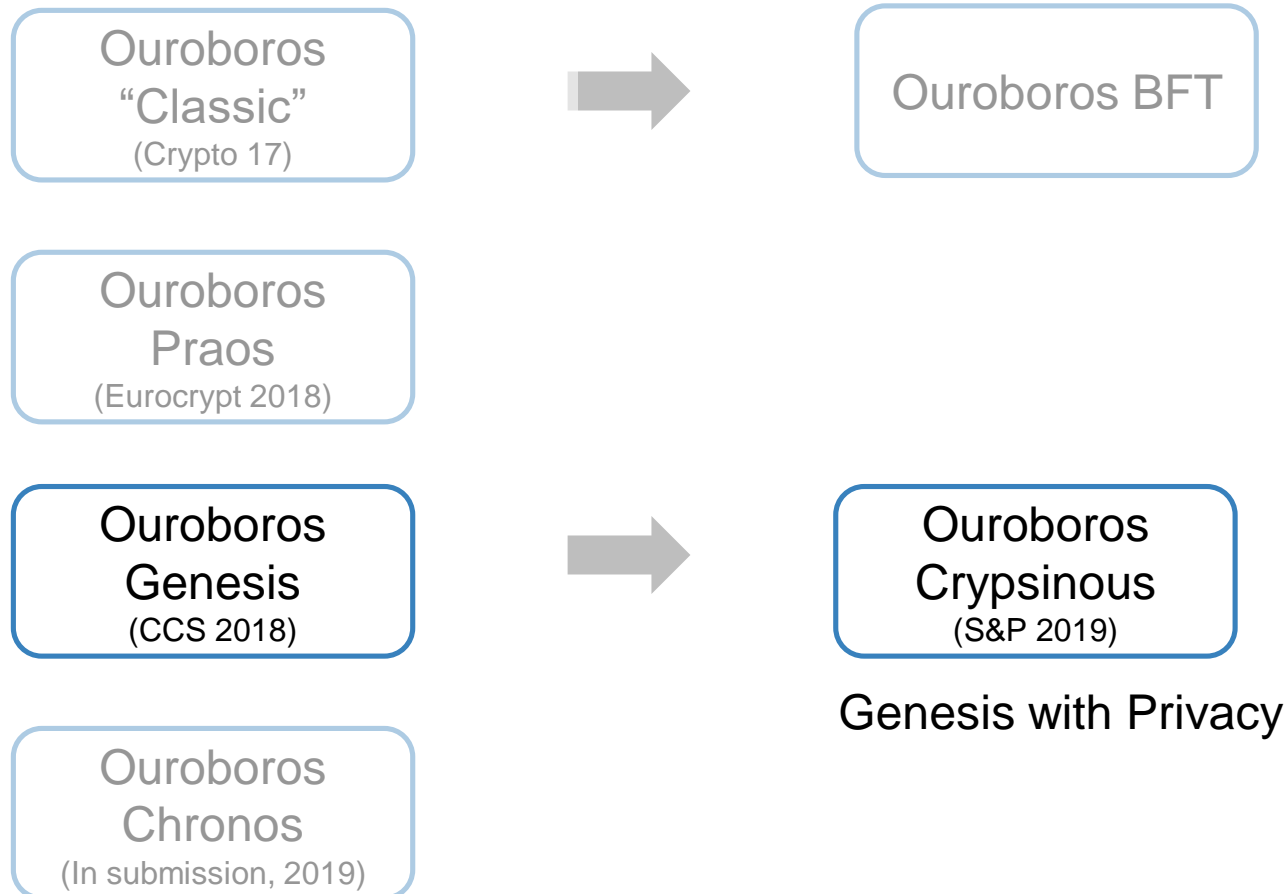


Claim 2 – Proof Idea

A substantial divergence does not occur.



Privacy in Ouroboros: Crypsinous



Privacy in Ouroboros: Crypsinous

Problem Summary:

Public verifiability of leader schedule
VS.

Hide amount of stake possessed

(Eurocrypt 2018)

Ouroboros
Genesis
(CCS 2018)

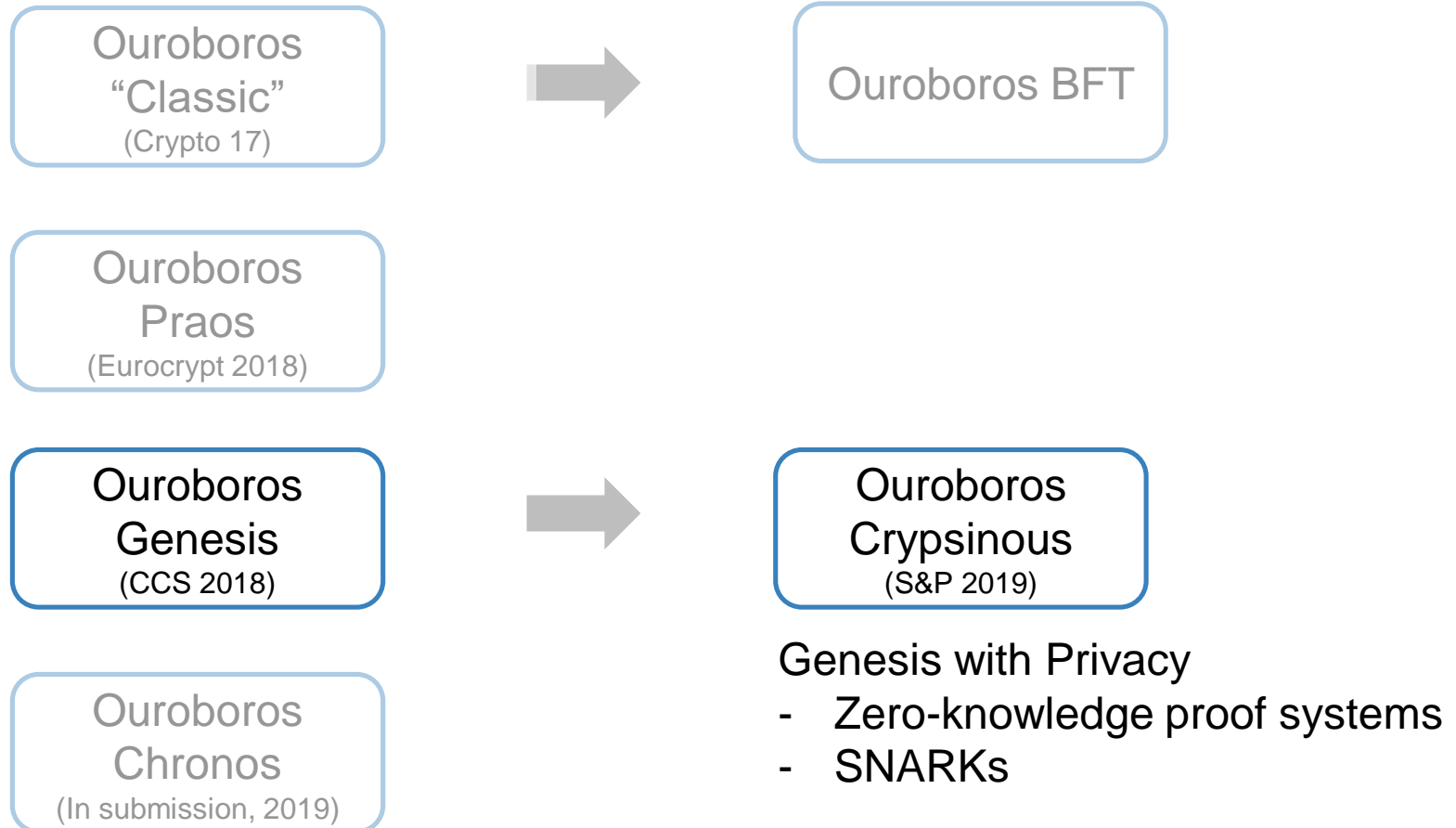


Ouroboros
Crypsinous
(S&P 2019)

Genesis with Privacy

Ouroboros
Chronos
(In submission, 2019)

Privacy in Ouroboros: Crypsinous



Ouroboros: Real-World Implementations

Ouroboros
“Classic”
(Crypto 17)

Ouroboros BFT

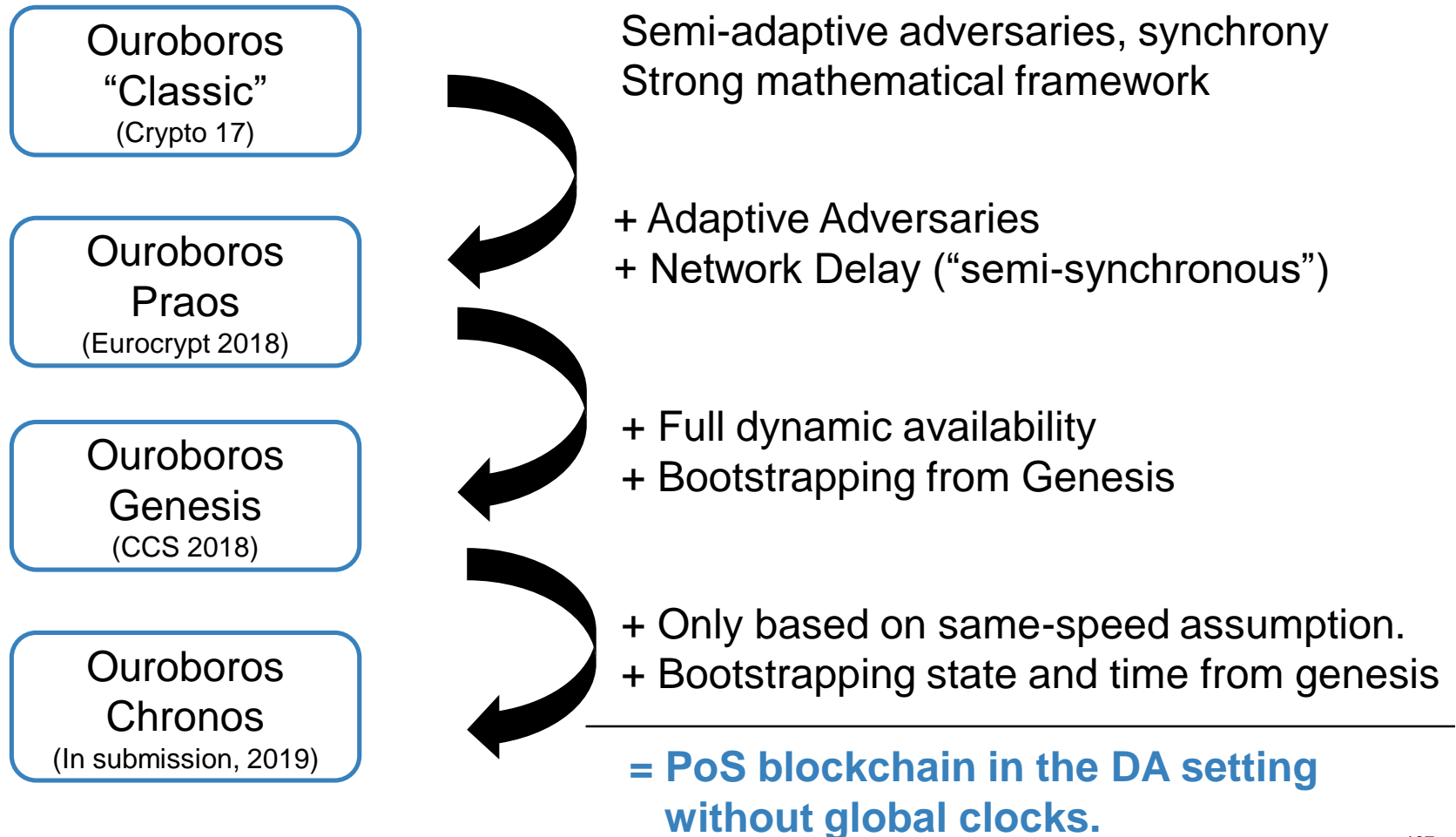
Ouroboros
Praos
(Eurocrypt 2018)

Ouroboros
Genesis
(CCS 2018)

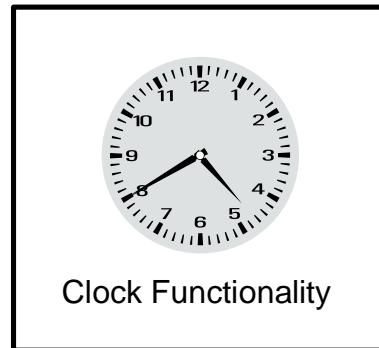
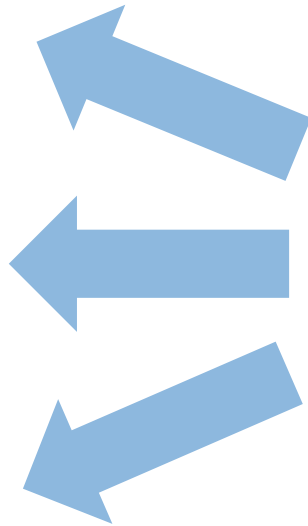
Cardano is running on Ouroboros PoS and other companies are implementing versions of it.

