Algorand: Scaling Byzantine Agreements for Cryptocurrencies

Presented by:
Jeremy Lin and Daniel Scalettar

Algorand Introduction

Cryptocurrency Problems

- Trade-off between latency and transaction confidence
- Simultaneously achieving decentralization, scalability, and security

Algorand

- Latency on order of a minute while scaling to high numbers of users
- Decentralized, scalable, and secure?

Presentation Outline

High level overview

- Blockchain Trilemma
- Blockchain Consensus Protocols
- Algorand Consensus Protocol
  - Cryptographic Sortition
- High Level Visualization

Low level details

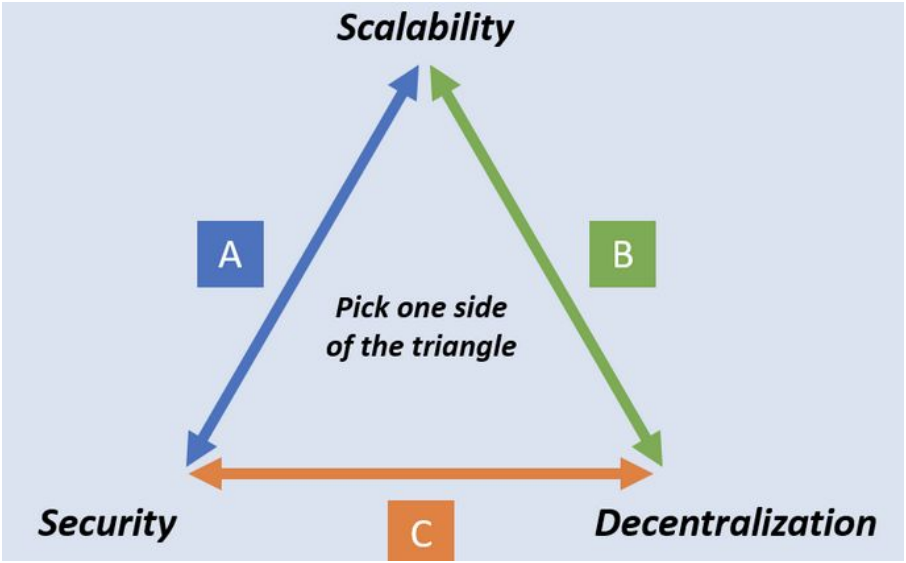
- Gossip Network
- Selection Procedure
- Block Proposal
  - Block format
- BA★
  - Reduction
  - Binary Agreement

Blockchain Trilemma

Can have at most two of the three:

- Security
- Scalability
- Decentralization

All three are important.



Is it really impossible to achieve all three?

Image credit: https://medium.com/logos-network/everything-you-know-about-the-scalability-trilemma-is-probably-wrong-bc4f4b7a7ef

Blockchain Fundamental Questions

Two main questions

- How do we make the blockchain tamper-proof?
  - One-way hash function
- How do we determine the next block?

Methods for Choosing Next Block

- Proof of Work (PoW): Solve a riddle to gain the right to add the next block
  - Problems: Slow, expensive, tends towards centralization (large mining pools)
- Delegated Proof of Stake (DPoS): Small number of users delegated (placed in charge) to determine next block
  - Problems: Centralized
- Bonded Proof of Stake (BPoS): Any number of users can participate by locking in money into the system. Influence over determining the next block proportional to money staked. Poor behaviour is punishable.
  - Problems: Vulnerable to wealthy malicious actors

Algorand: Pure Proof of Stake

- No slow, expensive riddles like in PoW
- Not centralized like in Delegated PoS or PoW (centralized mining pools)
- Money always available, not locked up like in Bonded PoS

Algorand Phases (High Level)

Phase 1: Propose

1. A token is randomly selected from all tokens in system
2. Selected token’s owner proposes and propagates a block to the other users

Phase 2: Agree

1. 1000 tokens are randomly selected from all tokens to form a committee
2. Committee members broadcast a message telling other users they have been selected
3. Committee members agree on and sign block sent by proposer in phase 1

Committee Selection

Single company or small group? Centralization

Individual users figure out who to trust? Difficult and time-consuming

Committee members choose themselves! Algorand's solution

- Each user runs a private lottery on their local machine to determine if that user is selected.
- If selected, the user propagates the winning result, proof of its validity, and a vote on whether to approve the block.

