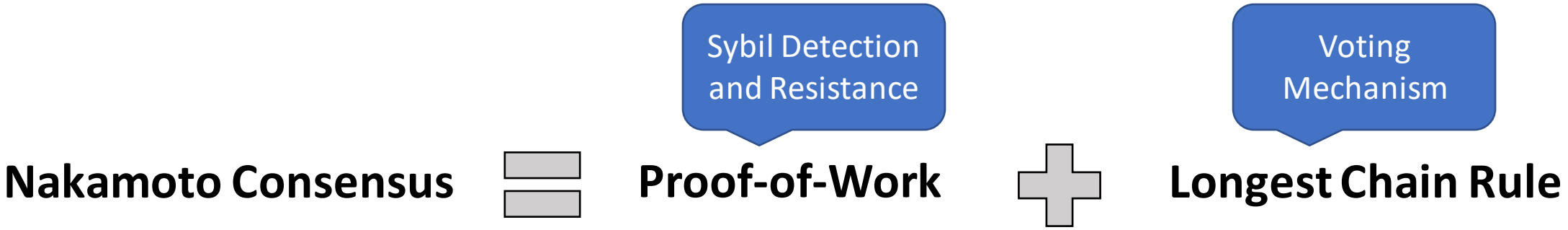
Proof-of-Stake and Ouroboros

CS839 – Kai Mast

Project Proposals

- Due on Monday, 10/11
- Structure
  - Motivation: Why is your problem important or interesting?
  - Problem description: Describe what you are trying to solve or which functionality you want to provide
  - Project Outline: Which steps are you planning to do to solve the problem
- Write a few (<5) sentences for each section

Permissionless Blockchains so far



- Proof-of-Work is not a consensus protocol by itself, but a mechanism to prevent Sybil attacks
- Sybils are fake identities set up by an attacker

Recap: Problems with Proof-of-Work

- Massive waste of electricity and hardware
- Centralization around mining pools
- Mining power can vary highly
  - Difficulty adjustment mechanisms have their own set of problems

Proof-of-Stake

Idea: Assign voting power by wealth, not mining power.

Key Problems

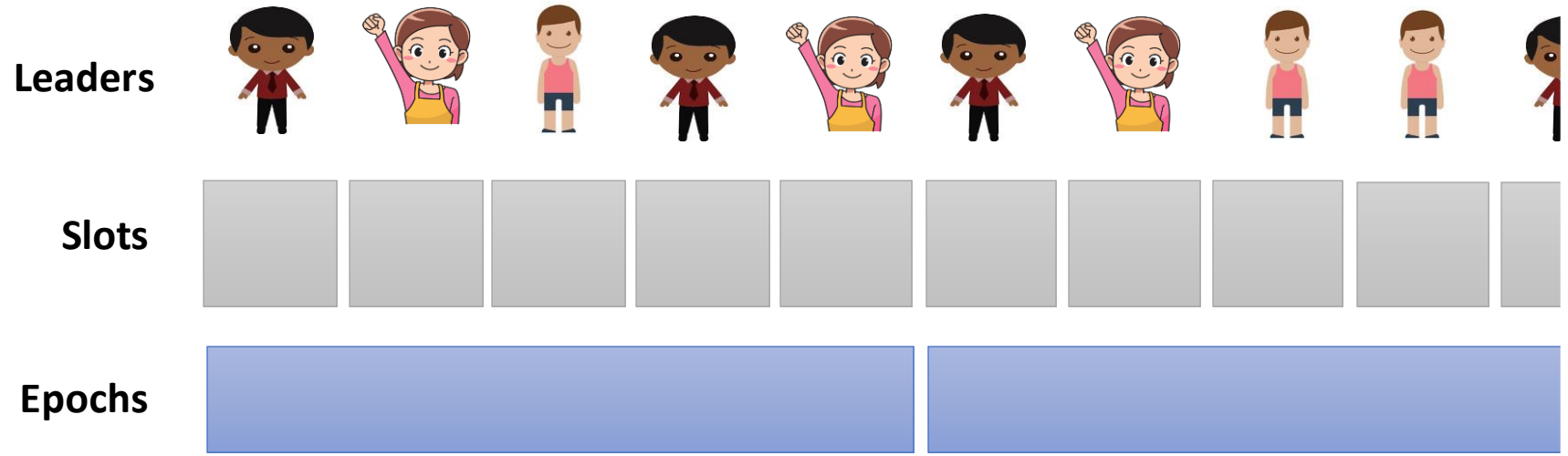
- Nothing-at-Stake: It is much easier for an attacker to issue blocks
- Grinding Attacks: Malicious stakers trying to manipulate the leader election process
- Most attacks from PoW: 51%, double spend, bribery, etc.