Ouroboros
Chronos
(In submission, 2019)

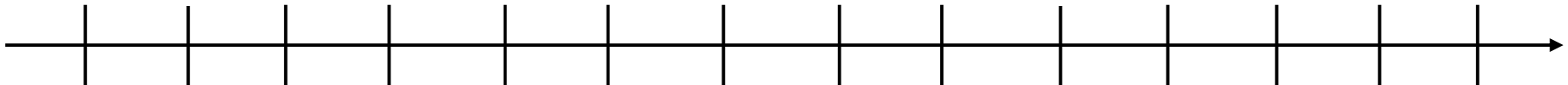
Ouroboros – Chronos



So far: Dependency on a Good Timing Service

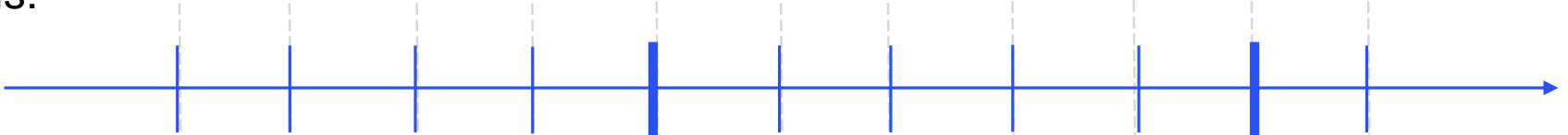


Time axis:

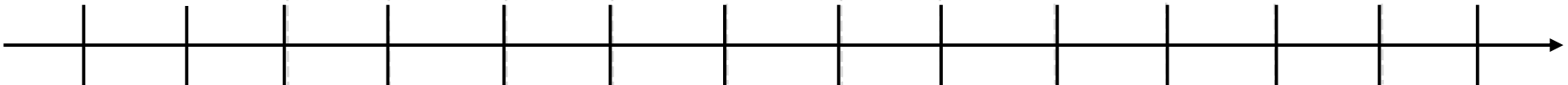


So far: Dependency on a Good Timing Service

Slot Axis:

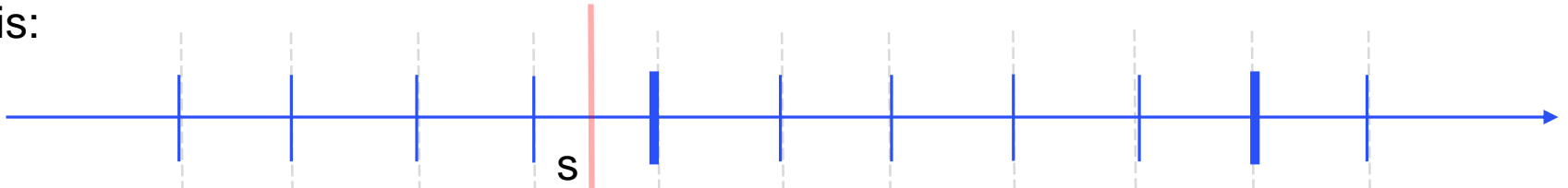


Time axis:

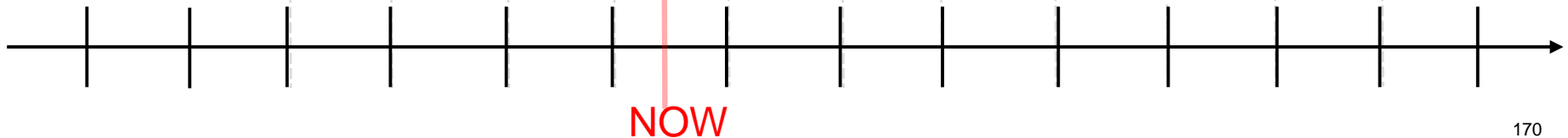


So far: Dependency on a Good Timing Service

Slot Axis:



Time axis:



So far: Dependency on a Good Timing Service

Slot Axis:

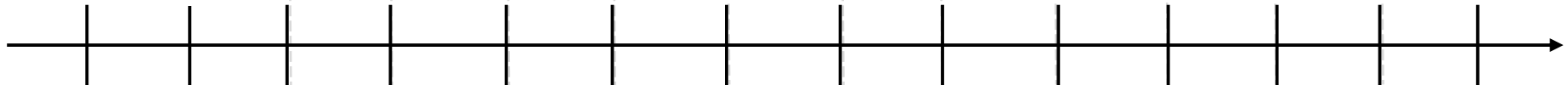


Recall Lottery (Nakamoto-Style Consensus):

- A party i is leader if and only if $VRF_{sk_i}(\text{TEST}, \text{seed}, \text{slot}) < T(\text{stake}_i)$
- All have the **same idea of future and past**
 - Example: “future chains are rejected - because bad anyway”.



Time axis:

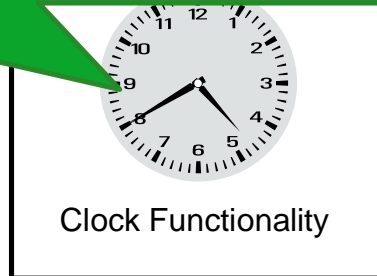
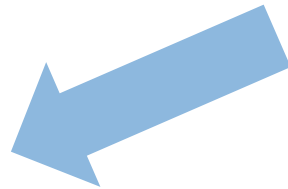


So far

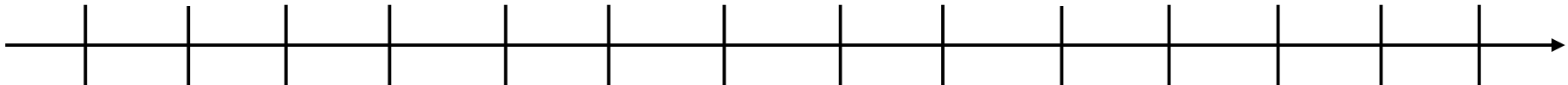
Service

Strong Assumption:

- Perfect time coordination for everyone, including newly joining parties.
- Needs another protocol eventually, e.g. NTP.

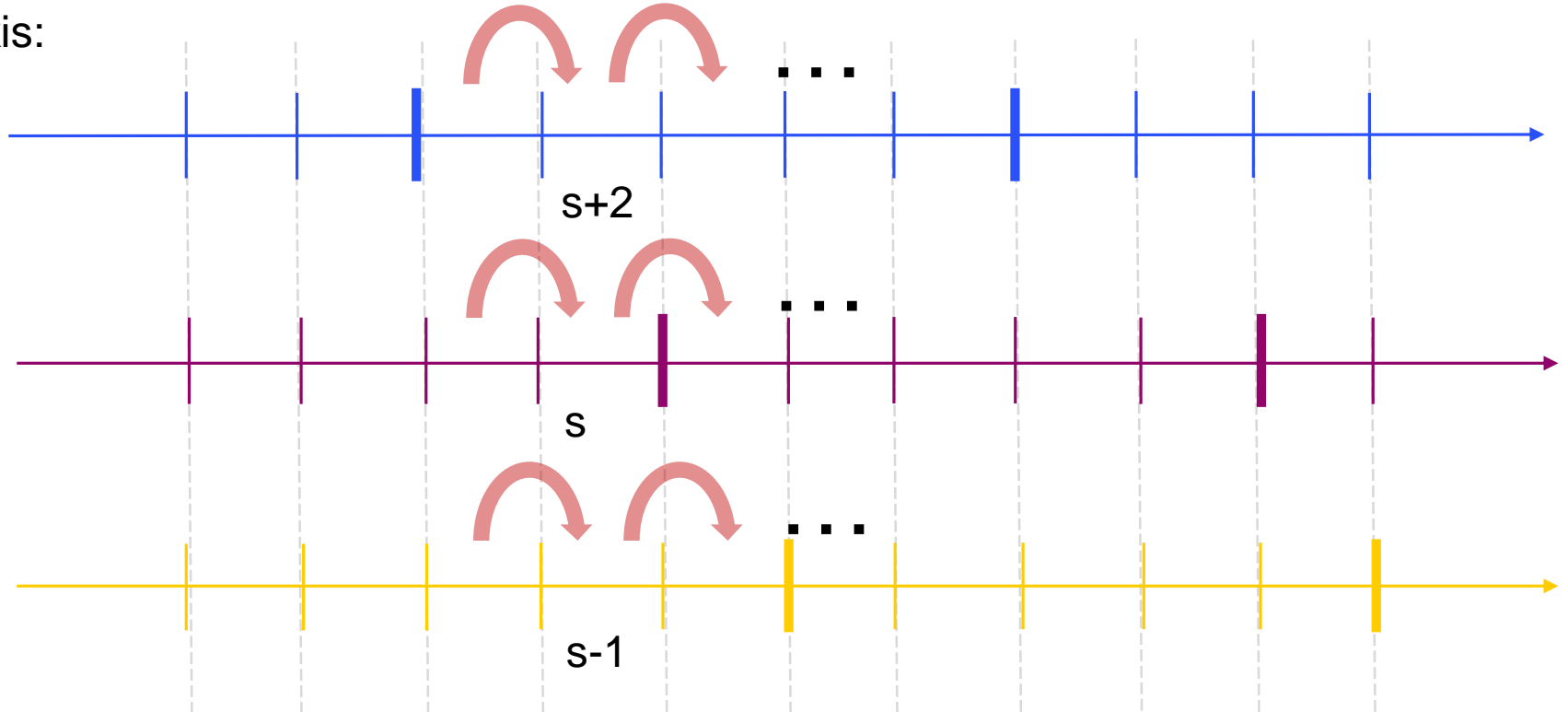
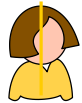


Time axis:



Better: Same-Speed Assumption

Slot Axis:

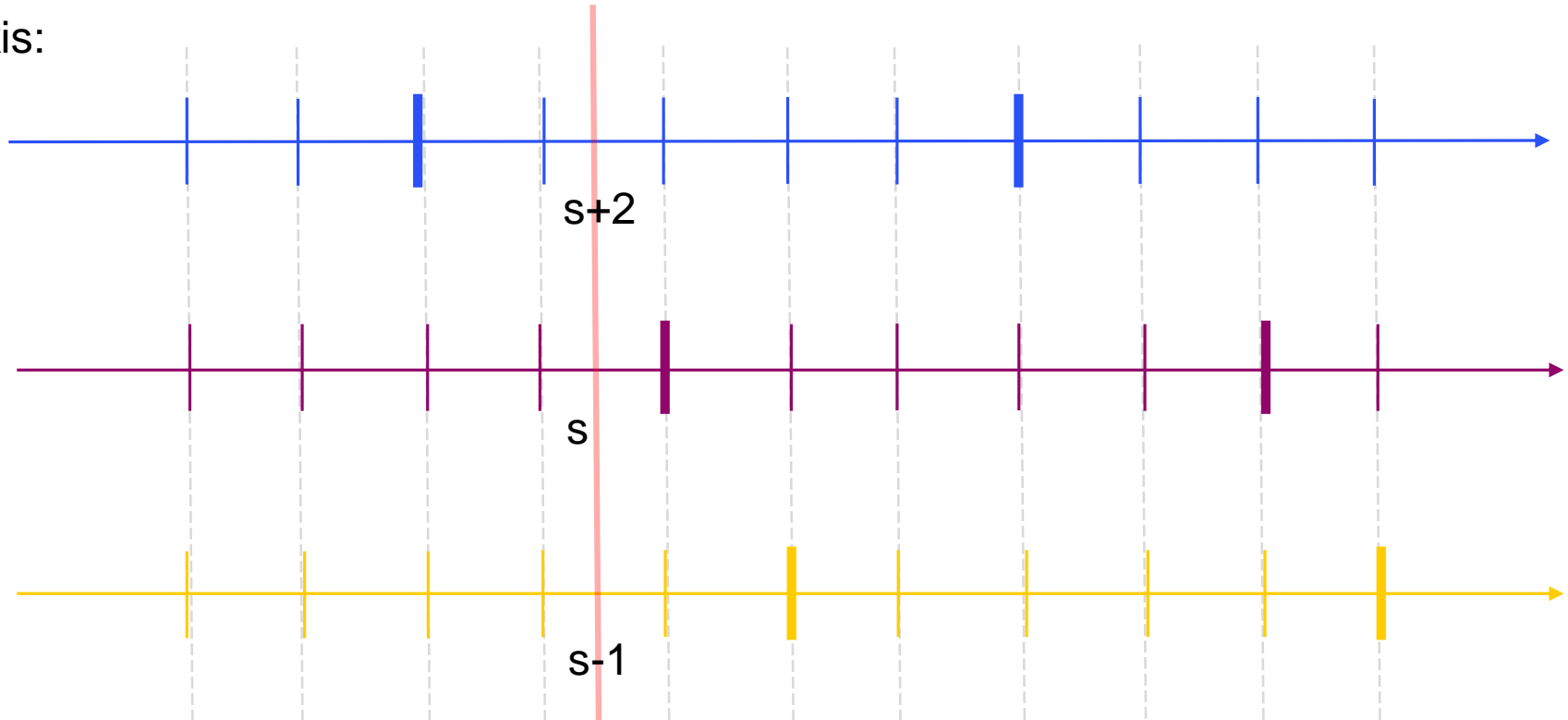
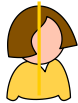


Time axis:



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 Δ apart (order of network delay).
- **Small adjustments** needed to Ouroboros Genesis to deal with future chains

Genesis: Situation not that bad...

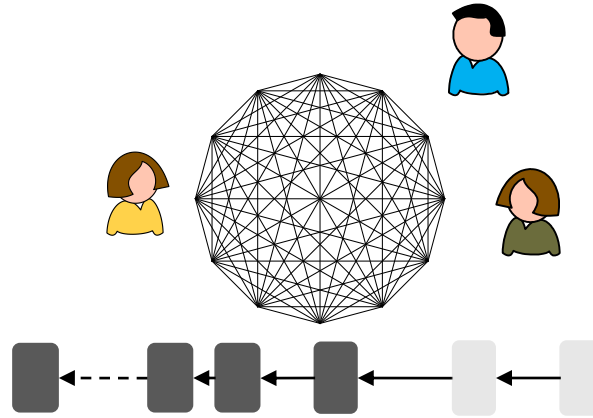
If we manage to keep **all honest parties somewhat close** we're kind of good.

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

→ Same-speed: Initial parties do stay close.

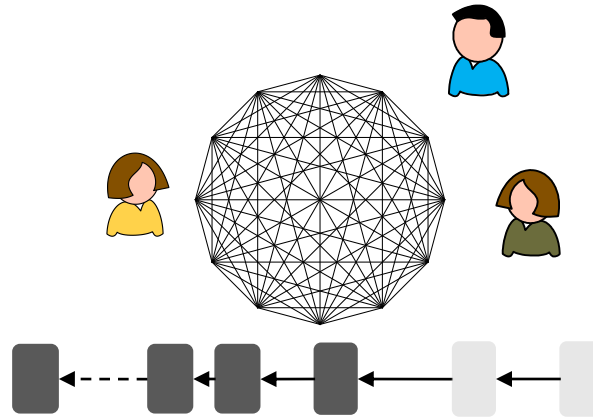
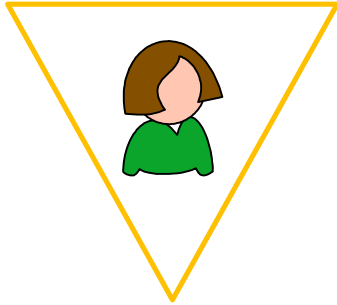
→ **Joining parties** have a harder life...

Genesis: Situation not that bad...



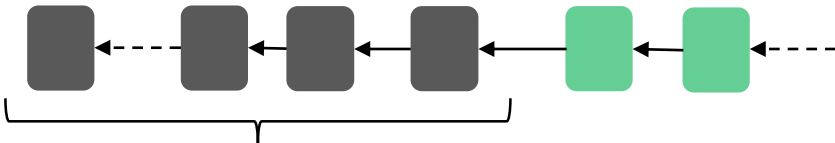
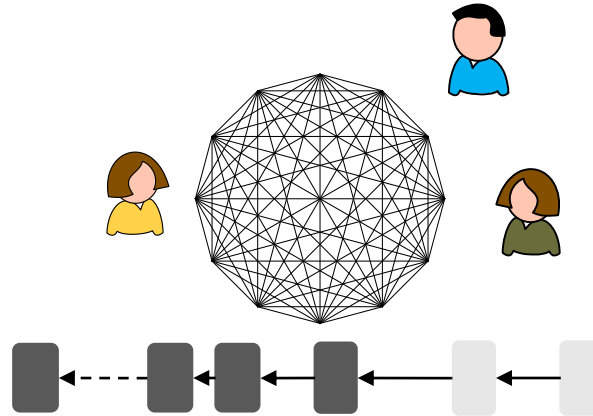
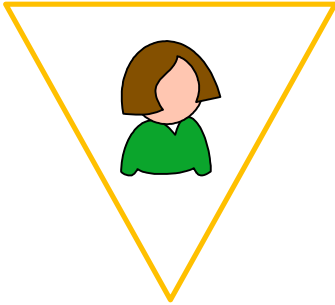
Genesis: Situation not that bad...

Joining party:



Genesis: Situation not that bad...

Joining party:



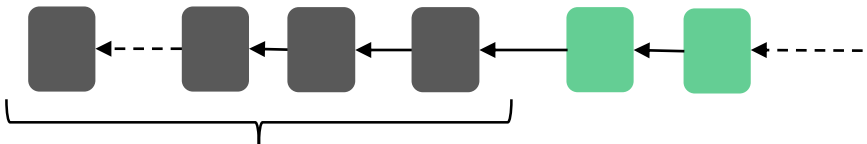
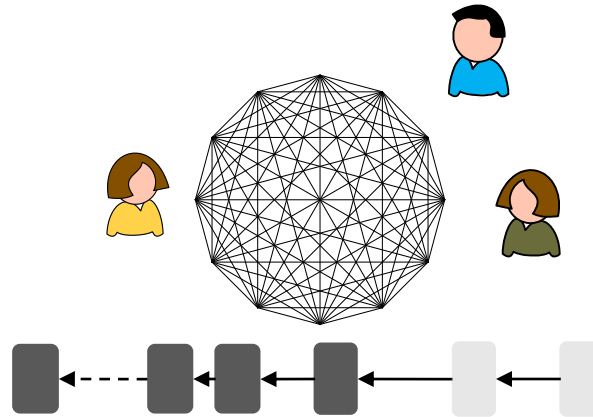
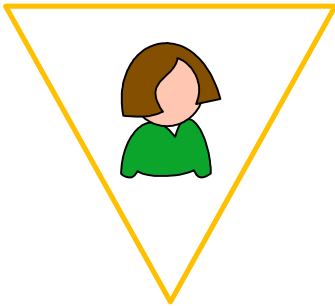
Genesis chain selection rule:

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



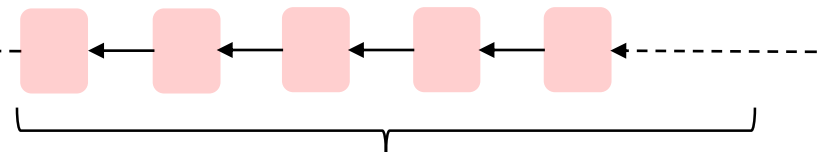
Genesis: ~~Situation not that bad...~~

Joining party:



Genesis chain selection rule:

- **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 \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).

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 \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.

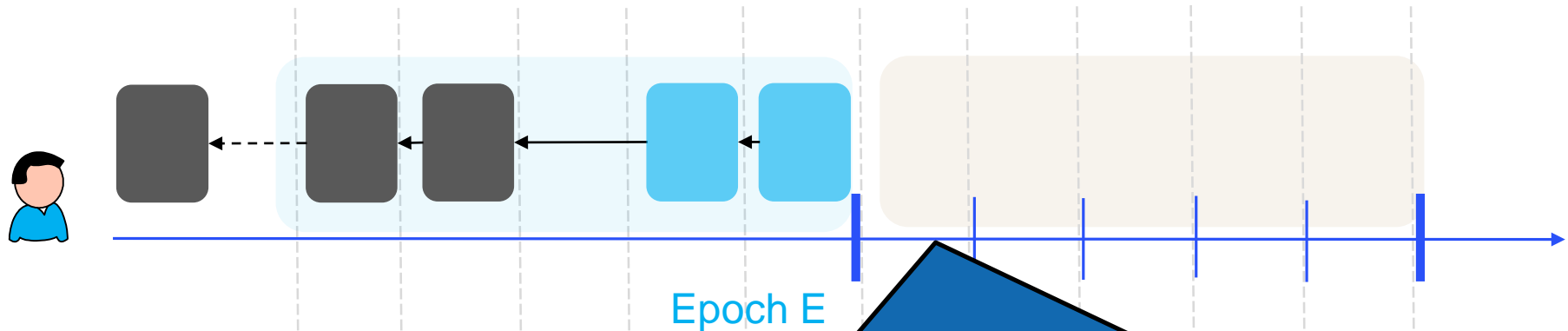
Chronos Overview

- **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

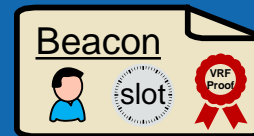
- **Alert parties:** broadcast **time-beacons** and **leave evidence** of beacons in the blockchain.
- They perform **local-clock adjustments** based on the evidence in the chain.
 - Small adjustments to local clocks at the end of an epoch
 - Based on the evidence left in the chain.
- **Joining parties:** 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

Chronos – Sync-Beacons



Additional “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.