Cryptographic Sortition

Verifiable Random Function (VRF)

- ***Keygen(r) → (VK, SK)***
- ***Evaluate(SK, X) → (Y, ρ)***
- ***Verify(VK, X, Y, ρ) → 0/1***

Private Lottery (each user runs own instance on local machine)

For block **r**

- Use **Evaluate(SK, Qr) → *(Y, ρ)*** to determine if won lottery to join the 1000 user committee

Algorand Overview (High Level)

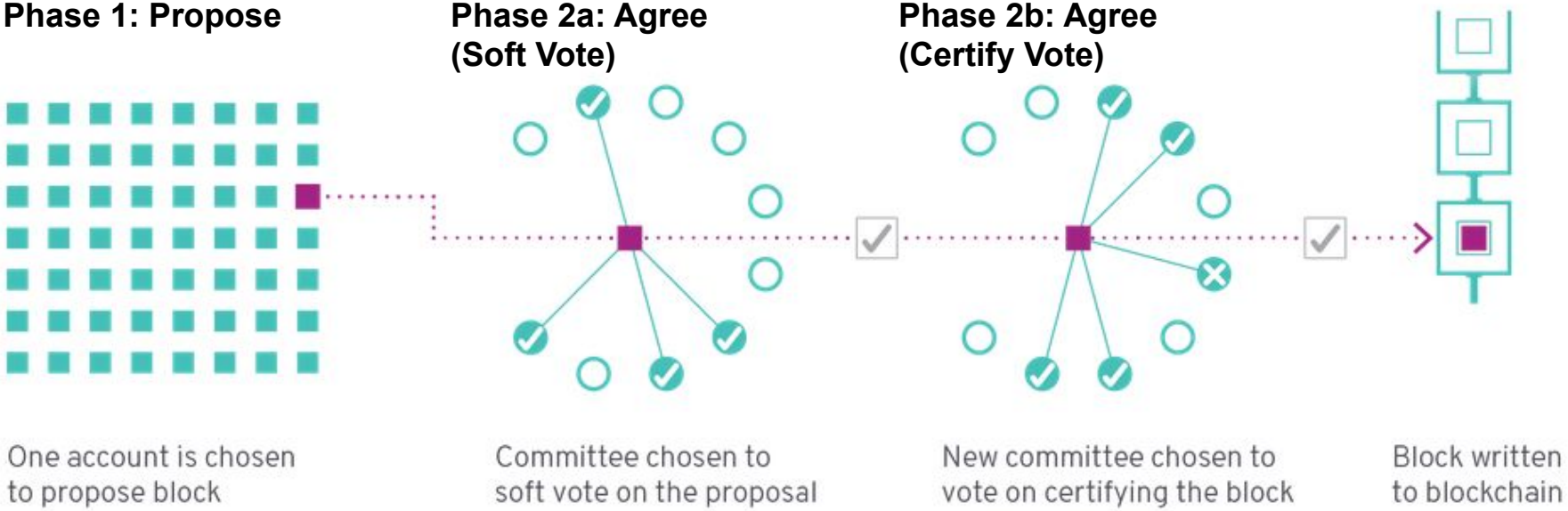
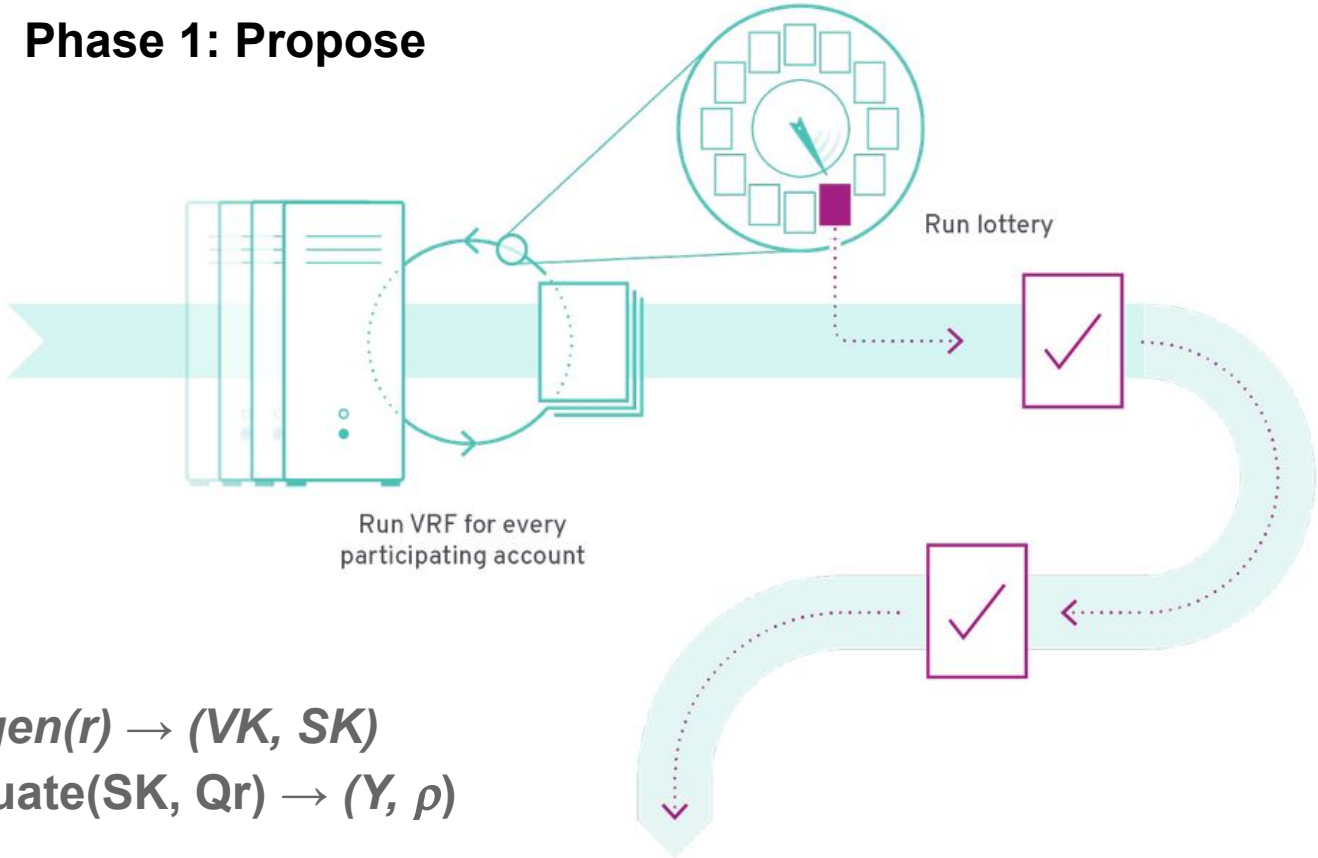