Ouroboros

- The first PoS protocol that is provably correct
- First presented at CRYPTO 2017
- Basis for the Cardano blockchain
- Developed by folks at IOHK and University of Edinburgh
- We only discuss the most basic variant today

CARDANO

Timing assumptions in Ouroboros



- Time is split into slots
  - Each slot has one leader*
  - Slots are grouped into epochs
- Network is synchronous
  - Each block will be visible to all correct nodes at the end of a slot
- Clocks are roughly synchronized (like in Bitcoin)

(*not true for some other versions of Ouroboros)

Leaders in Ouroboros

Election:

- All leaders for an epoch are selected at once
- Relies on randomness generated from a previous epoch (using multi-party coin tossing)

Honest Leaders:

- Will always extend the longest chain
- Create at most one block

Faulty Leaders:

- May attempt to extend multiple forks in one slot
- But can only append one block per fork

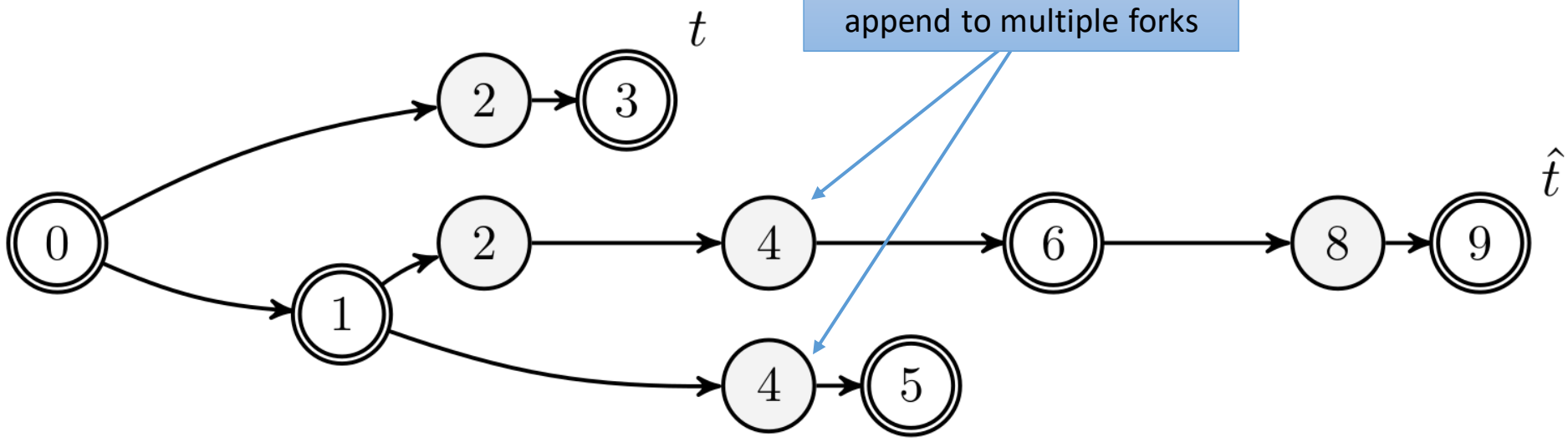
Staking Pools

Not every participant:

- may be able to run a full node
- has enough stake to become leader
- is online 24/7

Ouroboros allows delegating stake to address this.

Forks and Forkable Strings



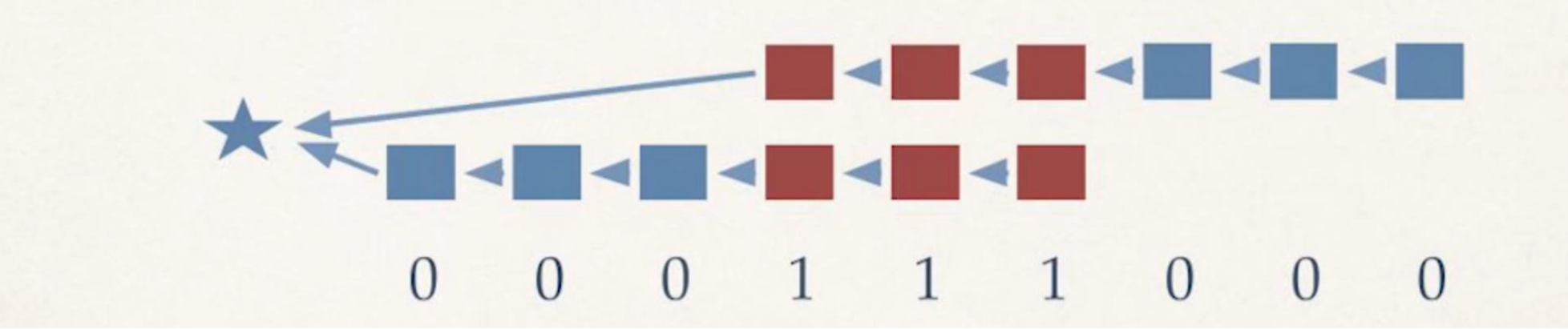
Malicious leader may append to multiple forks