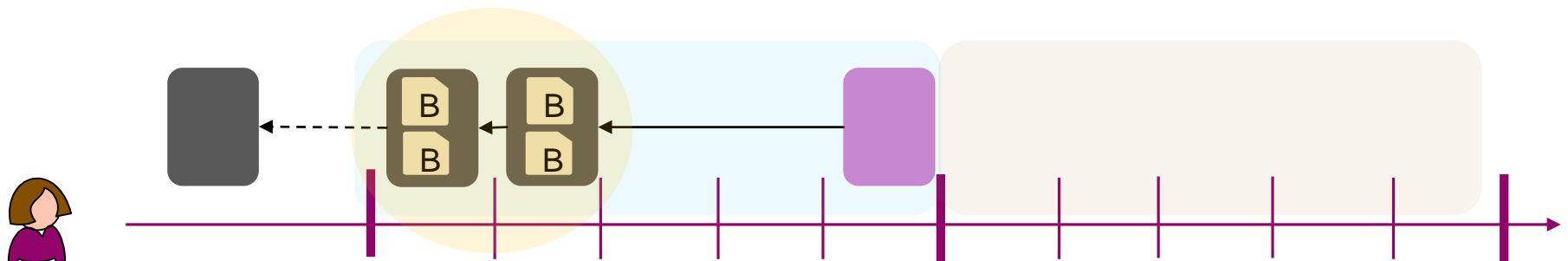


Chronos Overview

- **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

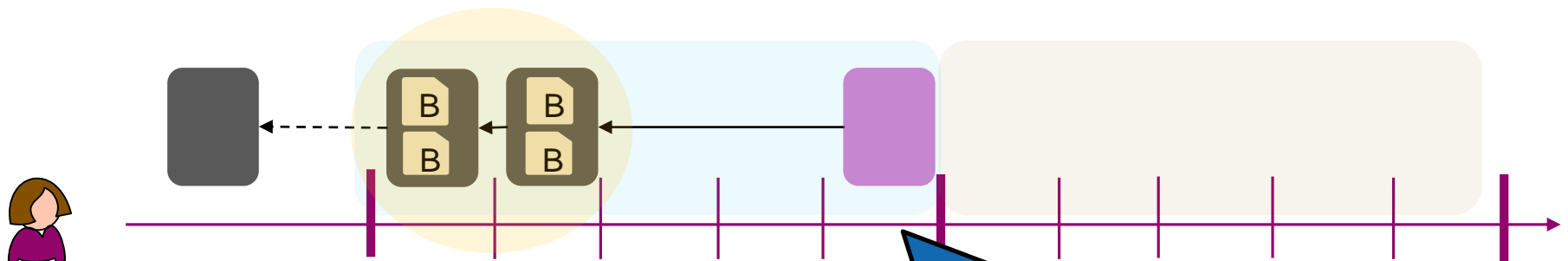
Chronos: Synchronization Procedure

- **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

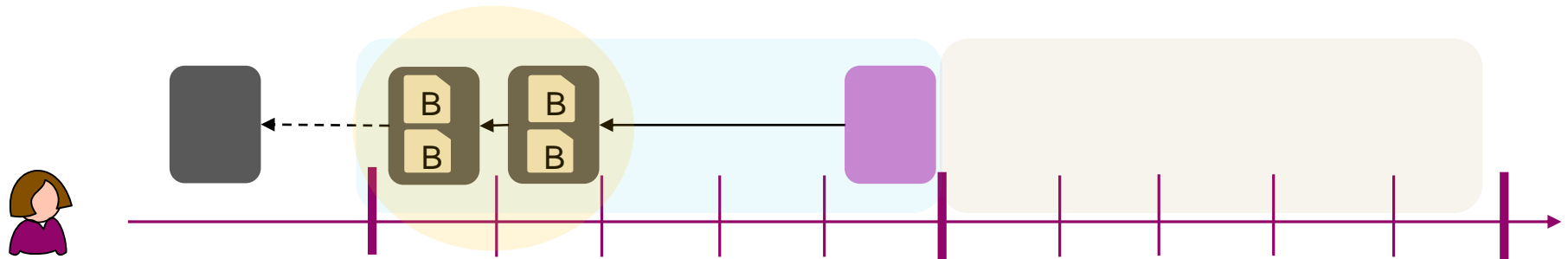
- **Throughout the epochs:** Alice records the arrival times of valid beacons (filter out duplicates, invalid ones etc.)
- **At the end of each epoch:** Compute local clock-adjustment.



- **At the end of epoch:** for each recorded beacon, do:

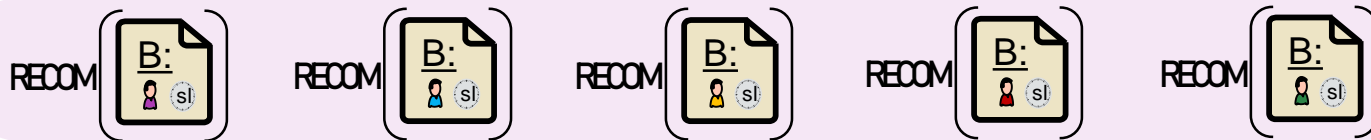
$$\text{RECOM} \left(\begin{array}{c} \text{Beacon} \\ \text{slot} \\ \text{VRF Proof} \end{array} \right) := \text{slot} - \text{ARRIVALTIME} \left(\begin{array}{c} \text{Beacon} \\ \text{slot} \\ \text{VRF Proof} \end{array} \right)$$

Chronos: Synchronization Procedure

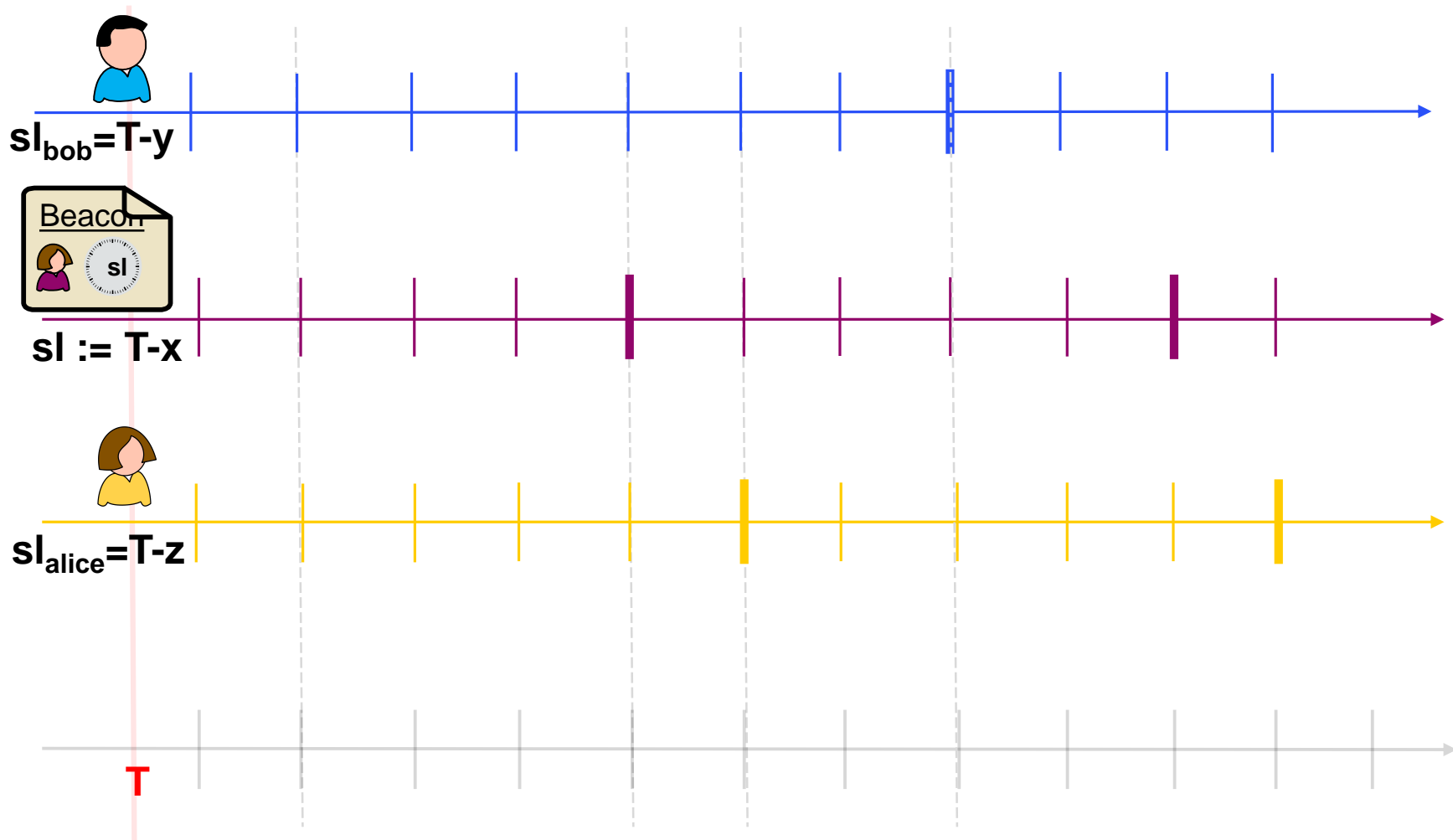


Adjustment rule:

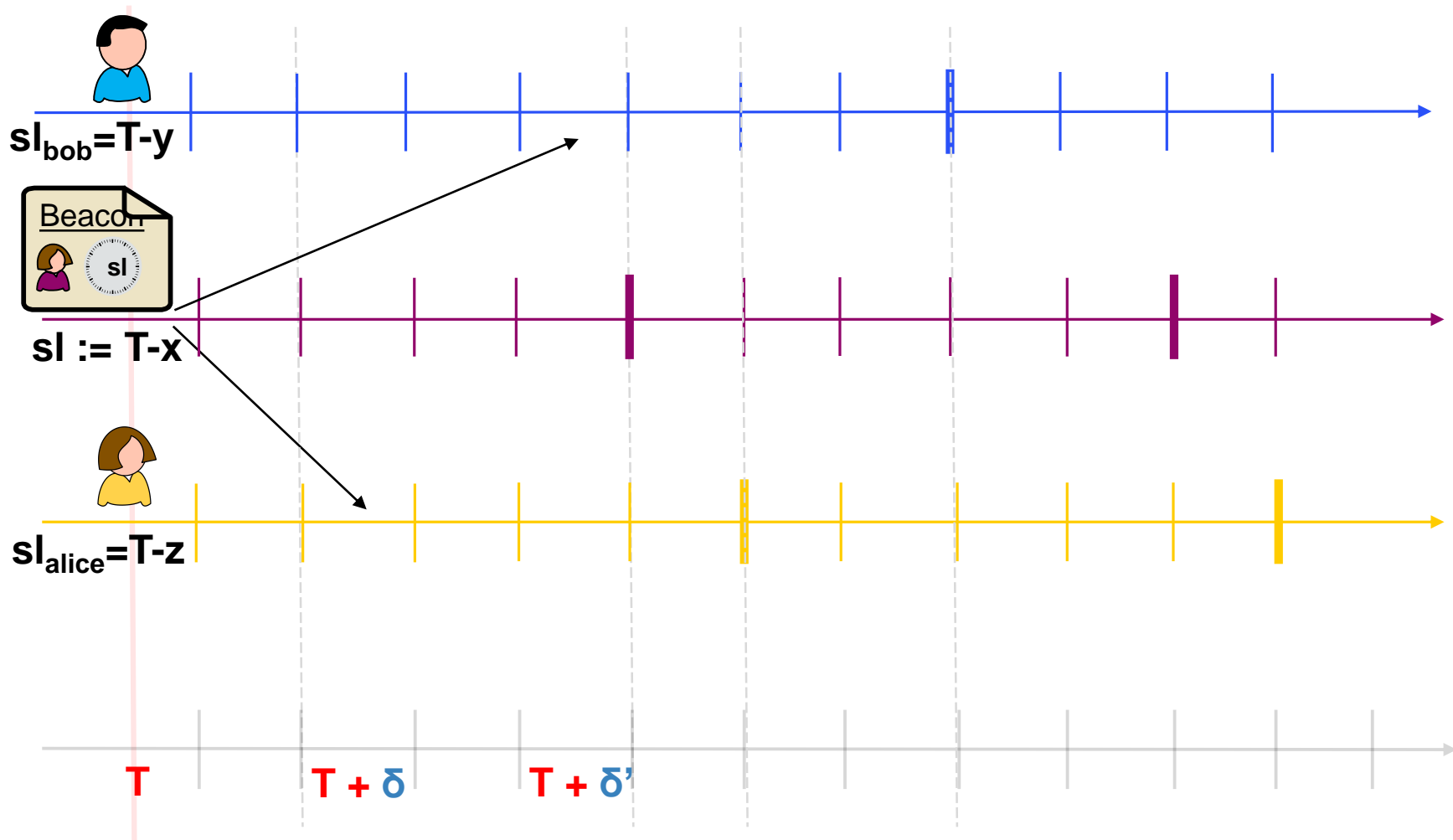
- At the end of epoch: **add** the **median of recommendations** to local time:



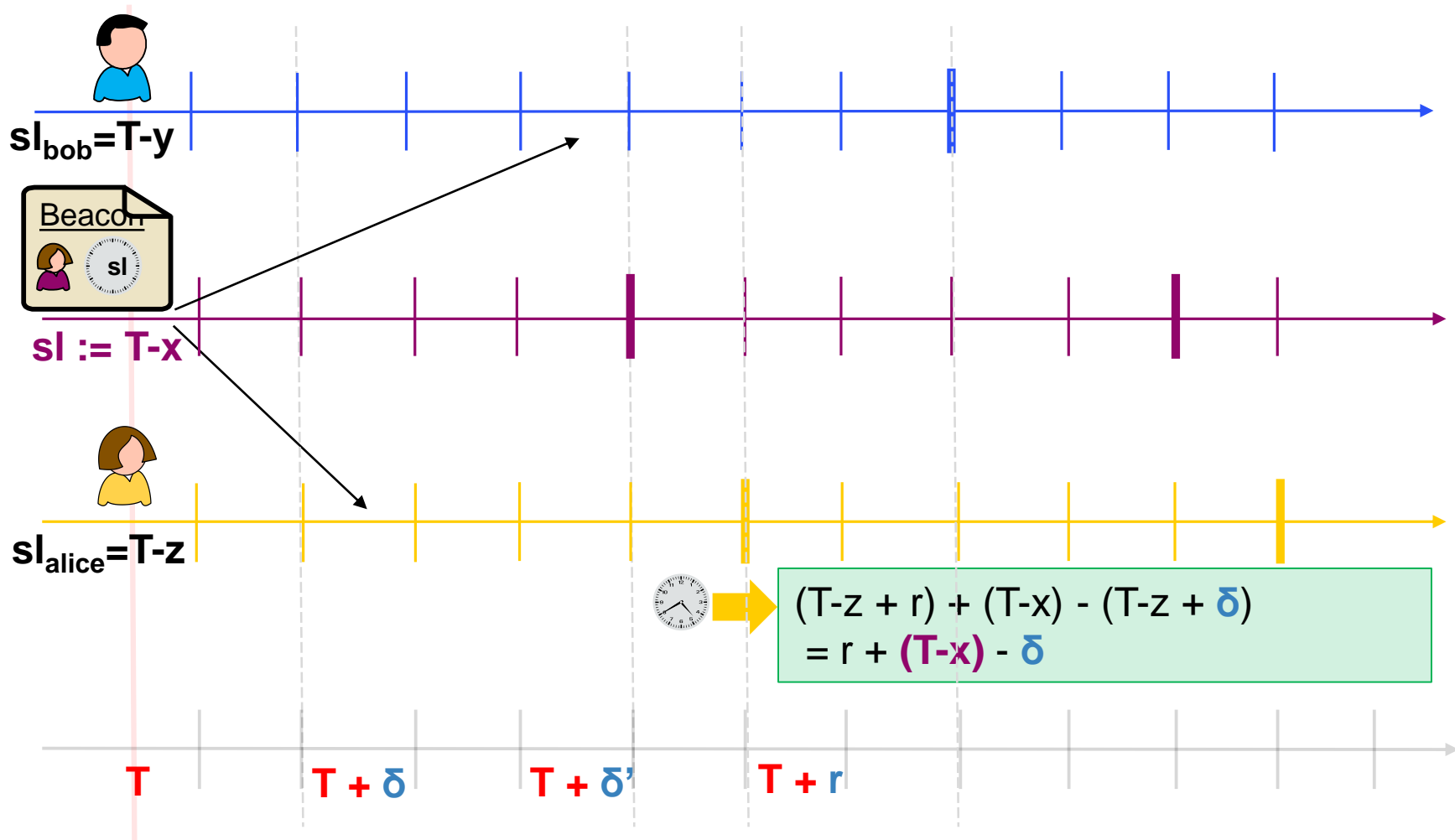
Example



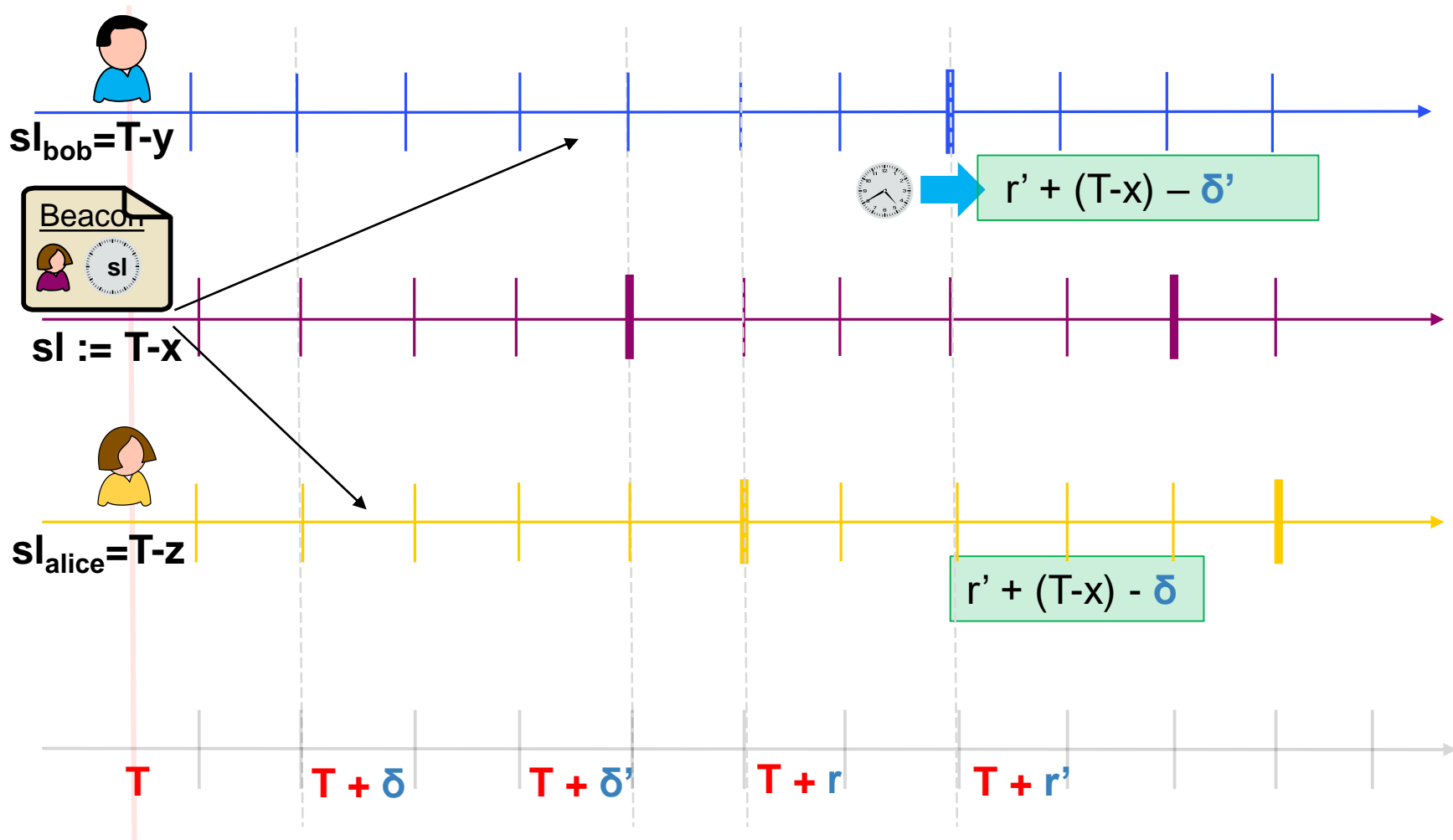
Example



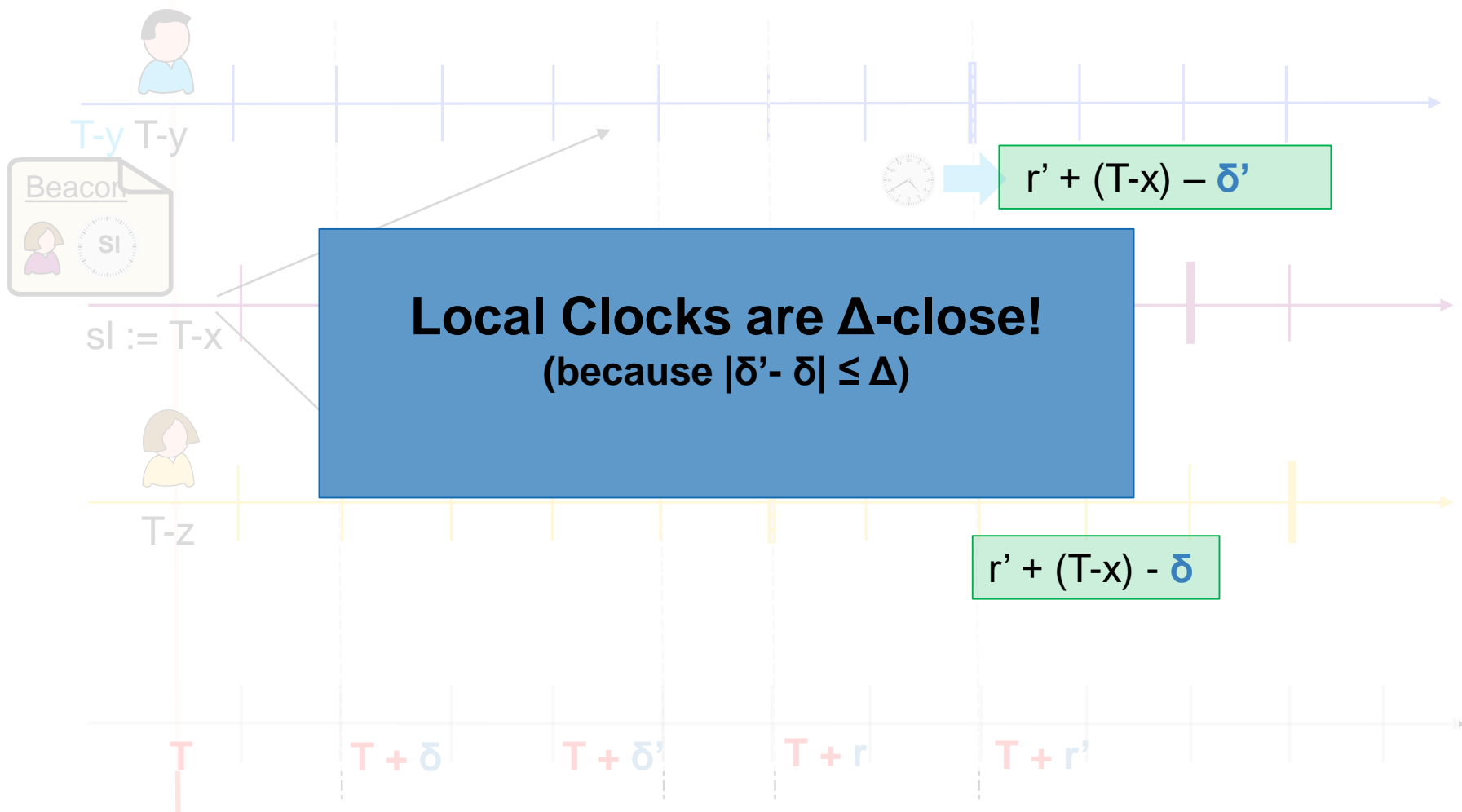
Example



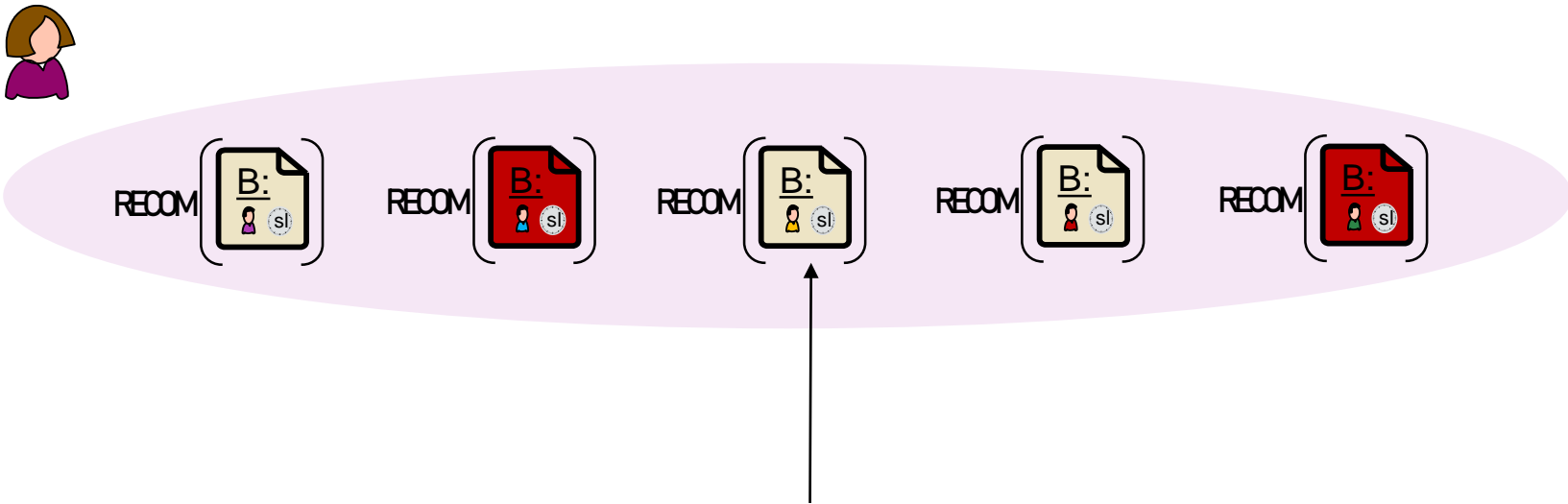
Example



Properties of the Synchronization Procedure



Properties of the Synchronization Procedure

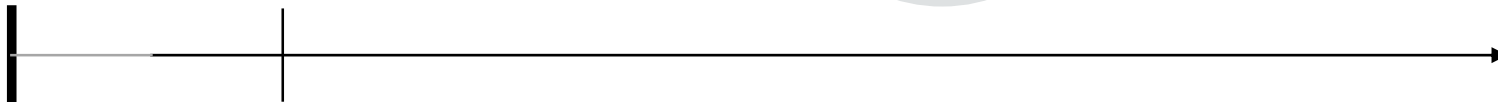
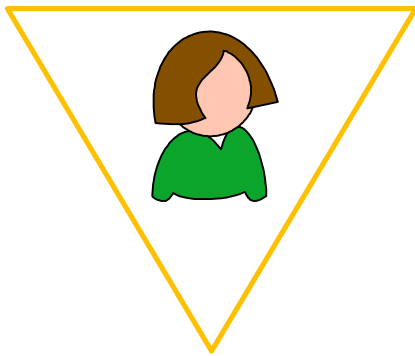


Furthermore, by honest-majority assumption:
→ Median, i.e., **adjustment is bounded**.