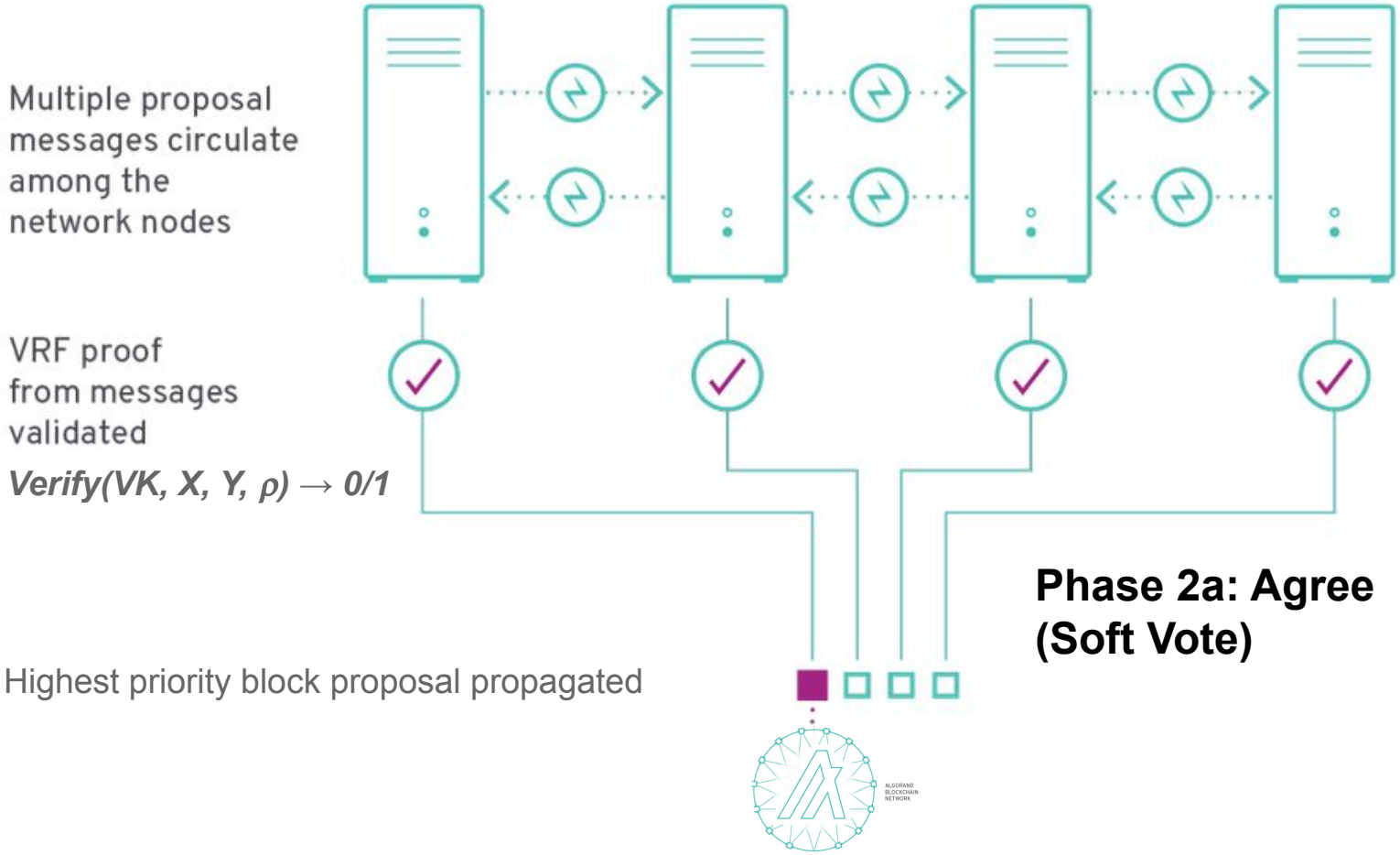
Image credit: https://info.binance.com/en/research/ALGO-2019-06-21.html

