Blockchains & Cryptocurrencies

Proof of Stake - II

Image from cedricwallburger.com

Instructor: Abhishek Jain
Johns Hopkins University - Spring 2021

Recap: 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

Recap: Agenda

- Algorand
  - Fast Byzantine Agreement (with player replaceability)
  - Private sampling to elect committees

- Other attacks and defenses for PoS systems

Recap: 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**

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 $\mathbf{b_i} = 0$
  - If $\#_{i,r}(1)$ **>= 2t+1**, set $\mathbf{b_i} = 1$
  - Else, set $\mathbf{b_i} = \mathbf{c(r)}$

$\#_{i,r}$**(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:

- <u>Unique Digital Signatures</u>: For every public key **pk** and message **m**, **only one** valid signature for **m** w.r.t. **pk**
- <u>Hash function</u>: Modeled as a random oracle
- <u>Common random string R</u>: 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 $\mathbf{s_i} = \mathbf{SIG_i(R, r)}$
- Compute **m** s.t. $\mathbf{H(s_m) <= H(s_i)}$ for all **i**
- Set $\mathbf{c_i(r) = lsb(h)}$, where $\mathbf{h = H(s_m)}$

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

<u>Think</u>: 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**

Remaining Problem

Can we simply repeat the protocol indefinitely until agreement is reached?

- The honest players do not know that agreement is reached
- Thus, the protocol may never terminate

Remaining Problem

Can we simply repeat the protocol indefinitely until agreement is reached?

- The honest players do not know that agreement is reached
- Thus, the protocol may never terminate

Idea: Simply repeat say **k = 200** times to ensure that agreement is reached, except with very small probability

Drawback: We waste many rounds since most times, agreement will be reached earlier

Micali’s Idea:

Protocol BBA* : It consists of **sequential repetitions** of **P’(r)**, where each **P’(r)** consists of three correlated executions of **P(r)**

1. Execution of **P(r)** where **c(r) = 0**
2. Execution of **P(r)** where **c(r) = 1**
3. Execution of **P(r)** where **c(r)** is implemented via **ConcreteCoin(r)**

<u>Note 1</u>: In the first two executions, a party will **terminate** if it thinks agreement is reached

<u>Note 2</u>: While the number of repetitions of **P’(r)** are not fixed in advanced, the expected number of repetitions will be 3 (will follow from protocol analysis)

Notation:

1. A party **i** may at any point send special value **b*** (and HALT) meaning that in all future steps, other parties should think of **i's** message as **b**
2. Counter γ which indicates how many times the 3-step loop has been executed. Initially set to **0**
3. **R** denotes the common random string

PROTOCOL $BBA^{\star}$

(COMMUNICATION) STEP 1. [Coin-Fixed-To-0 Step] *Each player i propagates b_i.*

1.1 *If $\#_i^1(0) \geq 2t+1$, then i sets $b_i = 0$, sends $0*$, outputs $out_i = 0$, and HALTS.*

1.2 *If $\#_i^1(1) \geq 2t+1$, then, then i sets $b_i = 1$.*

1.3 *Else, i sets $b_i = 0$.*

(COMMUNICATION) STEP 2. [Coin-Fixed-To-1 Step] *Each player i propagates b_i.*

2.1 *If $\#_i^2(1) \geq 2t+1$, then i sets $b_i = 1$, sends $1*$, outputs $out_i = 1$, and HALTS.*

2.2 *If $\#_i^2(0) \geq 2t+1$, then i set $b_i = 0$.*

2.3 *Else, i sets $b_i = 1$.*

(COMMUNICATION) STEP 3. [Coin-Genuinely-Flipped Step] *Each player i propagates b_i and $SIG_i(R,\gamma)$.*

3.1 *If $\#_i^3(0) \geq 2t+1$, then i sets $b_i = 0$.*

3.2 *Else, if $\#_i^3(1) \geq 2t+1$, then i sets $b_i = 1$.*

3.3 *Else, letting $S_i = \{j \in N : m_i^3(j) = SIG_j(R,\gamma)\}$,*
i sets $b_i = c_i^{(\gamma)} \triangleq \mathtt{lsb}(\min_{j \in S_i} H(SIG_i(R,\gamma)))$; increases γ_i by 1; and returns to Step 1.