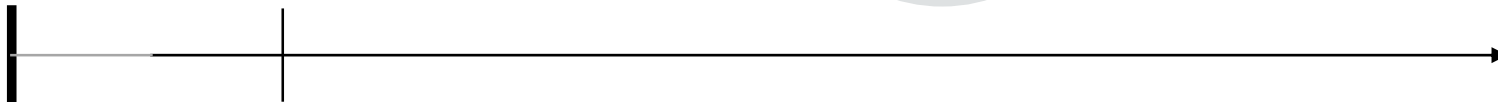
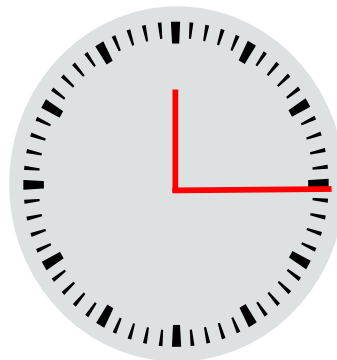
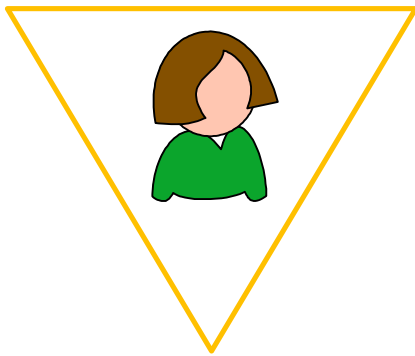
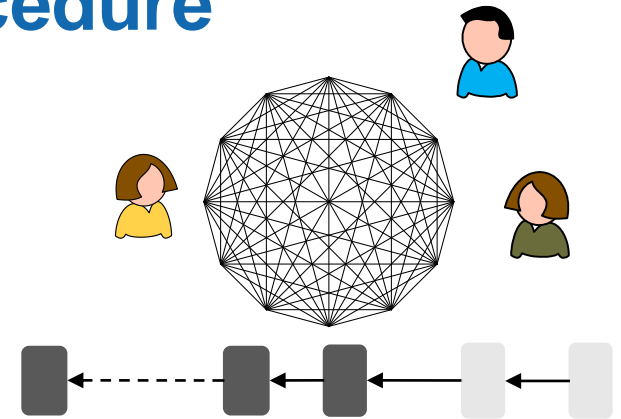
Chronos Overview

- **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

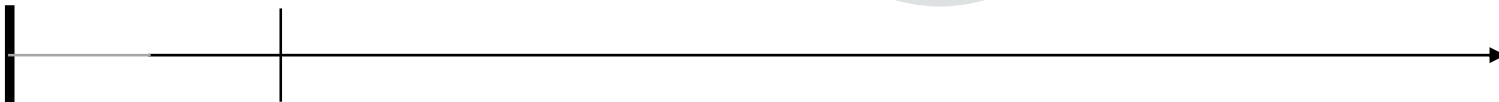
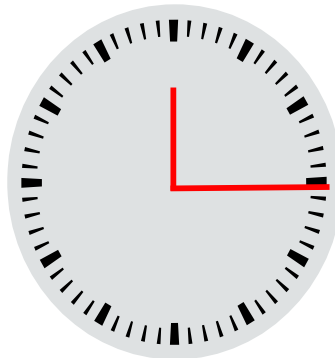
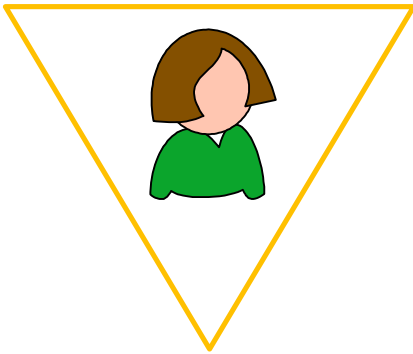
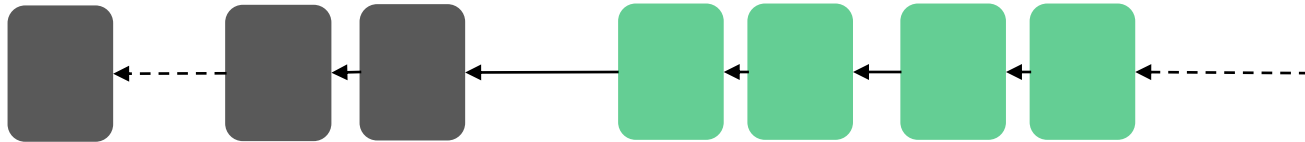
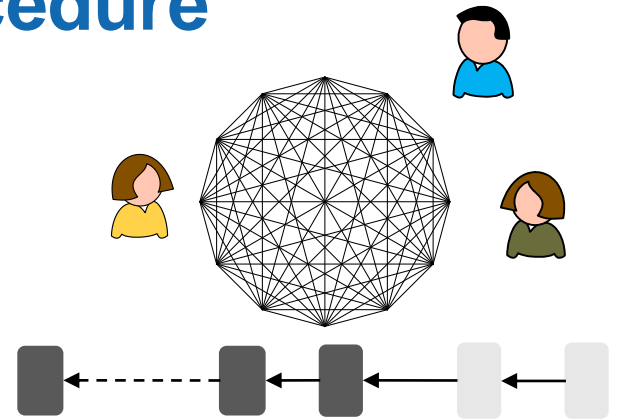
Chronos: Joining Procedure



