Basic Principles of Security and Blockchain

Spring 2021,

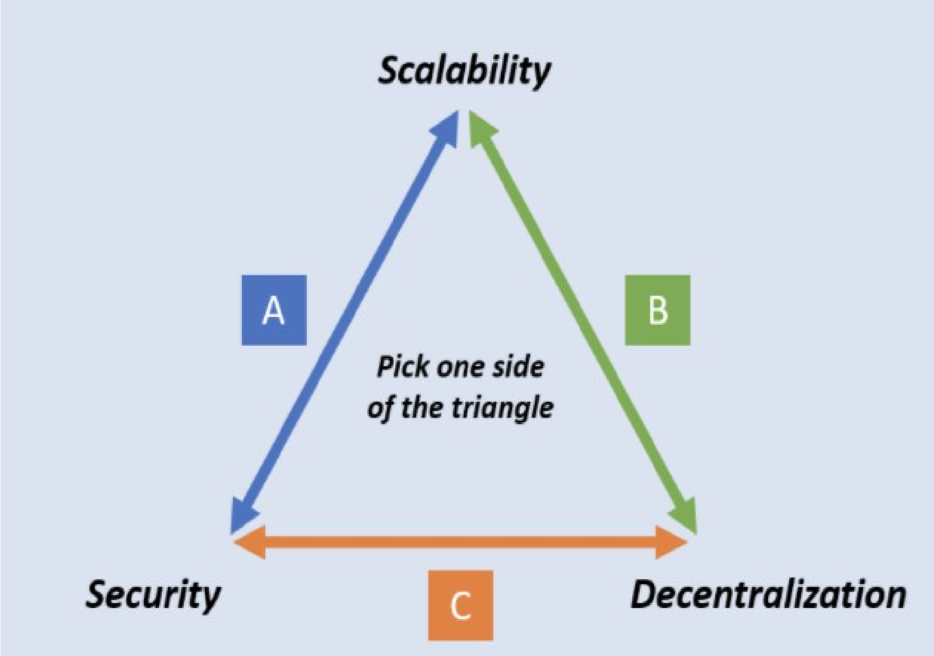
Instructor: Fabrizio Baiardi, Laura Ricci
f.baiardi@unipi.it
laura.ricci@unipi.it

Lesson 4: Algorand Pure Proof of Stake

19/3/2021

MICALI'S VIEW ON BLOCKCHAINS

- from Micali's talk: "what is really important in a blockchain? *"It is how the next block is chosen, because if this is in the hand of a few entities, or even worst, of one entity, these entities have more power than Luigi XIV"*
  - is the level of participation in the block generation that makes a blockchain really decentralized
- but, what about the famous blockchain "trilemma"?



Butlerin's trilemma

can take only one facet of the triangle

THE BLOCKCHAIN TRILEMMA

- the term has been introduced by Vitalik Buterin
  - is it possible to maximize the three desiderable attributes of the blockchain at the same time?
- decentralization
  - creating a blockchain system that does not rely on a central point of control
  - censorship resistance
- scalability
  - ability for a blockchain system to handle an increasingly growing amount of transactions per seconds
- security
  - ability of the blockchain to operate as expected and defend from attacks

THE BLOCKCHAIN TRILEMMA

- Decentralized & Secure
  - Bitcoin and Ethereum
  - not scalable at all, decentralized?
- Secure & Scalable
  - Hyperledger
  - not decentralized at all, secure because under a single administative domani
- Scalable & Decentralized
  - EOS, NEO
  - minimal censorship resistance: a few nodes control the netwok
- Algorand teams affirms that the trilemma is false and that they will solve it.

BLOCKCHAIN AND DECENTRALIZATION

- solving the trilemma: many solutions neglet the decentralization
- most of current blockchain technology fail to guarantee a good level of decentralization
  - Delegated Proof of Stake
    - select 21 nodes that will choose the next block for all of us
      - 21 nodes that are trusted, and will be trusted, by definition
    - actually, a semi-centralized solution
    - used by EOS.IO, Steemit
  - Proof of Work
    - controlled by a few big mining pools: really decentralized?
    - used by Bitcoin, Ethereum
    - this disregards the initial intention of Satoshi

IS BITCOIN REALLY DECENTRALIZED?