Analysis

Claim A: *If at start of an execution of step 3, no player has halted and agreement has not been reached, then with prob 1/3, players will be in agreement at the end of the step*

Analysis

Claim A: If at start of an execution of **step 3**, no player has halted and agreement has not been reached, then with prob **1/3**, players will be in agreement at the end of the step

Protocol $BBA^\star$

(Communication) Step 1. [Coin-Fixed-To-0 Step] *Each player i propagates b_i.*

1.1 If $\#_i^1(0) \geq 2t+1$, then i sets $b_i = 0$, sends $0*$, outputs $out_i = 0$, and HALTS.

1.2 If $\#_i^1(1) \geq 2t+1$, then, then i sets $b_i = 1$.

1.3 Else, i sets $b_i = 0$.

(Communication) Step 2. [Coin-Fixed-To-1 Step] *Each player i propagates b_i.*

2.1 If $\#_i^2(1) \geq 2t+1$, then i sets $b_i = 1$, sends $1*$, outputs $out_i = 1$, and HALTS.

2.2 If $\#_i^2(0) \geq 2t+1$, then i set $b_i = 0$.

2.3 Else, i sets $b_i = 1$.

(Communication) Step 3. [Coin-Genuinely-Flipped Step] *Each player i propagates b_i and $SIG_i(R,\gamma)$.*

3.1 If $\#_i^3(0) \geq 2t+1$, then i sets $b_i = 0$.

3.2 Else, if $\#_i^3(1) \geq 2t+1$, then i sets $b_i = 1$.

3.3 Else, letting $S_i = \{j \in N : m_i^3(j) = SIG_j(R,\gamma)\}$,
i sets $b_i = c_i^{(\gamma)} \triangleq \texttt{lsb}(\min_{j \in S_i} H(SIG_i(R,\gamma)))$; increases γ_i by 1; and returns to Step 1.

Proof: Consider 5 possible cases:

1. All honest **i** update **b$_i$** according to **3.1**
2. All honest **i** update **b$_i$** according to **3.2**
3. All honest **i** update **b$_i$** according to **3.3**
4. Some honest **i** update **b$_i$** according to **3.1**, others according to **3.3**
5. Some honest **i** update **b$_i$** according to **3.2**, others according to **3.3**

PROTOCOL $BBA^\star$

(COMMUNICATION) STEP 1. [Coin-Fixed-To-0 Step] *Each player i propagates b_i.*

1.1 If $\#_i^1(0) \geq 2t+1$, then i sets $b_i = 0$, sends $0*$, outputs $out_i = 0$, and HALTS.

1.2 If $\#_i^1(1) \geq 2t+1$, then, then i sets $b_i = 1$.

1.3 Else, i sets $b_i = 0$.

(COMMUNICATION) STEP 2. [Coin-Fixed-To-1 Step] *Each player i propagates b_i.*

2.1 If $\#_i^2(1) \geq 2t+1$, then i sets $b_i = 1$, sends $1*$, outputs $out_i = 1$, and HALTS.

2.2 If $\#_i^2(0) \geq 2t+1$, then i set $b_i = 0$.

2.3 Else, i sets $b_i = 1$.

(COMMUNICATION) STEP 3. [Coin-Genuinely-Flipped Step] *Each player i propagates b_i and $SIG_i(R, \gamma)$.*

3.1 If $\#_i^3(0) \geq 2t+1$, then i sets $b_i = 0$.

3.2 Else, if $\#_i^3(1) \geq 2t+1$, then i sets $b_i = 1$.

3.3 Else, letting $S_i = \{j \in N : m_i^3(j) = SIG_j(R, \gamma)\}$,
i sets $b_i = c_i^{(\gamma)} \triangleq \mathtt{lsb}(\min_{j \in S_i} H(SIG_i(R, \gamma)))$; increases γ_i by 1; and returns to Step 1.

Proof: Consider 5 possible cases:

1. All honest **i** update $\mathbf{b_i}$ according to **3.1**
   - Agreement holds on **0**
2. All honest **i** update $\mathbf{b_i}$ according to **3.2**
   - Agreement holds on **1**
3. All honest **i** update $\mathbf{b_i}$ according to **3.3**
   - Agreement holds on **c**
4. Some honest **i** update $\mathbf{b_i}$ according to **3.1**, others according to **3.3**
   - Agreement on 0 reached with prob ½ (assuming $\mathbf{c_i}$**'s** are same)
5. Some honest **i** update $\mathbf{b_i}$ according to **3.2**, others according to **3.3**
   - Agreement on 1 reached with prob ½ (assuming $\mathbf{c_i}$**'s** are same)

Overall, when **m** is ***honest***, agreement is reached with probability at least **1/2** since $\mathbf{c_i}$**'s are same in this case**. **m** is honest with prob **2/3**, so overall agreement prob is **1/3**

Analysis (contd.)

Claim B: *If at some step, agreement holds on bit b, then it continues to hold on bit b*

Analysis (contd.)

Claim B: *If at some step, agreement holds on bit b, then it continues to hold on bit b*

Proof: If agreement held at the start of step, then all honest parties send bit **b**, which means $\#_i(b) >= 2t+1$

Analysis (contd.)

Claim B: *If at some step, agreement holds on bit b, then it continues to hold on bit b*

Proof: If agreement held at the start of step, then all honest parties send bit **b**, which means $\#_i(b) >= 2t+1$

Claim C: *If at some step, a player i halts, then agreement will hold at the end of the step*

Analysis (contd.)