Chronos: Joining Procedure

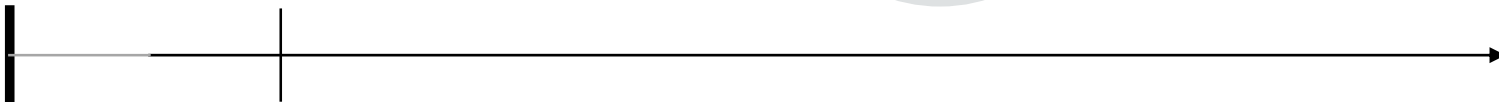
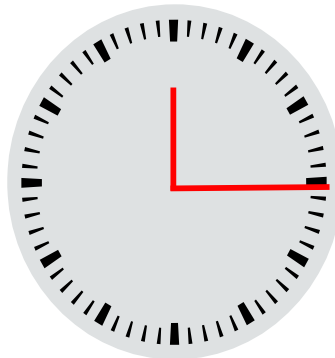
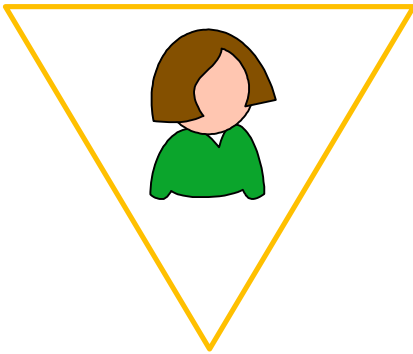
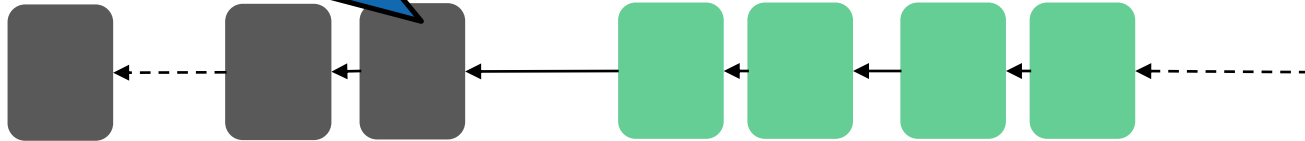
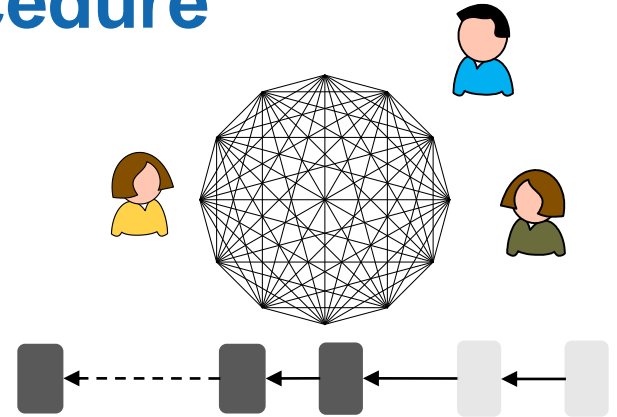


Chronos: Joining Procedure

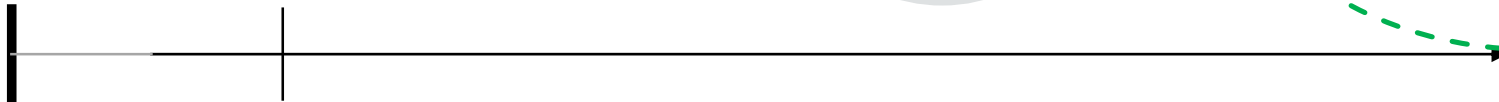
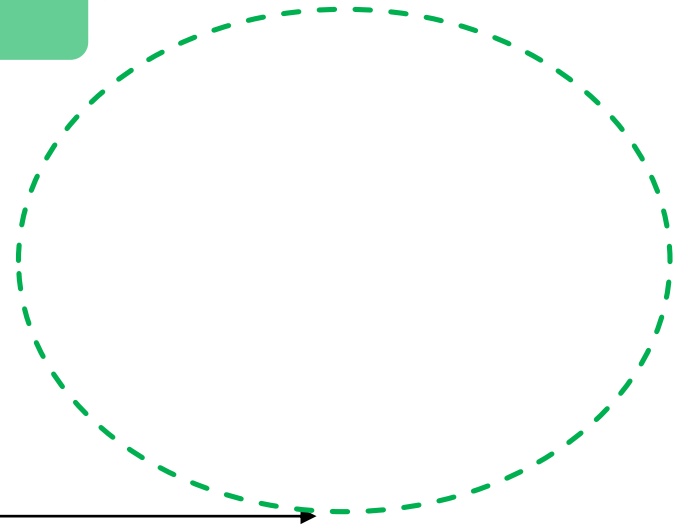
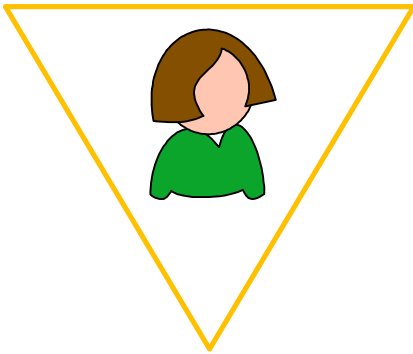
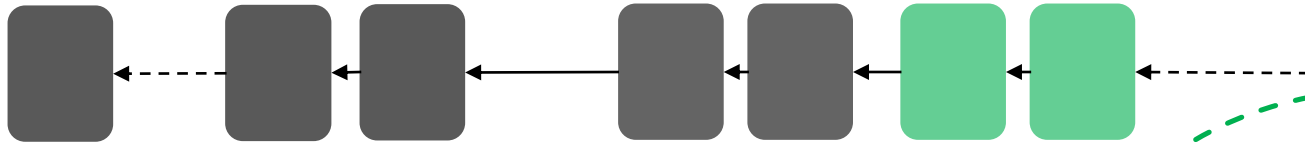
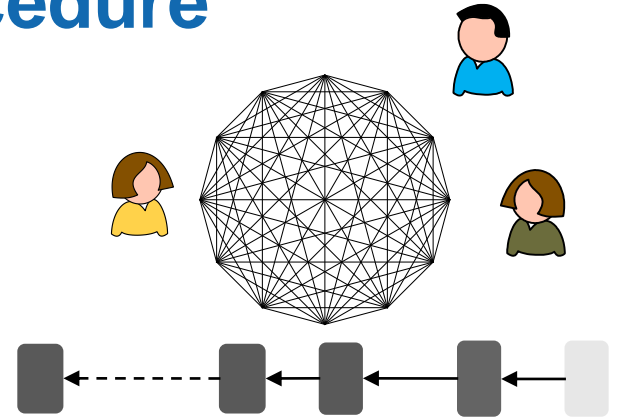


Chronos: Joining Procedure

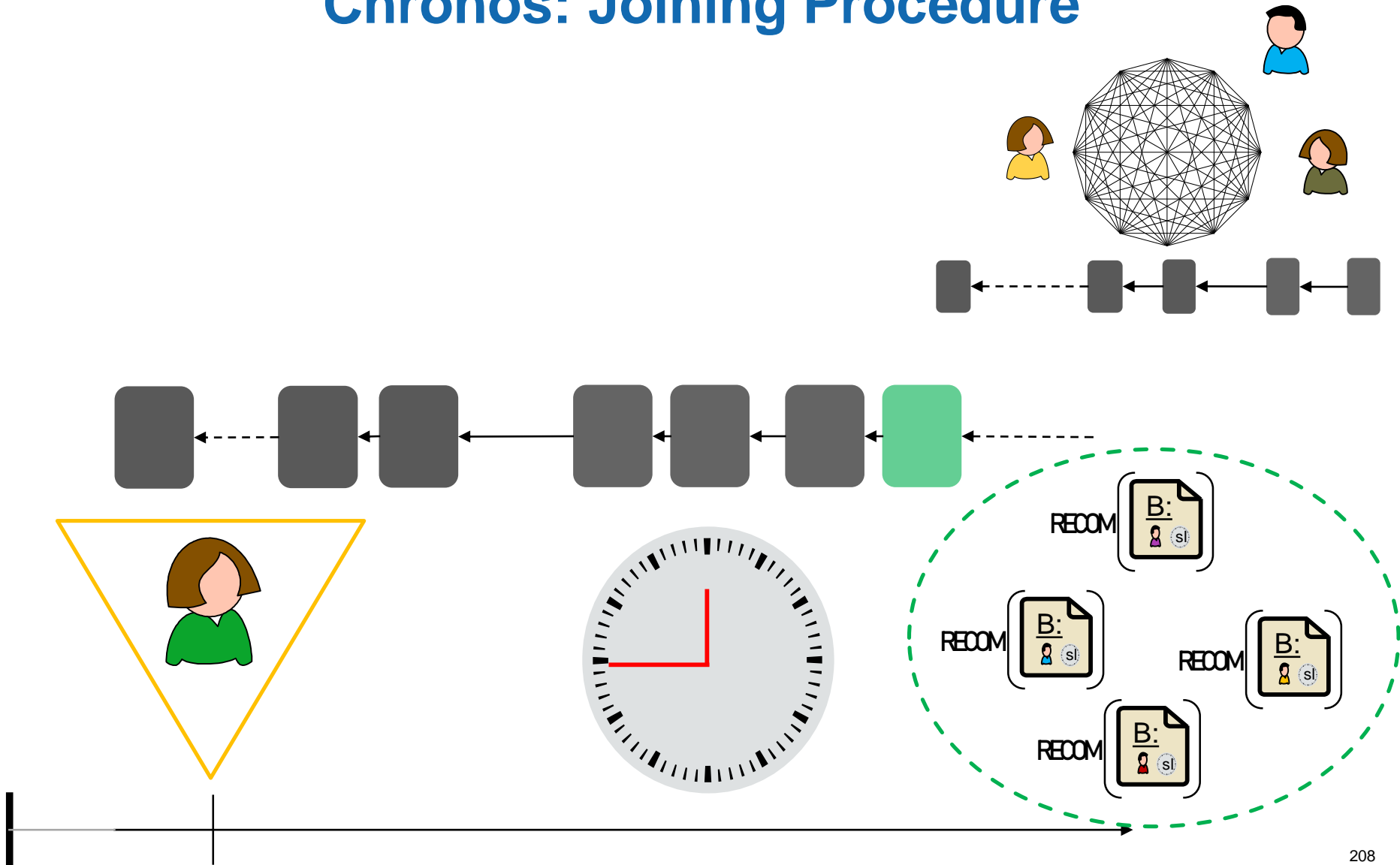
- Densest chain wins, good prefix.