Phase 1: Propose



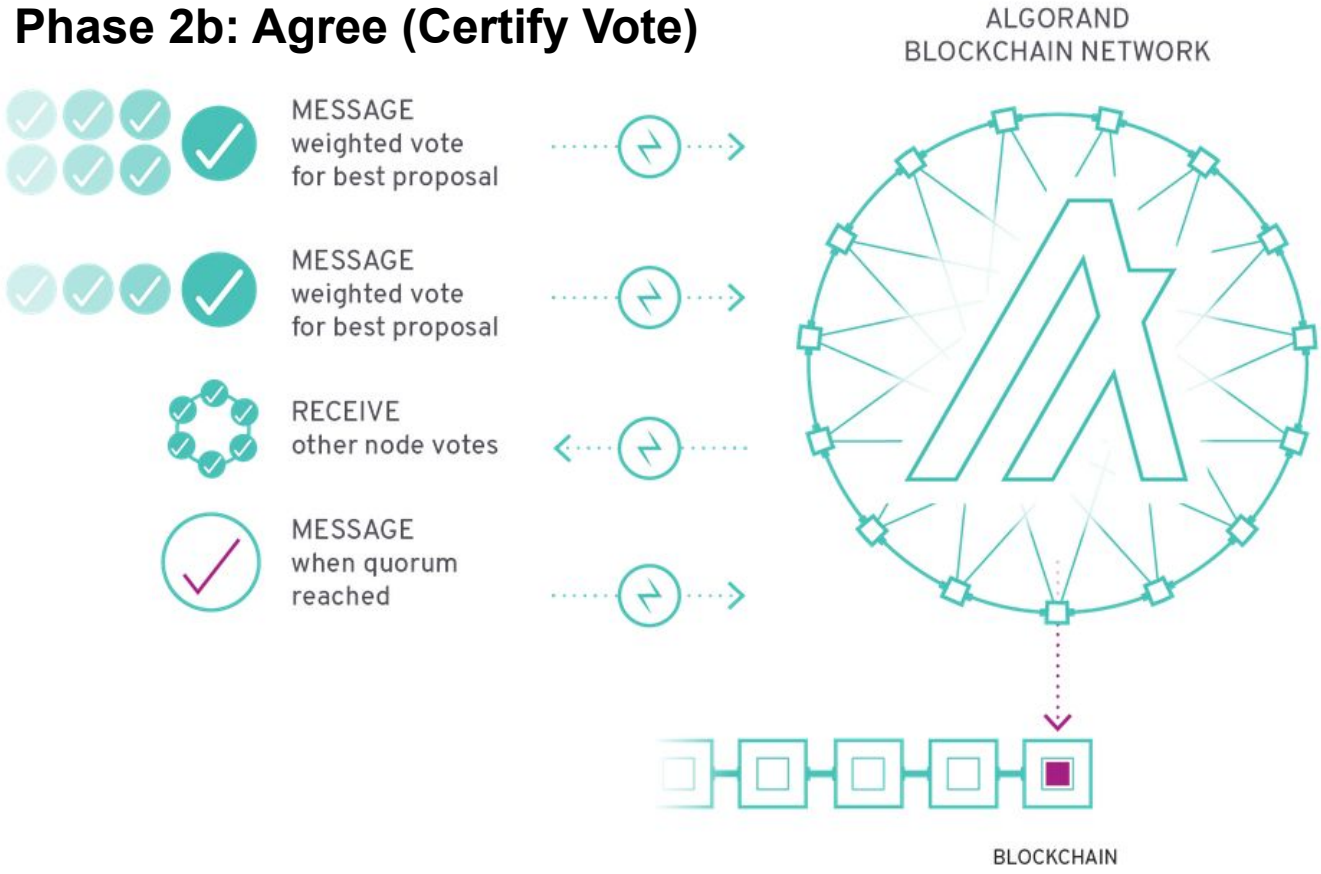
- *Keygen(r) → (VK, SK)*
- **Evaluate(SK, Qr) → *(Y, ρ)***

Image credit: https://info.binance.com/en/research/ALGO-2019-06-21.html



Phase 2a: Agree (Soft Vote)

Phase 2b: Agree (Certify Vote)



Blockchain Trilemma Revisited

Can only have 2 of the 3?

- Decentralized
- Scalable
- Secure

Algorand

- Decentralized
- Scalable
- Secure

Algorand Assumptions

- Honest users run bug free software
- Honest users hold majority of the money (Greater than 2 / 3)
- Adversaries can’t corrupt a large number of users to hold more than 1 / 3 of the money
- Loosely synchronized clocks across all users. (most honest users start recovery protocol at approximately the same time)

Safety and Liveness assumption

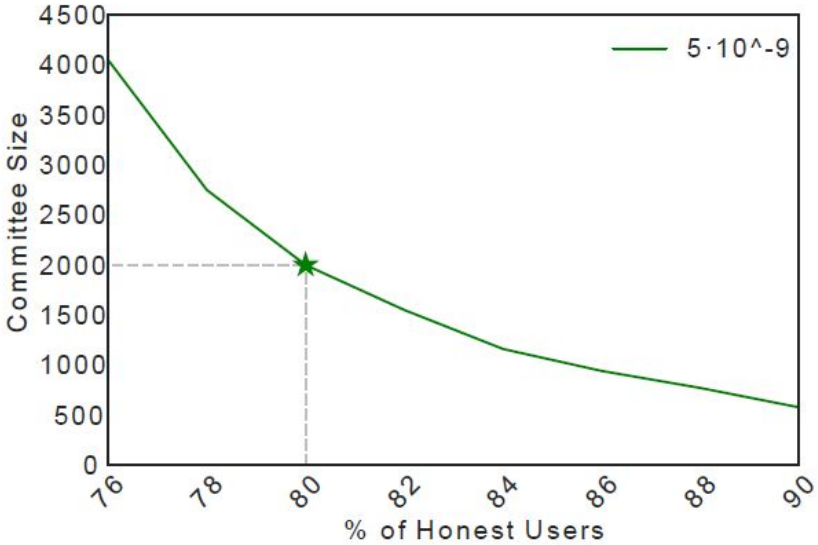
- To achieve liveness assumes “strong synchrony”
  - Most honest users messages sent will be received by most other honest users within a known time bound (no network partitions)
- To achieve safety assumes “weak synchrony”
  - Network can be asynchronous - adversary controls for a long but bounded time after which there is a reasonably long period of time where the network is strongly synchronous
  - Formally: every period of length b (day or week), there must be a synchronous period of length s < b (s of a few hours)

