Blockchains & Cryptocurrencies

Proof of Stake

Image from cedricwallburger.com

Instructor: Abhishek Jain
Johns Hopkins University - Spring 2021

Recap: Proof of Work

- Bitcoin uses proof of work puzzles to prevent sybil attacks *and* implement distributed consensus
- Chance of "winning" in a block mining round proportional to one's (hash) computing power
- Security requires majority ownership of computing power to be honest

Proof of Stake

- Participants must have some "stake" (i.e., money) in the system
- Chance of "winning" in a block mining round proportional to one's current stake
- Security requires majority ownership of stake to be honest

Potential benefits

- Lower overall costs
  - No harm to the environment
- Stakeholder incentives - good stewards?
- No ASIC advantage
- 51% attack might be harder (this is debatable)

Examples of PoS based Cryptocurrencies

- Cardano
- Algorand
- Ethereum 2 (hopefully!)
- Nxt
- Neucoin
- …

Examples of PoS systems with security analysis

- Algorand [Full version: Chen-Micali'17]
- Cardano/Ourboros [Kiayias-Russel-David-Oliynykov'17]
- Snow white [Daian-Pass-Shi'17]

Agenda

- Key design considerations in PoS systems
- Closer look at **Algorand**
  - Committee Election
  - Fast Byzantine Agreement (with "player replaceability" feature)
- Attacks in PoS systems and defenses

Starting Ideas

- <u>Goal</u>: In every round, select a random "block proposer" in proportion to current stake
- Suppose a random **seed** is fixed in genesis block
  - Use a (fixed) sampling function on input seed to select block proposer weighted by stake
- <u>Challenges</u>:
  - Stake distributions change
  - Adversary can **adaptively corrupt** block proposers

A potential blueprint

- Imagine a “randomness beacon” which emits random strings at regular intervals
- Divide blockchain lifespan in “epochs”
  - In every epoch, use **new random string** (and **current** stake distribution) to select block proposers
- Challenges:
  - Where does this randomness beacon come from?
  - A**daptive security** still challenging

Ouroboros/Cardano (high-level)

- Randomness beacon via **multiparty "coin-tossing"**
  - Select "small" committee of parties with honest majority
  - Run a coin-tossing protocol with guaranteed output
- Adaptivity?
  - Original version [CRYPTO'17] does not allow "instantaneous" corruption
  - Later versions consider stronger models

Algorand (high-level)

- **“Private Sampling”** using cryptography
- Select a “committee” of potential block proposers
- Run a fast byzantine agreement protocol to agree on a block
- **“Player replaceability”** feature to address adaptive corruptions

Algorand (high-level)

- <u>Private Sampling</u>: Each party can privately determine whether it is "selected". Announce it to others with a "proof"
  - Adversary cannot a priori determine who will be selected
  - But what if Adversary corrupts parties after they announce their selection?
- <u>Player Replaceability</u>: Each selected party only needs to send "one message". After that new parties are selected
  - If adversary corrupts after announcement, its "too late"

Agenda (revised)

- **Fast Byzantine Agreement** (with player replaceability)
- Later:
  - Private sampling to elect committees
  - Other attacks and defenses for PoS systems

Fast and Furious Byzantine Agreement

Micali [ITCS'17]

Byzantine Agreement

- Consider **n** parties that have inputs $\mathbf{v_i}$
- Let **t** be the number of maximum corrupted parties
- Communication model: P2P (assume full-connectivity; synchronous)
- <u>Goal:</u> Design an interactive protocol that terminates (with high probability), where
  - <u>Agreement</u>: All honest players output the **same** value
  - <u>Consistency</u>: if all honest players started with same input **v**, then output of all honest players must be **v**

Byzantine Agreement (contd.)

- Typically **t** must be at most 1/3 (lower bounds known)
  - Can be overcome using cryptography
- Known lower bounds on round complexity
  - Typically, need rounds proportional to **t**
  - Can be overcome using randomness. Desired goal: **expected constant rounds**

Binary BA vs Arbitrary value BA

- Binary BA: Input values are {0,1}
- [Turpin-Coan'84]: general reduction to convert binary BA into arbitrary value BA
  - assuming 2/3 honest majority
  - requires only two additional rounds of communication
- <u>This talk</u>: Focus on Binary BA

Why is Byzantine Agreement Hard?

- Protocol executed over point-to-point channels
- Adversarial parties **may send different messages** (including no message) to different honest parties
- BA over broadcast channels is trivial

Byzantine Agreement (History)

- Expected constant round protocols with honest majority known
- Drawbacks:
  - Very complex designs
  - Large constants
- Recent goal (motivated by blockchains):
  - Simple designs
  - Small constants

Micali's Protocol [ITCS'17]

- Simple design via clever use of cryptography; small constant
- Assumes 2/3 honest majority
  - Reduced to 1/2 in follow-up work
  - Further improvements such as relaxing synchronicity assumptions
- Achieves player replaceability!

Micali's Protocol: Main Intuition

Consider "idealized" protocol **P(r)**, where $\mathbf{b_i}$ is the initial input of party **i**:

- Each player **i** sends $\mathbf{b_i}$ to all other players
- A new random and independently selected bit **c(r)** ***appears in sky***
- Player **i** updates bit $\mathbf{b_i}$ as follows:
  - If $\#_{i,r}(0) >= 2t+1$, set $b_i = 0$
  - If $\#_{i,r}(1) >= 2t+1$, set $b_i = 1$
  - Else, set $b_i = c(r)$

$\#_{i,r}(b)$: Number of players from which **i** received **b** in "iteration" number **r**

Quick Analysis

Assume at least **2t+1** players are honest:

- If honest players start in agreement, then they remain in agreement
- If honest players do not start in agreement, then they end in agreement (on some bit) with probability **1/2**

- <u>**Think**</u>: Why?

Implementing "coin in sky" using cryptography

Three Ingredients:

- Unique Digital Signatures: For every public key **pk** and message **m**, **only one** valid signature for **m** w.r.t. **pk**
- Hash function: Modeled as a random oracle
- Common random string R: Fixed at the start of the protocol execution, known to each party, and not controlled by adversary

Implementing "coin in sky" using cryptography

ConcreteCoin(r): Each player **i** does the following,

- Send $\mathbf{s_i = SIG_i(R, r)}$
- Compute **m** s.t. $\mathbf{H(s_m) \le H(s_i)}$ for all **i**
- Set $\mathbf{c_i(r) = lsb(h)}$, where $\mathbf{h = H(s_m)}$

Think: What is the probability that $\mathbf{c_i(r) = c_j(r)}$ for all honest **i,j** ?

Think: Why is $\mathbf{c_i(r)}$ random?

Using ConcreteCoin(r)

Replacing coin in sky with ConcreteCoin(r) in P(r):

- If honest players start in agreement, then they remain in agreement
- If honest players do not start in agreement, then they end in agreement (on some bit) with probability **1/3 (Think: Why?)**