$w = \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0$

String encodes leader schedule (whether leader is malicious)

Density of string is equal to the voting power of attackers (here 4/9)

Forkable Strings cont.

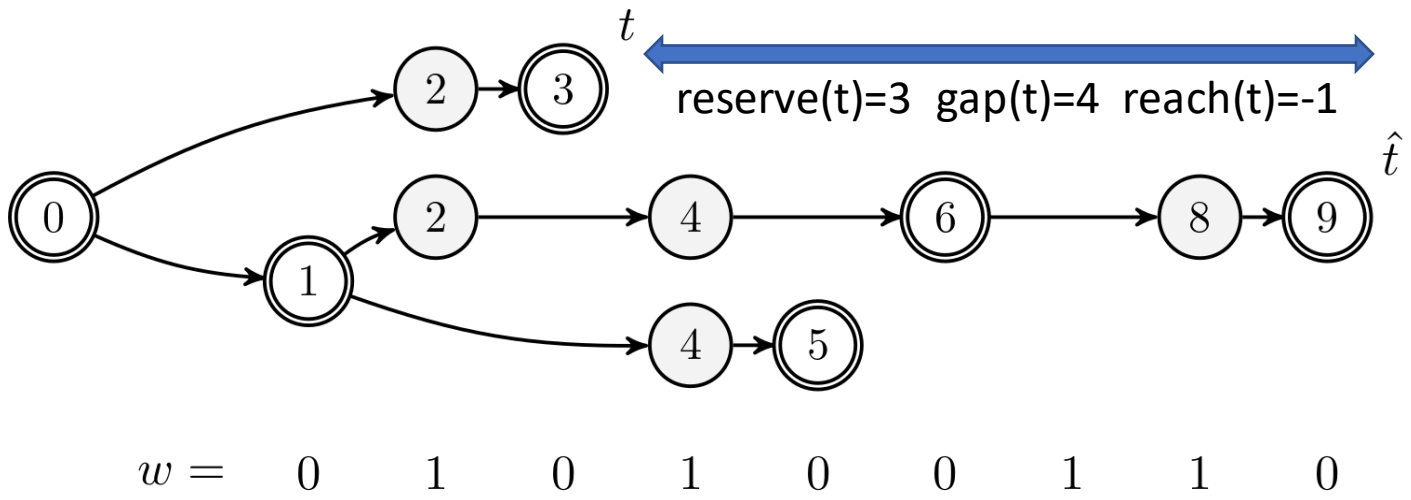


A leader schedule (or "string") is **forkable** if the adversary can produce two disjoint paths with the same length.

- Forkable strings are impossible if density is <1/3
- But we want to prevent all adversaries <½, like in Nakamoto consensus

(from Peter Gaži's talk at MIT)

Forkable String Terminology



gap(t): length difference between deepest honest node
reserve(t): remaining adversarial slot after the end of t
reach(t) = reserve(t) - gap(t)

Calculating Attacker Advantage

We only look at the two longest forks.

- ρ(w) = maximum reach for w (also just called "reach")
- margin(w) = μ(w) = second largest reach for w

Lemma 4.28. $\mathbf{m}(\epsilon) = (0,0)$ *and, for all nonempty strings* $w \in \{0,1\}^*$,

If we append a malicious slot to string margin and reach grow

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1), \text{ and}$$

In case of a benign slot there are multiple cases

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \text{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \text{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \text{otherwise.} \end{cases}$$

Calculating Attacker Advantage cont.

$$\mathbf{m}(\epsilon) = (0,0) \text{ and, for all nonempty strings } w \in \{0,1\}^*,$$

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1), \text{ and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \text{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \text{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \text{otherwise.} \end{cases}$$