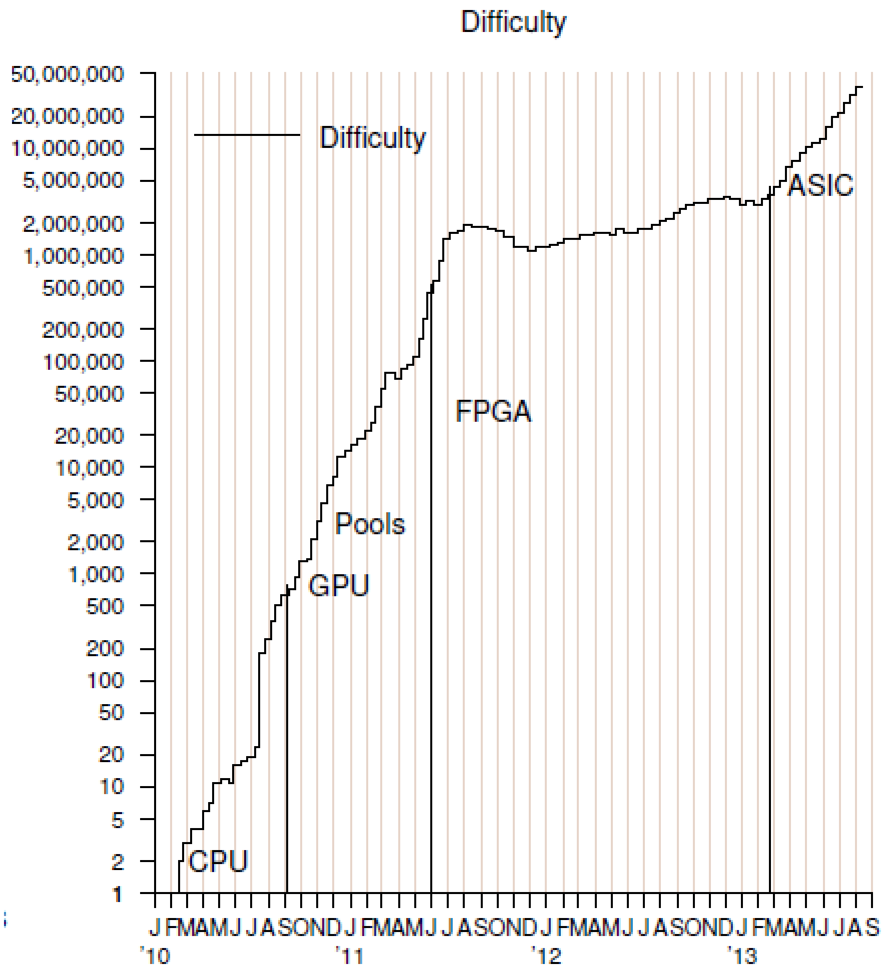
- basically, mining is SHA computation

  https://en.wikipedia.org/wiki/SHA-2#Pseudocode

- the core cycle

```
while (1)
      HDR[kNoncePos]++;
      if (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
            return;
```

IS BITCOIN REALLY DECENTRALIZED?



APPROACHES TO MINING

solo mining

mining alone

a very risky process

- mining pools
  - mining together other miners
  - miners create cartels called mining pools
  - allow them to reduce the variance of their income
  - an old idea
    - mutual insurance to lower the risk

