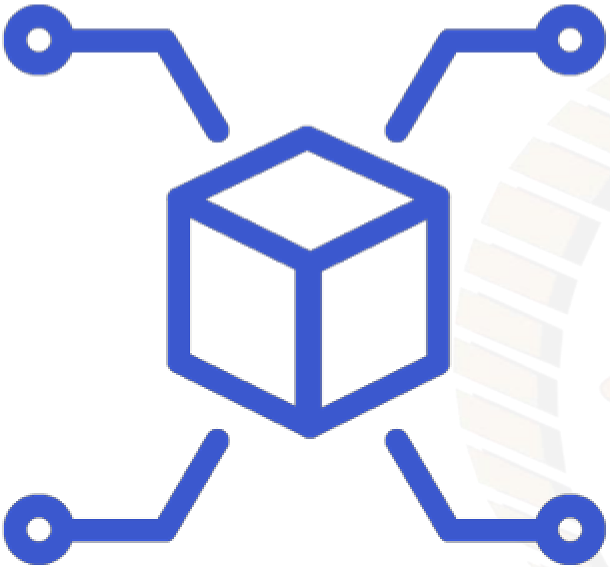
BLOCKCHAINS

ARCHITECTURE, DESIGN AND USE CASES

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

PRAVEEN JAYACHANDRAN
IBM RESEARCH,
INDIA



Gilad, Y., Hemo, R., Micali, S., Vlachos, G., & Zeldovich, N. (2017, October). *Algorand: Scaling byzantine agreements for cryptocurrencies*. In *Proceedings of the 26th Symposium on Operating Systems Principles* (pp. 51-68). ACM.

Algorand: Scaling Byzantine Agreements for Cryptocurrencies

Cryptocurrencies

- Bitcoin
- Ethereum
- Ripple
- Zcash
- Algorand

Bitcoin Overview

- Key Idea:
  - Consensus through proof-of-work (PoW)
- Communication:
  - Gossip protocol
- Key Assumption:
  - Honest majority of mining computation power

Bitcoin: Technical Limitation

- Resource wastage:
  - high computational, electricity cost
- Concentration of power
  - only ~5 mining pools control the entire system
- Vulnerable
  - easy to track miners
- Scalability
  - number of users not clear (1M, 10M, 100M??), high latency(~10minutes)
- Ambiguity
  - fork in blockchain

Algorand: Overview

- Key Idea:
  - Consensus through Byzantine Agreement Protocol
- Communication:
  - Gossip protocol
- Key Assumption:
  - Honest majority of money

Algorand: Technical Advancement

- Trivial computation
  - simple operation like add, count
- True decentralization
  - no concentration of mining pool power, all equal miners and users
- Finality of payment
  - fork with very low probability, block appears and payment fixed forever
- Scalability
  - millions of users, only network latency (~1minute)
- Security
  - against bad adversary

Algorand: Security Perspective

- Avoid Sybil attacks
  - adversary creates many pseudonyms to influence the Byzantine agreement protocol
- Resilient Denial-of-service attacks
  - operate even if an adversary disconnects some of the users

Algorand: Architecture

- Select a random user
  - prepare a block
  - propagate block through gossiping
- Select random committee with small number of users (~10k)
  - run Byzantine Agreement on the block
  - digitally sign the result
  - propagate digital signatures
- **Who select the committee??**

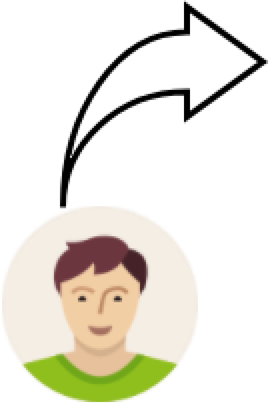
Cryptographic Sortition

- Each committee member selects himself according to per-user weights
- Implemented using verifiable random functions (VRFs)