Algorand Challenges - avoid Sybil attacks

- Solution: weighted users
  - Each user is assigned a weight by Algorand based on how much of the currency they own
  - As long as a weighted fraction of the users are honest (greater than 2 / 3) the Byzantine Agreement protocol BA★ can guarantee consensus

Algorand Challenges - Scale to millions of users

- Solution: Consensus by committee
  - A relatively small randomly selected committee of users
    - τ - expected committee size
    - T*τ - number of votes needed to reach consensus
  - Ideally T is small for liveness/latency, □T □τ



Gilad et. al. 2017

Algorand Challenges - resilient to DoS attack

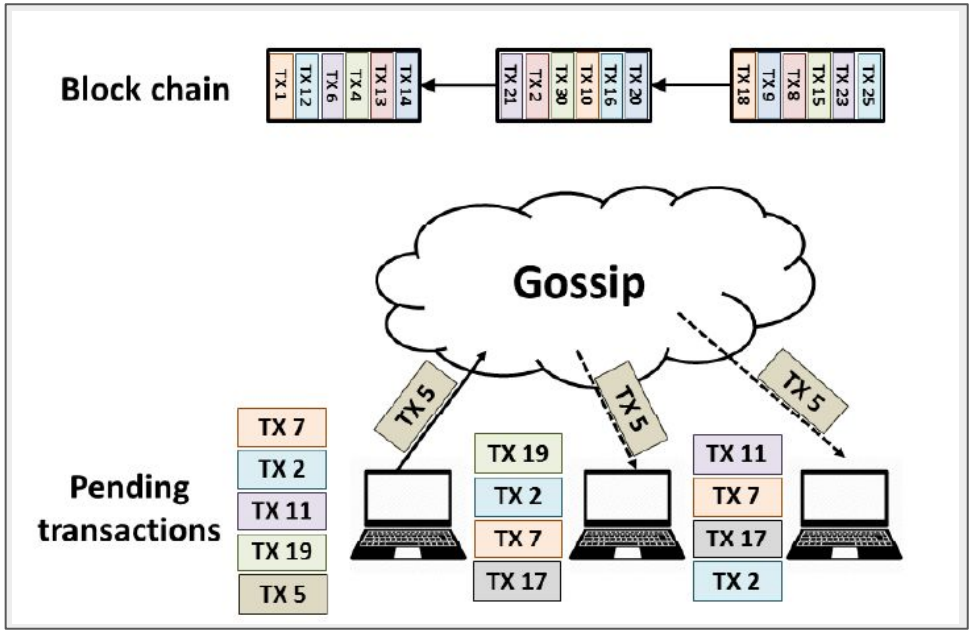
- Solution: Cryptographic sortition
  - An algorithm for choosing a random subset of users according to their given weight
  - Implemented using Verifiable Random Function (VRF) takes input and users secret key and generates a pseudo-random hash value that can be publicly verified with a variable *proof*
  - Given a set of weights w and the total weight of all users W
    - Probability user i is selected is w_i / W
    - Randomness based on a publicly known random seed
    - Each user has secret and public key pair used to generate and verify VRF output

Algorand Overview

- Gossip Network
- Selection Procedure
- Block Proposal
  - Block format
- BA★
  - Reduction
  - Binary Agreement

Gossip Network

- Users submit new transactions
- Each user selects a small random set of peers (based on weight) to gossip messages to
- Each user collects a block of pending transactions they hear about in case they are chosen to propose the next block