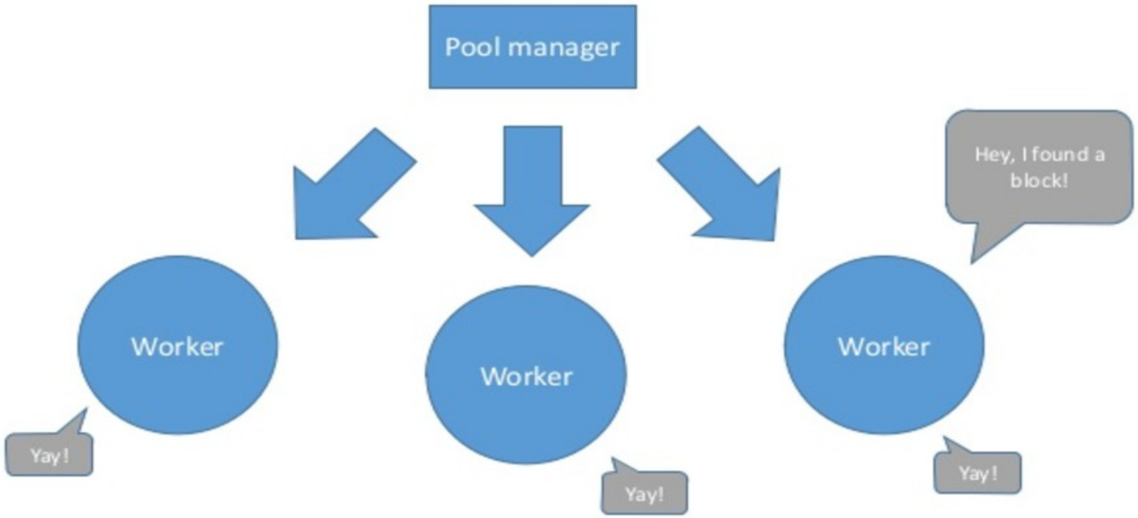
MINING TOGETHER: MINING POOLS

BitFury mining center, Republic of Georgia

MINING POOLS

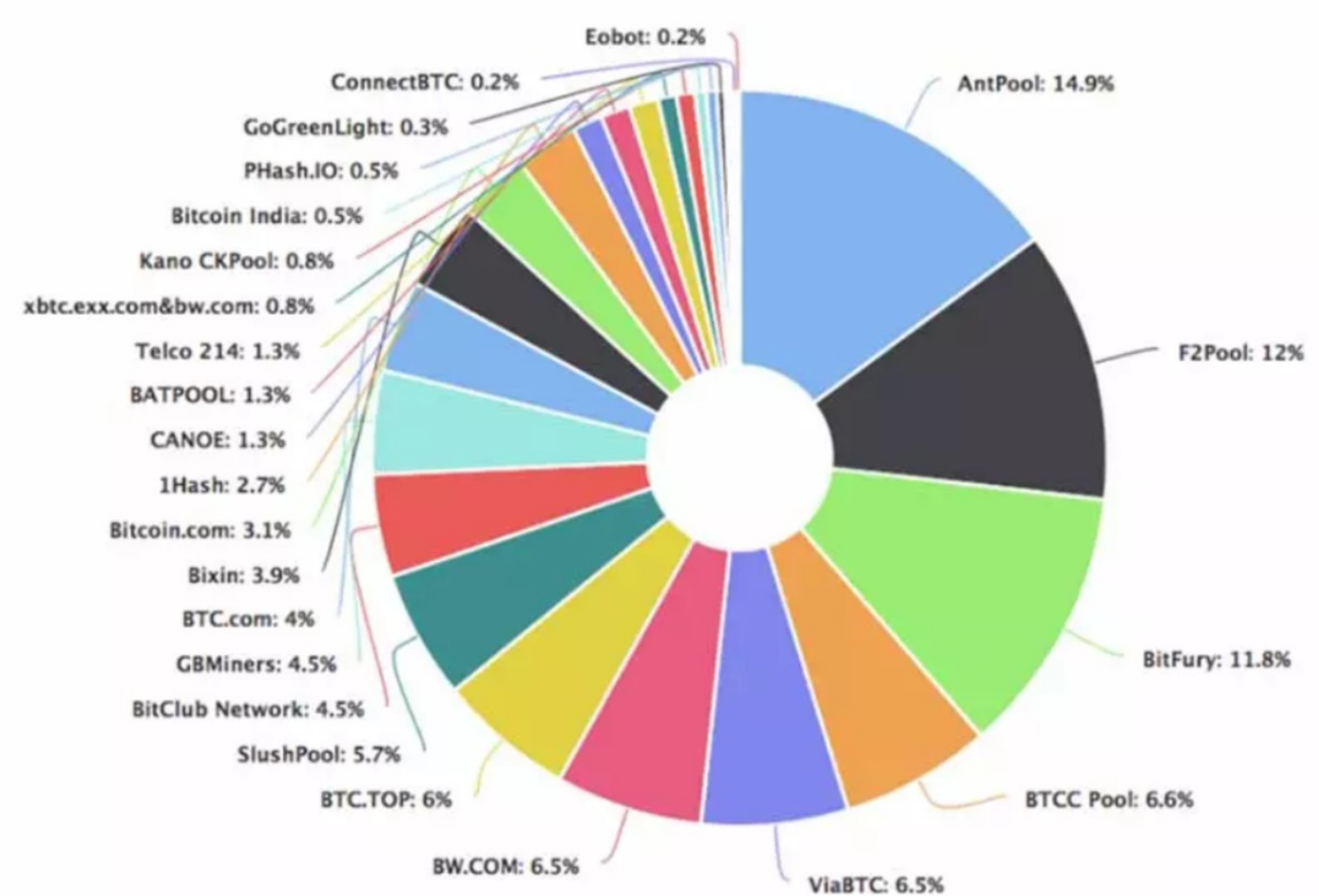
- a pool: a group of small/larger miners working together
- the mining pool may operate:
  - through a centralized mining operator
    - miners should trust the pool manager
  - in a p2p way
    - use a private blockchain to manage the pool