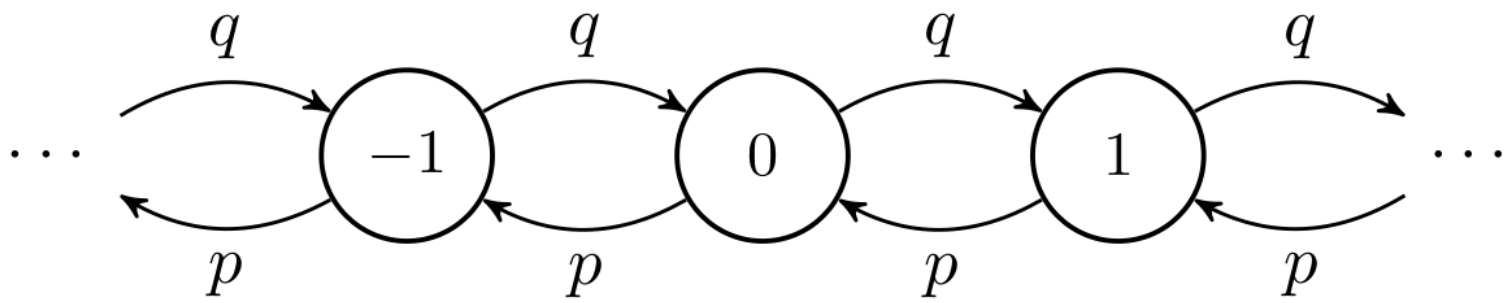
Benign leader extends longest chain, so reach decreases by one

Here, margin tracks the longest chain

Reach is never negative (we can always extend the longest chain)

Same as first case, but there's also a second longest fork controlled by the attacker

Random Walks



A one-dimensional random walk
e.g., in (simplified) Ouroboros p = (1 + Ɛ)/2 and q = 1 − p

In general:

- Models an infinite sequence of states*
- Some probability to move forward or backward*
- Useful to simulate state over time in probabilistic systems
  - e.g., also used in Avalanche's proofs