<hash,proof> $\leftarrow$ $VRF_{sk}(x)$

- **x:** input string
- **(pk_i,sk_i):** public/private key pair
- **hash:** hashlenbit-long value that is uniquely determined by sk and x
- **proof:** enables to check the hash indeed corresponds to x

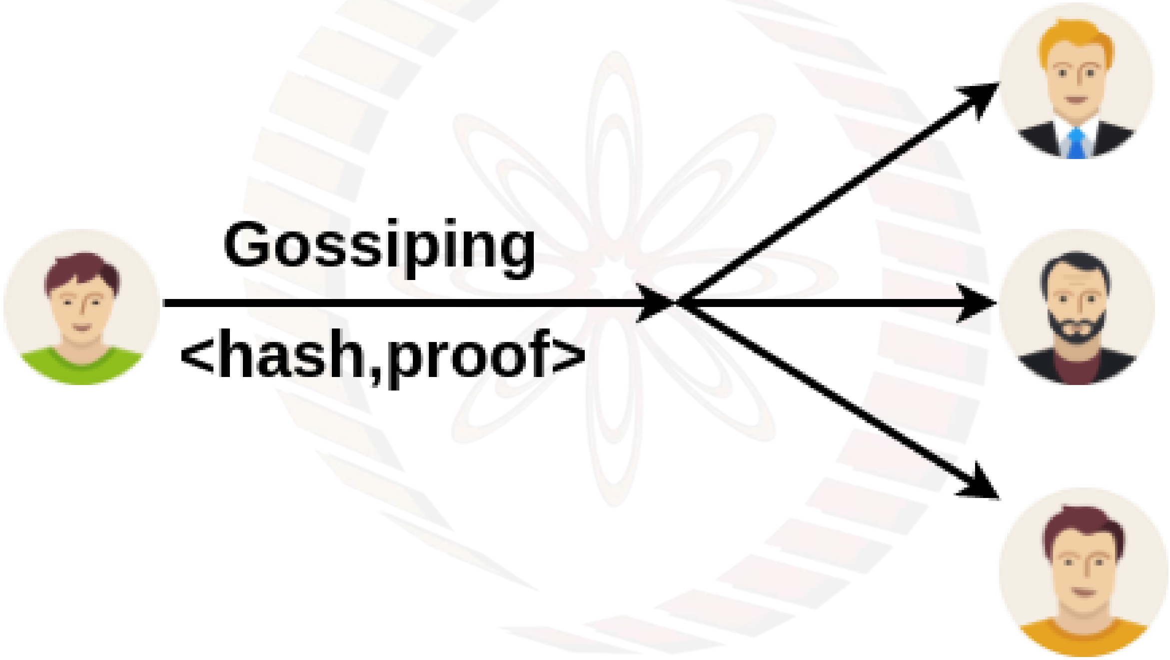
Cryptographic Sortition: Selection Procedure



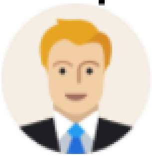
<hash,proof,j> <---
Sortition(sk,seed,threshold,role,w,W)

- **seed:** publicly known random value
- **threshold:** determines the expected number of users selected for that role
- **role:** user for proposing a block/ committee member
- **w:** weight of a user
- **W:** weight of all users
- **j:** user gets to participate as j different “sub-users.”

Cryptographic Sortition: Selection Procedure



Cryptographic Sortition: proof Verification



<j> <--- VerifySort(pk,hash,proof,seed,threshold,role,w,W)

- check if that user was selected
  - number of selected sub-users
  - zero if the user was not selected at all

Cryptographic Sortition: Seed Selection

- seed published at Algorand's round r using VRFs with the seed of the previous round $r - 1$
- Algorand reaches agreement on the block for round $r - 1$
  - everyone computes $seed_r$ at the start of round r
  - $seed_0$: chosen at random by initial participants

    $\langle seed_r, proof \rangle \leftarrow VRF_{sk_u}(seed_{r-1} || r)$
- To limit adversary's ability selection seed refreshed once every R round $seed_{r-1-(r \bmod R)}$

thank you!