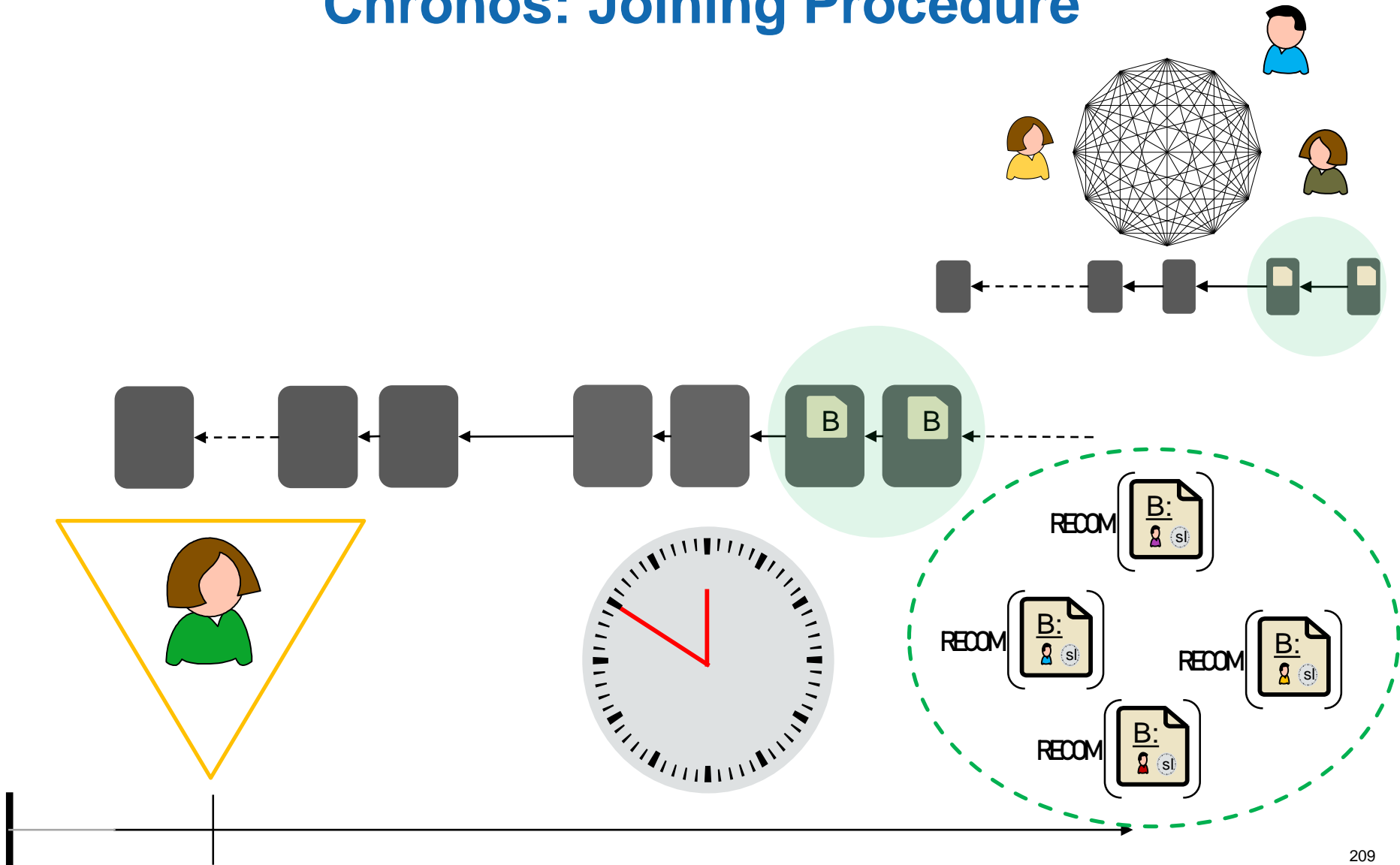
Chronos: Joining Procedure



Chronos: Joining Procedure



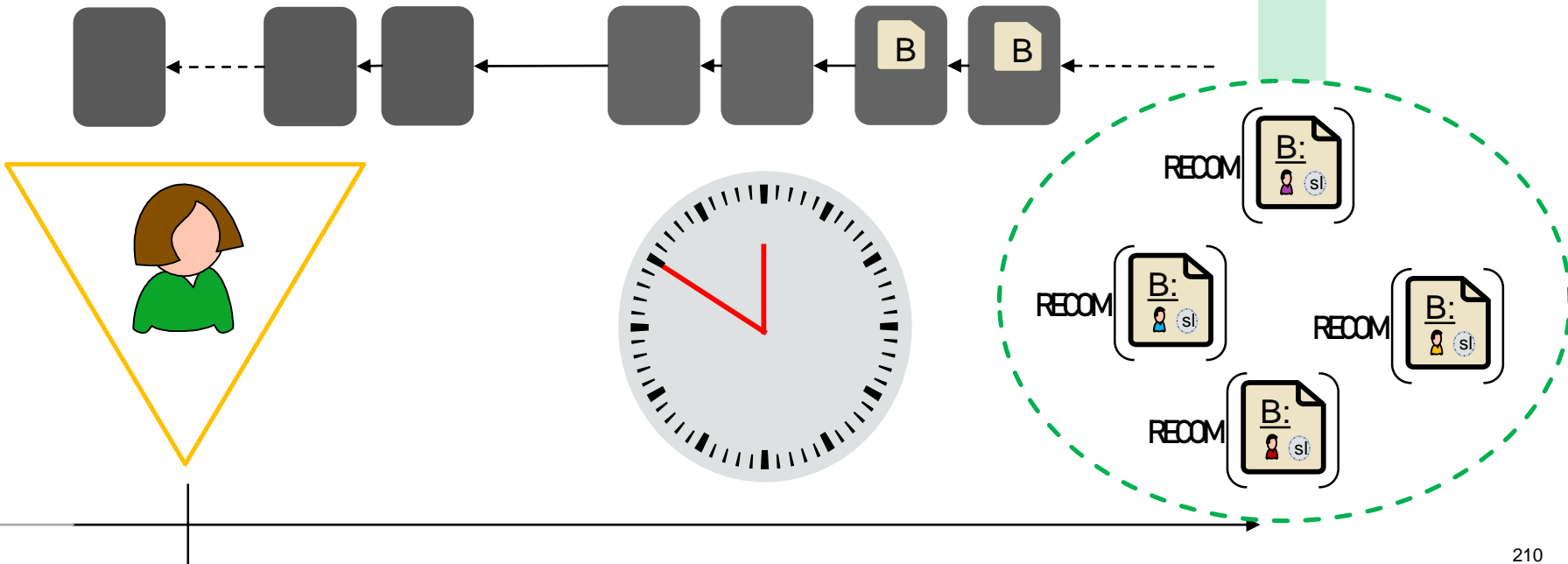
Chronos: Joining Procedure



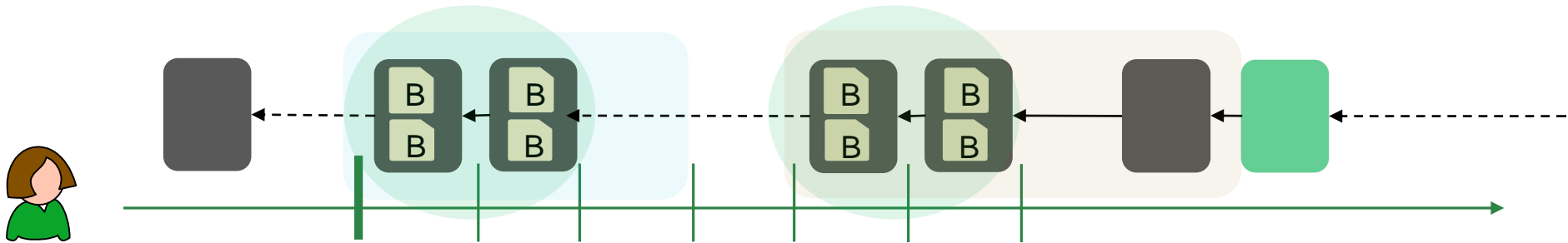
Chronos: Joining Procedure

Required Beacon Properties:

- **Fresh information:** Only generated after becoming online.
- **Validated and filtered** w.r.t. **fresh lottery.**
- Contained in common prefix.

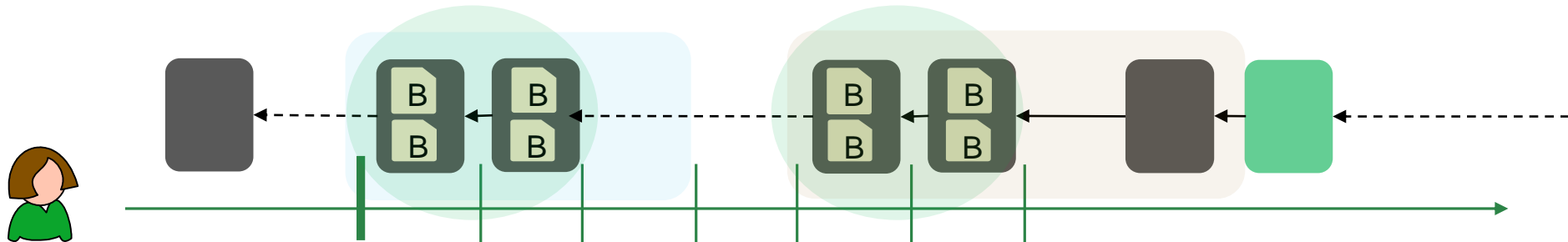


Chronos - Summary



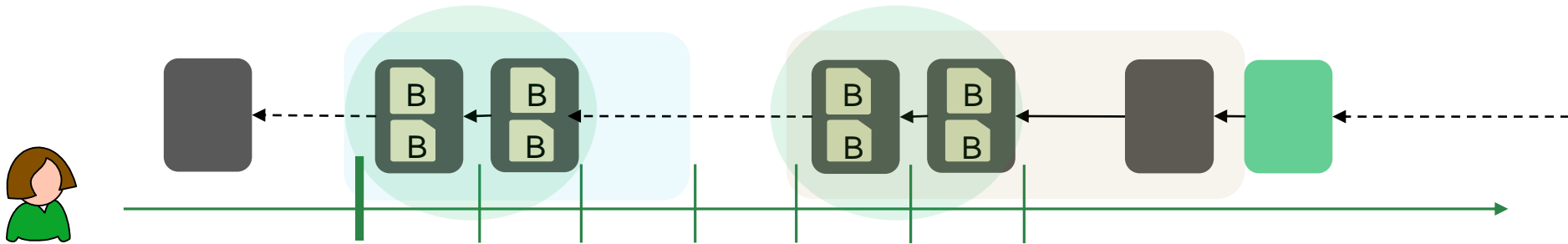
- **Bootstrapping the local clock is possible thanks to**
 - Agreement on evidence
 - Freshness of beacons: reasoning as before to get Δ -close

Chronos - Summary



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

Chronos - Summary



- **Bootstrapping the local clock is possible thanks to:**
 - Agreement on evidence
 - Freshness of beacons: reasoning as before to get Δ -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 res

Playing With Ouroboros

Check out the interactive Ouroboros animation:

ouroboros.iohk.io

End of the Tutorial – Thank you!

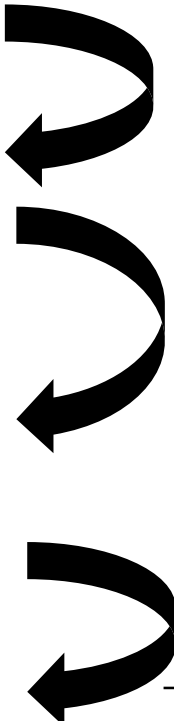
Ouroboros
“Classic”
(Crypto 17)

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:

- Classic:** A. Kiayias, A. Russell, B. David, R. Oliynikov: Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. Crypto 2017
- Praos:** B. David, P. Gaži, A. Kiayias, A. Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Eurocrypt 2018.
- Genesis:** C. Badertscher, P. Gaži, A. Kiayias, A. Russell, V. Zikas. Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. CCS 18.
- Chronos:** C. Badertscher, P. Gaži, A. Kiayias, A. Russell, V. Zikas. Ouroboros Chronos: Permissionless Clock-Synchronization via Proof-of-Stake. ia.cr/2019/838
- Crypsinous:** T. Kerber, A. Kiayias, M. Kohlweiss, V. Zikas: Ouroboros Crypsinous: Privacy-Preserving Proof-of-Stake. IEEE S&P 2019.
- O-BFT:** A. Kiayias, A. Russell: Ouroboros-BFT: A Simple Byzantine Fault Tolerant Consensus Protocol. ia.cr/2018/1049

[BMTZ17]: C. Badertscher, U. Maurer, D. Tschudi, V. Zikas. Bitcoin as a Transaction Ledger: A Composable Treatment. Crypto 17.

[GKR18]: P. Gaži, A. Kiayias, A. Russell. Stake-Bleeding Attacks on Proof-of-Stake Blockchains. Crypto Valley Conference 2018.

GKL Analysis: J. Garay, A. Kiayias, N. Leonardos: The Bitcoin Backbone Protocol: Analysis and Applications. Eurocrypt 2015.

PSs Analysis: R. Pass, L. Seeman, A. Shelat: Analysis of the Blockchain Protocol in Asynchronous Networks. Eurocrypt 2016.

Fruitchain: R. Pass, E. Shi: Fruitchains: A Fair Blockchain. PODC 2017

Images: <https://openclipart.org/>