Claim B: *If at some step, agreement holds on bit b, then it continues to hold on bit b*

Proof: If agreement held at the start of step, then all honest parties send bit **b**, which means $\#_i$**(b) >= 2t+1**

Claim C: *If at some step, a player i halts, then agreement will hold at the end of the step*

Proof: WLOG, suppose **i** halts in **Coin-Fixed-To-0** step. Since $\#_i$**(0) >= 2t+1**, at least **t+1** honest players sent **0**. Thus, $\#_j$**(0) >= t+1** for every other honest **j**. If $\#_j$**(0) >=2t+1**, then **j** sets b_j**=0** in step **1.1**, else it sets b_j**=0** in step **1.3**. (<u>Step 1.2 cannot be executed</u>)

Analysis (contd.)

Property 1: Consistency (if initial bit b for all honest players, then out_i=b)

Proof: Easily follows from step **1.1** or **2.2** (depending upon whether starting input was **0** or **1**)

Property 2: Agreement (out_i = out_j for all honest i,j)

Proof: Follows from Claims A, B and C

Player Replaceability

- Parties don’t have any protocol “state” across different rounds
- Hence, each round can be executed by different sets of parties
  - Parties “listen” to network
  - When a previous round is completed, check if selected (using private sampling) and participate

Algorand using Byzantine Agreement

Main Ideas

- Builds on BA protocol of Micali for consensus
- BA protocol executed between a small committee of users for scalability
- Committee chosen at random, using **cryptographic sortition** (aka, private sampling)
- Committee members change in every round of BA protocol

Main ideas (contd.)

- For every block generation round, a small committee of parties is selected at random based on user stakes
- Each selected party gets to propose a block with some associated "priority"
- Each party distributes its block together with "proof" of selection
- Parties prioritize the blocks based on proposer's "priority"
- However, different parties may have different views
- To reach consensus on same block, the committee runs BA protocol

Verifiable Random Functions

- On any input **x**, $\mathrm{VRF}_{sk}(x)$ outputs **(hash,proof)**
- **hash** is uniquely determined given **sk** and **x** but indistinguishable from random to anyone who does not know **sk**
- Given **pk** and proof, *anyone can check* that hash corresponds to **x**
- Can be built from standard cryptographic assumptions

Notation

- **W**: total amount of currency units
- **t**: threshold, denoting expected number of users selected
- **p**: **t/W**
- **w_i**: stake/money of user **i**
- **B(k;w,p)**: Prob of getting **k** successes in **w** trials, where prob of success in each trial is **p** (Binomial distribution)

$$\sum_{k=0}^{w} B(k;w,p) = 1$$

- Division of interval **[0,1)** into multiple consecutive internals

$$I_j = \left[\sum_{k=0}^{j} B(k;w,p), \sum_{k=0}^{j+1} B(k;w,p)\right)$$

Cryptographic Sortition

Sortition(sk,seed,p,w):

- VRF$_{sk}$(seed) → (hash,proof)
- j → 0
- While $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k;w,p), \sum_{k=0}^{j+1} B(k;w,p) \right)$

  j++
- Return **(hash, proof, j)**

Cryptographic Sortition (contd.)

- Any party can compute on its own whether it was selected, together with **proof**, using its **sk**
- A user **i** with $\mathbf{w_i}$ units of Algorand is viewed as $\mathbf{w_i}$ potential "sub-users". Each sub-user is selected with probability **p=t/W.** Counter **j** denotes how many selected.
- The value of hash determines priority

Consensus with player replaceability

- For each round of the BA protocol, a fresh set of users is chosen, also using cryptographic sortition
- All users can passively participate in the protocol by listening to the gossip network. Whenever selected for a round, they send a message based on what they heard so far on the network
- BA protocol has player replaceability; therefore using different users in each step is possible

Security Challenges

- For BA consensus, high majority of players must be honest
- Why can’t adversary simply corrupt all the committee members?
- **Main Idea**: Committee members for any step are disclosed only when they send their respective messages. If adversary corrupts now, its too late. The messages are already sent.

Security Challenges (contd.)

- How to select the threshold t?

- Use a threshold such that:
  - #good > threshold: for agreement
  - ½ #good + #bad <= threshold: to avoid forks

Other Points:

- The seed (used in sortition) has to be chosen carefully. Initially, it is set to be a common random string; later, for each round r, seed is determined from seed for round **r-1** by using $\mathbf{VRF_{sk}}$ of the block proposer in round **r-1**

- What are the chances of forks? – Forks can happen with some probability (if network has weak synchrony), but a recovery process can be used to eliminate fork assuming there is a strong synchrony period, using same BA procedure

Algorand: Summary

- High throughput: ~1 min to confirm transactions vs an hour in Bitcoin
- Public ledger with low probability of forks
- Assumes 2/3-honest stake majority
- Uses a gossip communication protocol