CENTRALIZED MINING POOLS



- the pool manager
  - sends blocks to all the miners
  - distributes revenues to members based on the work they have performed
  - may take a cut for itself (a fee)
  - must be trusted by everyone

IS BITCOIN REALLY DECENTRALIZED?



PURE PROOF OF STAKE IN A NUTSHELL

- the solution? maybe Pure Proof of Stake, proposed by Algorand. How it works?
- two phases
  - commettee selection
  - then, the commette reaches a consensus
- commette selection:
  - there are a set of tokens in the system, owned by the nodes
  - each token
    - has a owner, belongs to a public key
    - is equal to each other one
  - the majority of the tokens is in the hands of honest nodes
    - if I own lots of tokens, why should I attack the system?
    - why kicking myself?

PURE PROOF OF STAKE IN A NUTSHELL

phase 1: commette election

- how to select the members of the commetee?
  - choose X tokens at random
    - X configurable system-wide parameter
  - each of these tokens belongs to some node
  - the node partecipates to the consensus, if one of its token has been chosen
  - the more tokens the node has, the more chances it has to partecipate to the consensus
  - the blockchain grow up thanks to the elected commette
- but, who choose the tokens?
  - cannot be a centralized node, otherwise we are not solving the trilemma

PURE PROOF OF STAKE IN A NUTSHELL

- the election of the commette is democratic
  - commette is elected by the nodes themselves
- how the election is implemented?
  - each node owns a “cryptographic slot machine” for each one of its token
    - pull the lever and see if you lose or win
    - strong cryptography: the node cannot hack the machine
  - for each token
    - the lever can be lowered only once
  - for each winning token
    - obtain a ticket, with a cryptographic proof, guaranteeing the correctness of the winning
    - the more tickets I have, the more likely is that I can partecipate to the commettee

PURE PROOF OF STAKE IN A NUTSHELL

- consensus is organized in rounds
  - a different lottery for each round
  - each token has its own lottery
- distributed lottery
  - high scalability pulling the lever needs a few microseconds
  - high level of decentralization
- why proof of stake?
  - your winning probability depends on the stake, i.e. the number of tokens you hold

PURE PROOF OF STAKE: SECURITY ANALYSIS

- how can an hacker attack the system?
- if you know in advance who is going to be partecipating to the commette you can target them
  - corrupt them
  - steal their keys
  - denial of service attacks
- we need a kind of “last minute” selection of partecipants
- implemented through a technique called cryptographic sortition

PURE PROOF OF STAKE: SECURITY ANALYSIS

how an hacker can attack the system?

- it should corrupt the members of the commettee
- possible only if the hacker knows the identity of the winners which will form the commette
- but their identity is known only to the winners themselves
  - the winners propagate their tickets as soon they have finished the lottery
  - the attacker can only know the identity of the winners from the information logged from the network, but, in the meanwhile this information is epidemically spreading
- initially, the attacker does not know who it has to corrupt
- when the attacker has information about the winners, it is too late for it

PROOF OF STAKE: SOLVING THE TRILEMMA?

- decentralized
  - all tokens are equal
- scalable
  - needs a few microsecond to pull the lever, even with a large number of tokens
- secure
  - an higher number of partecipants make it more difficult to corrupt the system
    - decentralization means security
- obtains
  - disentermediation without costs and frauds

PROOF OF STAKE: SOLVING THE TRILEMMA?

this may bring to centralization? the rich gets richer as in the mining pools?