Overview of transaction flow in Algorand
Gilad et. al. 2017

Two types of gossip messages

1. Contains just priorities (based on weights) and proofs (from VRF) ~200 Bytes so it can propagate quickly through gossip network
    a. Enables most users to learn who the highest priority proposer is quickly
2. Contains entire block, including proposer’s sortition hash and proof

Selection Procedure

- Uses sortition with parameter *role,* what role user will play
- Algorand specifies a threshold (τ), determines the expected number of users selected for that role.
- Sortition selects users in proportion to their weight. Users may be selected more than once if they have high weight. Sortition returns a parameter, j, that indicates how many times the user was chosen. Being chosen j times means that the user gets to participate as j different “sub-users”

- Ex: If user i owns w_i units out of W, then the simulated user represents the j^{th} unit of currency i owns ($j \in \{1,...,w_i\}$), j is selected with probability τ/W

Block Proposal

- Threshold variable for proposer is set $\tau > 1$ to ensure some block is proposed
- Several proposers each gossip their own proposed block - significant communication cost
- Reduce communication cost by getting priority of block proposal from sortition hash output concatenated with sub-user index
- Highest priority of all block proposer’s selected sub users is the priority of the block
- Users discard messages about lower priority block seen so far

Block Proposal (continued)

- Selected block proposer users distributes their own block of pending transactions through gossip protocol, together with their priority and proof
- To ensure convergence on one block with high probability, block proposals are prioritized based on proposing user’s priority, users must wait for a certain amount of time to receive the block.
- How long to wait?
  - Too short - no received proposals initialize next step with empty block
  - Too long - unnecessarily increase confirmation latency
- In practice conservatively about 5 seconds.

Block Proposal - block format

- List of transactions - logically translates to a set of weights for each user's public key, along with total weight of all outstanding currency
- Metadata used by BA★ (round number, proposer's VRF-based seed, hash of previous block in ledger, timestamp when block was proposed.

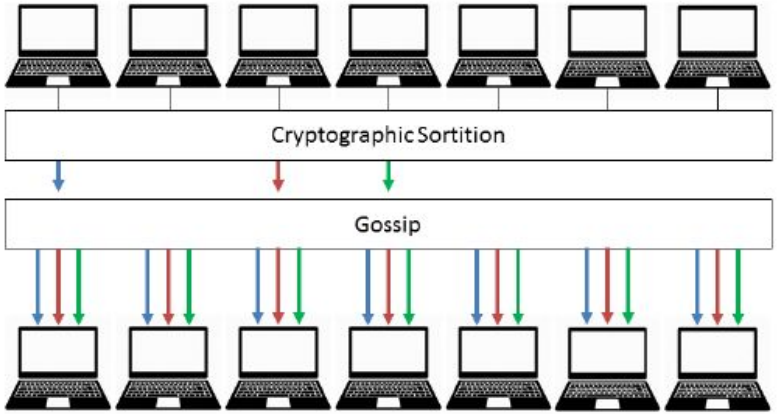
- Once a user receives block from highest-priority proposer validates contents:
  - Check all transactions are valid, seed is valid, previous block hash is correct, block round number is correct, and timestamp is greater than previous block and timestamp is within an hour of current time.
  - If any checks fail, empty block is passed to BA★

BA★ - Byzantine Agreement Protocol

- Consensus on one of the pending blocks
- Communicates with gossip protocol
- Keeps no secrets (except secret keys).
- Produces two kinds of consensus: final consensus and tentative consensus
  - Final consensus - any other user that reaches final or tentative consensus in the same round must agree on the same block value.
  - Tentative consensus - other users may have reached tentative consensus on a different block (as long as no user reached final consensus) a user will confirm a transaction from a tentative block only if and when a successor block reaches final consensus.
- Algorand confirms a transaction when the transaction’s block (or any successor block) reaches final consensus

Agreement using BA★

- Each user initializes BA★with highest-priority block that they received.
- Executes in repeated steps
    i. Each step begins with cryptographic sortition: all users check whether they have been selected as committee members in that step.
    ii. Committee members broadcast message which includes proof of selection
    iii. These steps **repeat** until in some step of BA★ enough users in the committee reach consensus.



