Algorand: Scaling Byzantine Agreements for Cryptocurrency

Gilad et al. SOSP’17
Presenter: Qingzhao
Limitation of Bitcoin

- Energy consumption
  - Mining
  - 0.2 percent of global electricity use.
- Long confirmation time
  - Forks
  - More than 10 minutes for a transaction to be included in a block.
Improvements of Algorand

- Byzantine Agreement (BA) vs PoW
- Benefits
  - No mining
  - Few forks
  - Low latency
Challenges of Algorand

- **Scalability**: BA is not scalable
  - Random small committees
  - New BA (voting)
- **Sybil attack**: Pseudonyms
  - Proof of Stake
- **Denial of service**: The committee is the weakness
  - Cryptography
  - Frequent committee change
Definitions

● **User**
  ○ Each user has a private/public keypair as his identity.
  ○ Each user possesses an amount of currency.
  ○ User could be malicious

● **Transaction**
  ○ The transaction records an unit of payment, signed by sender.

● **Block**
  ○ The block contains a set of transactions; the unit of Blockchain

● **Blockchain**
  ○ Collection of blocks

● **Round**
  ○ One block is updated to the ledger
Assumptions

- Safety under weakly synchronous network
- Liveness under strongly synchronous network
- Resistance of 1/3 malicious money
- Synchronized global clock
Overview

- **Gossip network**
  - Broadcast transactions etc to the system *(same as Bitcoin)*

- **Cryptographic Sortition**
  - Randomly select a group of users (as a block proposer or committee member)

- **Block Proposal**
  - The proposers propose blocks

- **Byzantine Agreement**
  - Committee members vote for values
  - All correct users agree on a block (or empty block)
Cryptographic Sortition

● Target
  ○ Select a random set of users according to users’ weight. **Weight = Stake**
  ○ More weight, more probability to be selected.

● Steps
  ○ Each user can run **sortition** to learn whether himself is selected.
  ○ Other users can **verify** that the user is selected.

● One tool
  ○ Verifiable Random Function (VRF)
    ■ Use a **private key** and a **seed** to generate a **random value (hash)** and a **proof**.
Cryptographic Sortition: Sortition

**Inputs**
- Private key of user A
- Role
  - Block proposer
  - Committee member
- Weight of user A
- Public random seed

**Outputs**
- A hash
- A proof
- Success (Selected as the role) or not

User $A$

\[ p = f(\text{weight}) \]

Hash, Proof = VRF(sk, seed)

Hash < $p$

Success

Otherwise

Failure

Proof
Cryptographic Sortition: Verification

- **Inputs**
  - Hash & proof
  - Public key of user A

- **Checks**
  - VRF verification
  - Hash < threshold

- **Outputs**
  - The user A is selected or not
Block Proposal

- **Role selection**
  - User runs cryptographic sortition to be selected as a block proposer.

- **Block generation**
  - Generate a block containing transactions.
  - Add the **proof** of sortition into the block.

- **Broadcast the block**
  - Broadcast the block through gossip network

---

A: I am a block proposer!

Block:
- Hash
- Proof
- PK_A

B: The block is indeed from a block proposer!
Block Proposal

● Problem
  ○ There are a lot of proposers
  ○ Lots of different blocks are proposed in one round

● Block Priority
  ○ Hash(sortition output)

● Benefits
  ○ Fewer conflicts: honest users vote for the block with the highest priority
  ○ Less communication cost: abandon low priority blocks while gossiping
Byzantine Agreement (BA)

- Two phases of consensus
  - Reduction:
    - Two options: a highest-priority block and an empty block
  - BinaryBA
    - Which option

- Each step has one voting process
BA: Voting

- **Sending votes**
  - Sender: Committee members
    - each step has different committees (sortition)
  - Gossip a vote \(<pk, \text{signed}<\text{round, step, hash, proof, } H(\text{last_block}), \text{value}>>\)

- **Counting votes**
  - Collect incoming votes with specific \(<\text{round, step}>\)
  - Validate votes
    - Valid signature?
    - Extending the chain? \(H(\text{last_block})\)
    - From a committee member? (sortition)
  - Count votes \(> 2/3 \times \text{committee size}\)
    - Otherwise, timeout
BA: Reduction

- **Two steps**
  - Vote1: Vote for a block hash
  - Vote2: If Vote1 timeout, vote for an empty block hash
  - Vote2: Otherwise, vote for the voted block hash.

At most one non-empty block is returned by honest users.
BA: Binary Agreement

- Reach consensus on one of
  - the block hash from Reduction
  - Hash of empty block
BA: Binary Agreement

● **Problem**
  ○ If some users returned, there may not be enough votes in the next step.

● **Solution**
  ○ If a user returned a value, he votes for the value in the next iteration (2 steps)
BA: Binary Agreement

- Problem: If the network is weak synchronous (controlled by adversary)
BA: Binary Agreement

- **Final** and **tentative** consensus
  - Final: No other block could be returned by honest users in this round
  - Tentative: No guarantee of safety. Wait for a final consensus block as a successor
BA: Binary Agreement

- Adversary can control users’ next vote
- Adversary can indefinitely split honest users into two groups

Diagram:

Step 1

- Malicious
  - Return empty
  - Vote empty
  - In next step

- Step 1
  - Empty
  - Hblock
  - Timeout
  - Vote hblock
  - In next step
BA: Binary Agreement

- Introduce a random process to determine the vote in the next step
  - Common Coin
    - Same for all users
    - Probability distribution {0: 0.5, 1: 0.5}
Evaluation

● Experiment setup
  ○ 1000 AWS EC2 m4.2xlarge VMs
  ○ 50 users per VM
  ○ 1MB block

● Latency
● Throughput
● CPU, bandwidth, storage cost
● Resistance of misbehave
● Reasonable timeout
Evaluation: Latency

- Fixed expected **committee size** (2000)
- Fixed expected number of block proposers
- Change total number of users

*Figure 5*: Latency for one round of Algorand, with 5,000 to 50,000 users.
Evaluation: Throughput

Throughput = 1 / latency * block size

- BA latency is independent with throughput
- Block Proposal depends on throughput (bandwidth)

**Figure 7**: Latency for one round of Algorand as a function of the block size.
Conclusion

● **Strength**
  ○ Make BA-based blockchain scalable (BA + VRF)

● **Weakness**
  ○ Lots of configurations (committee size, timeout, thresholds)
  ○ Lack of incentive analysis