* for the one-dimensional case

Attacker Advantage with Random Walks

It is more complicated than the previous slide

- Attacker advantage is never <0
- We need to track the two longest forks

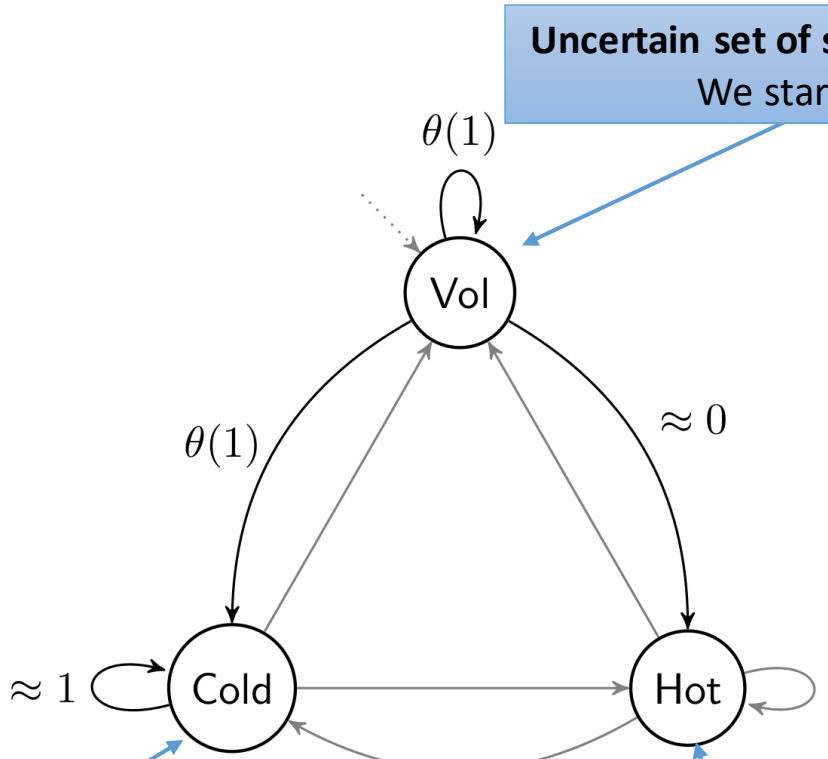
Once in the cold state, we stay there

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

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

It is virtually impossible to move to the Hot state

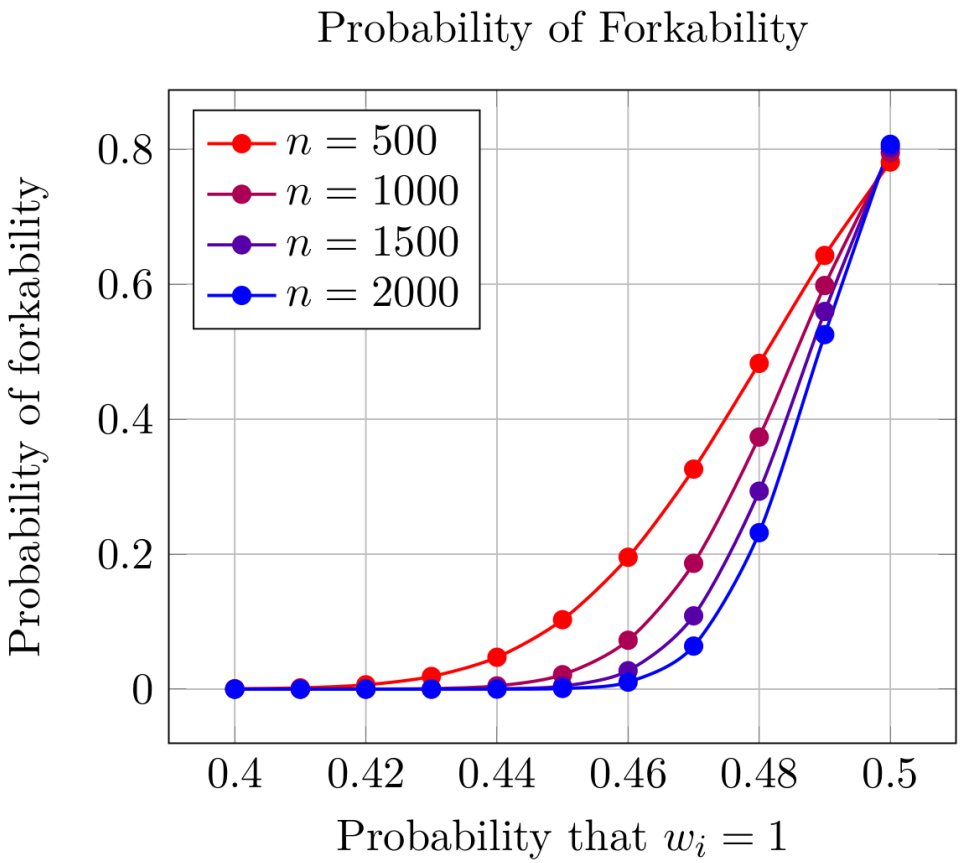
$$\Pr[\mathsf{Hot}_{t+1} \mid \mathsf{Vol}_t] \leq 2^{-\Omega(\sqrt{n})}.$$



Confirmation Delay

Goal: We want to provide "persistence" (immutability) within some constant interval of slots n (also called "k" in the paper)

- Intuitively we need to pick a bigger n with a larger fraction of attackers
- Epochs must be a multiple of n to ensure stake is stable before we reallocate voting power



Confirmation Delay

Fraction of the attacker's sake or mining power

Subset of OB General, where the attacker tries to remain hidden

Adversary	BTC	OB Covert	OB General
0.10	50	3	5
0.15	80	5	8
0.20	110	7	12
0.25	150	11	18
0.30	240	18	31
0.35	410	34	60
0.40	890	78	148
0.45	3400	317	663

Time (in minutes) after which transaction is guaranteed to be 99.9% stable

Discussion

Is Proof-of-Stake permissionless?

- Not fully permissionless: Someone needs to sell you currency for you to participate in consensus
- But most likely less centralized than any network based on PoW

Why is a known leader schedule problematic?

- Potential for Denial-of-Service and bribery attacks
- Attackers can create a more sophisticated strategy

Other PoS-compatible Protocols

Avalanche

- Works with a transaction graph and does not have longest chain rule
- Instead, transactions are confirmed using a gossip-like protocol
- Works with other Sybil detection mechanisms as well

(More about this on 10/21)

Algorand

- Use a verifiable random function to pick a committee for every slot
  - Majority of committee is guaranteed to be honest
  - Needs to run a full instance of consensus every time
- Not probabilistic
  - Once the committee has voted, a transaction is confirmed

Ouroboros Praos (not in the paper)

- Works in the partially synchronous model
- Stronger attack model
  - Attacker can corrupt other participants
- Private leader selection
  - Uses verifiable random functions like Algorand to do this
  - Can have concurrent leaders

That's all for Today

- Next time: Plasma Chains
- Project Proposals due 10/11