Overview of one step of BA★
Gilad et. al. 2017

BA★ - Reduction(): step 1

- Important to ensure liveness
- Converts the problem of reaching consensus on an arbitrary value (hash of a block) to reaching consensus on one of two values.
  - Specific proposed block hash
  - Hash of an empty block
- First step: each committee member votes for the hash of the block passed to Reduction() by BA★

BA★ - BinaryBA★(): step 2

- Second step: committee members vote for the hash that received at least [T*τ] votes in the first step OR the hash of the default empty block if no hash received enough votes.
- BinaryBA★() reaches consensus on one of two values: either
  - hash passed in or hash of empty block.
- BinaryBA★() relies on reduction() to ensure at most one non-empty block hash is passed to BinaryBA★() by all honest users

Bootstrapping the system

A common genesis block must be provided to all users

along with initial cryptographic sortition seed

Bootstrapping a user

Algorand generates a certificate for every block that was agreed upon by BA★. Certificate is an aggregate of votes from last step of BinaryBA★() (not including final step).

New user uses certificate to validate prior blocks. Starting with genesis block and going in order ensures user knows correct weights for verifying sortition proofs in any given round.

Conclusion / Results

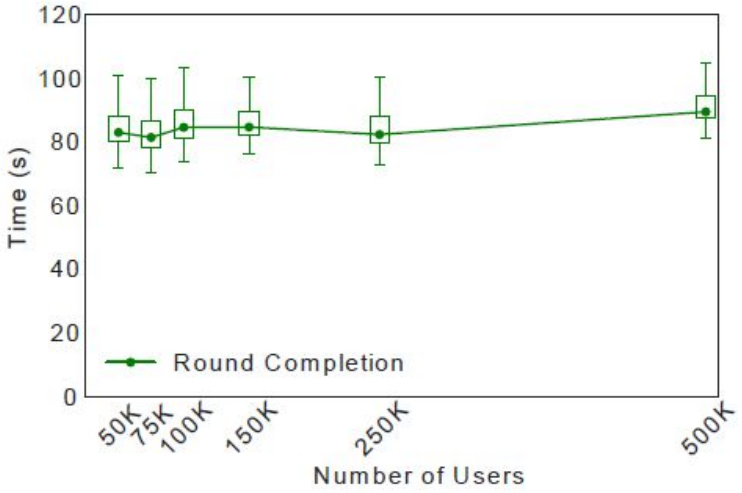


Figure 6: Latency for one round of Algorand in a configuration with 500 users per VM, using 100 to 1,000 VMs.

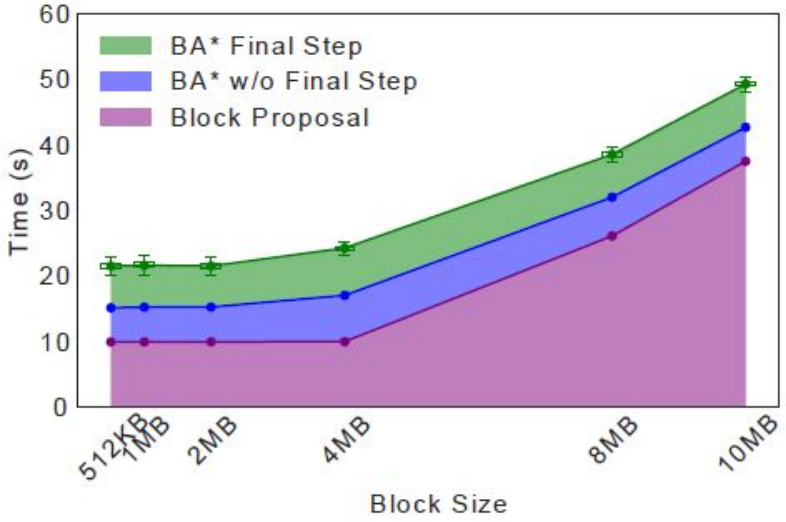


Figure 7: Latency for one round of Algorand as a function of the block size.

Citations

Silvio Micali. (2019). Algorand’s Forthcoming Blockchain Technology [slides]. Retrieved from https://www.youtube.com/watch?v=i0_ifqIowc4

Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017. 51–68. https://doi.org/10.1145/3132747.3132757

https://medium.com/algorand/algorand-releases-first-open-source-code-of-verifiable-random-function-93c2960abd61