- I own only just my investments in the system
  - there are only limited forms of reward
- the right economic ecosystem
- main ideas
  - you enter the system, you have the right to mantain a good level of security
  - you invest in the system, you are interested in guaranteeing that the system works correctly

ALGORAND'S BASIC CONCEPTS

- a public and permissionless blockchain

- all users can partecipate in the consensus, can be validators
  - minimal computational resources required to run a node
  - a really democratic system

- the Algorand blockchain
  - a log of signed transactions transferring money to a public keys
  - like in Bitcoin, block contains a set of transactions and a cryptographic hash to the previous block
  - the structure of the blockchain is similar, the consensus is new

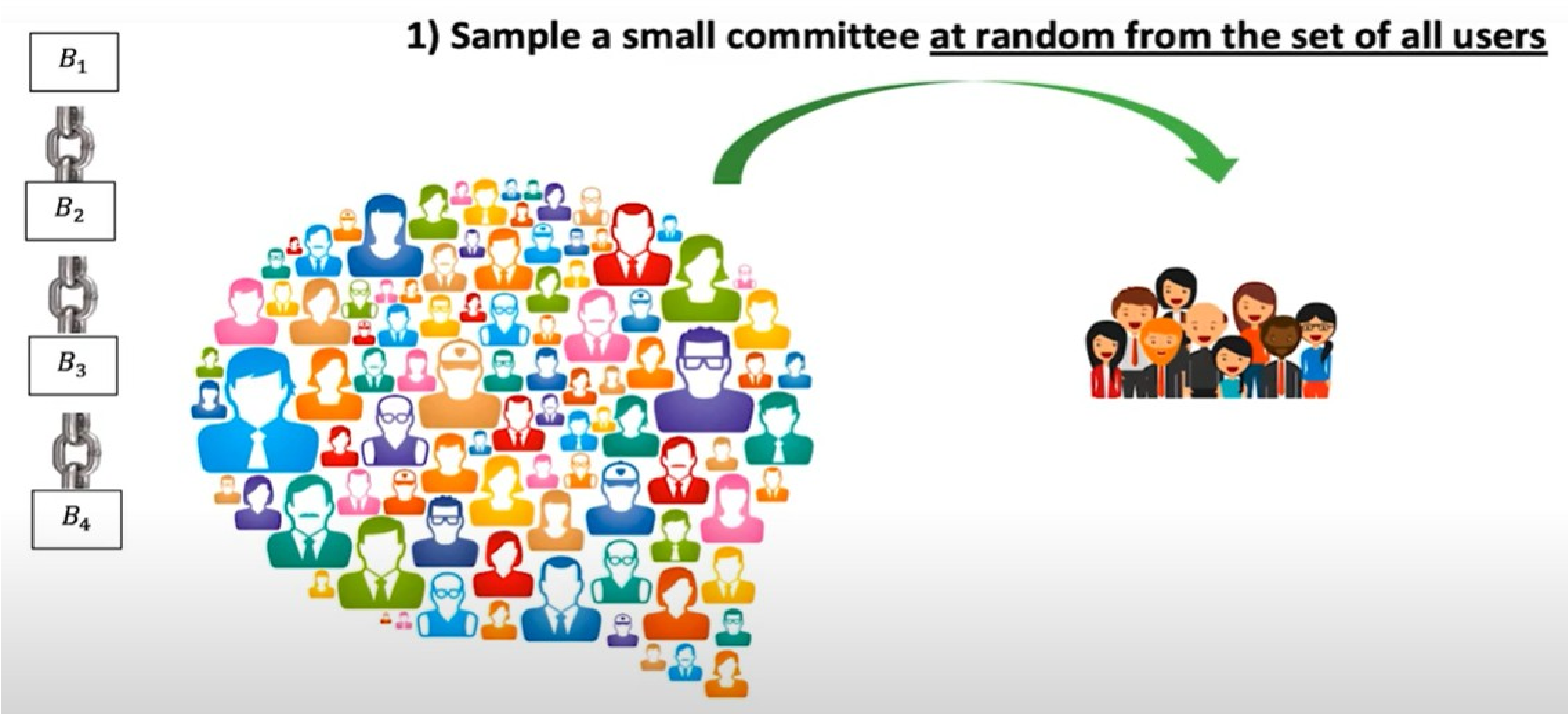
ALGORAND'S CONSENSUS: KEY IDEA

- uses a variant of Practical Byzantine Fault Tolerance

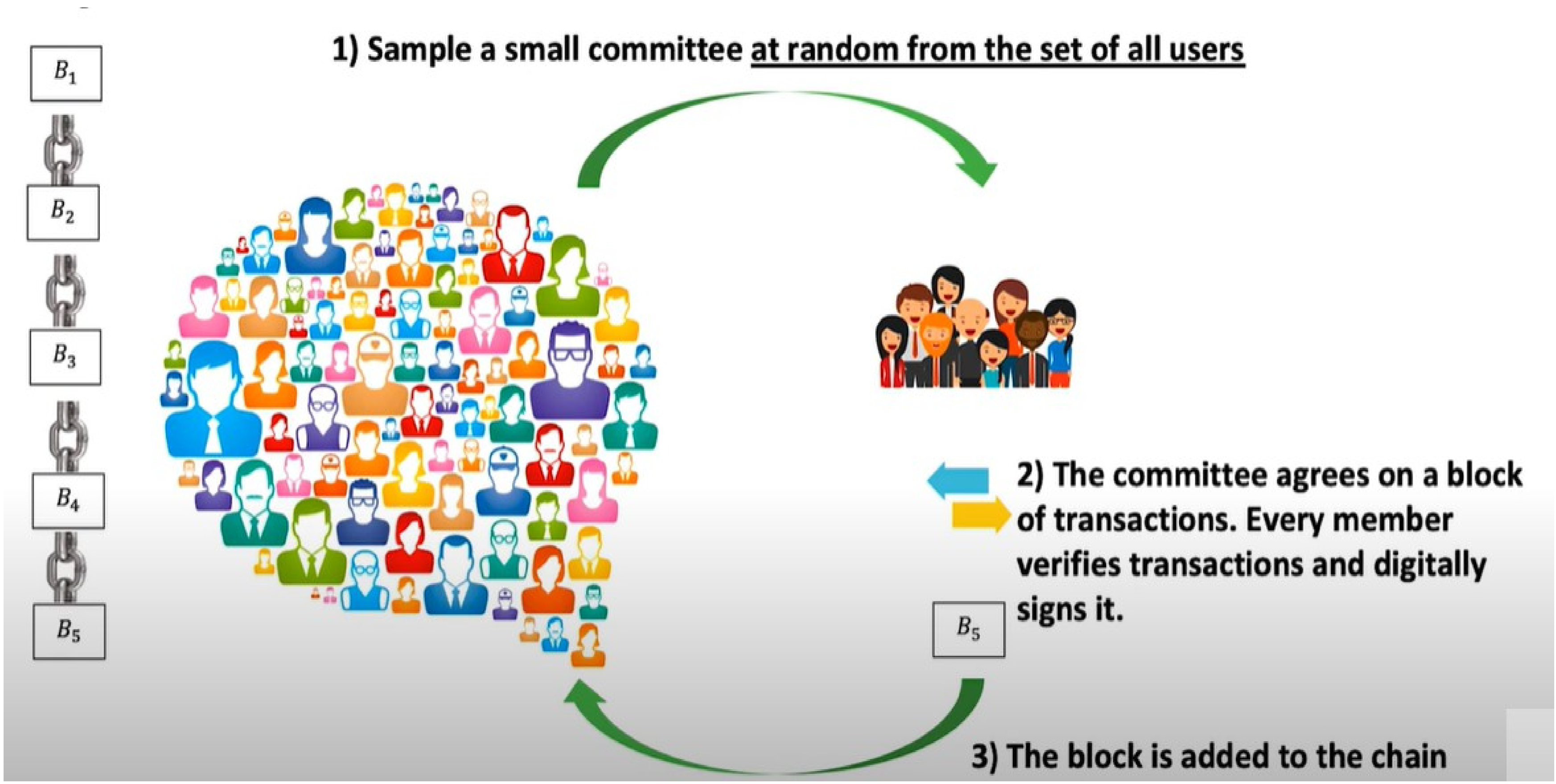
- PBFT
  - does not scale well
    - may scale till few thousands of nodes
    - do not scale in a pemissionless setting
  - nodes must be fixed and known in advance

- to make PBFT work, you need
  - to limit the number of partecipants
  - to randomly choose the partecipants from the nework, at each round
  - different partecipants (commettes) at each step (rounds) of the algorithm

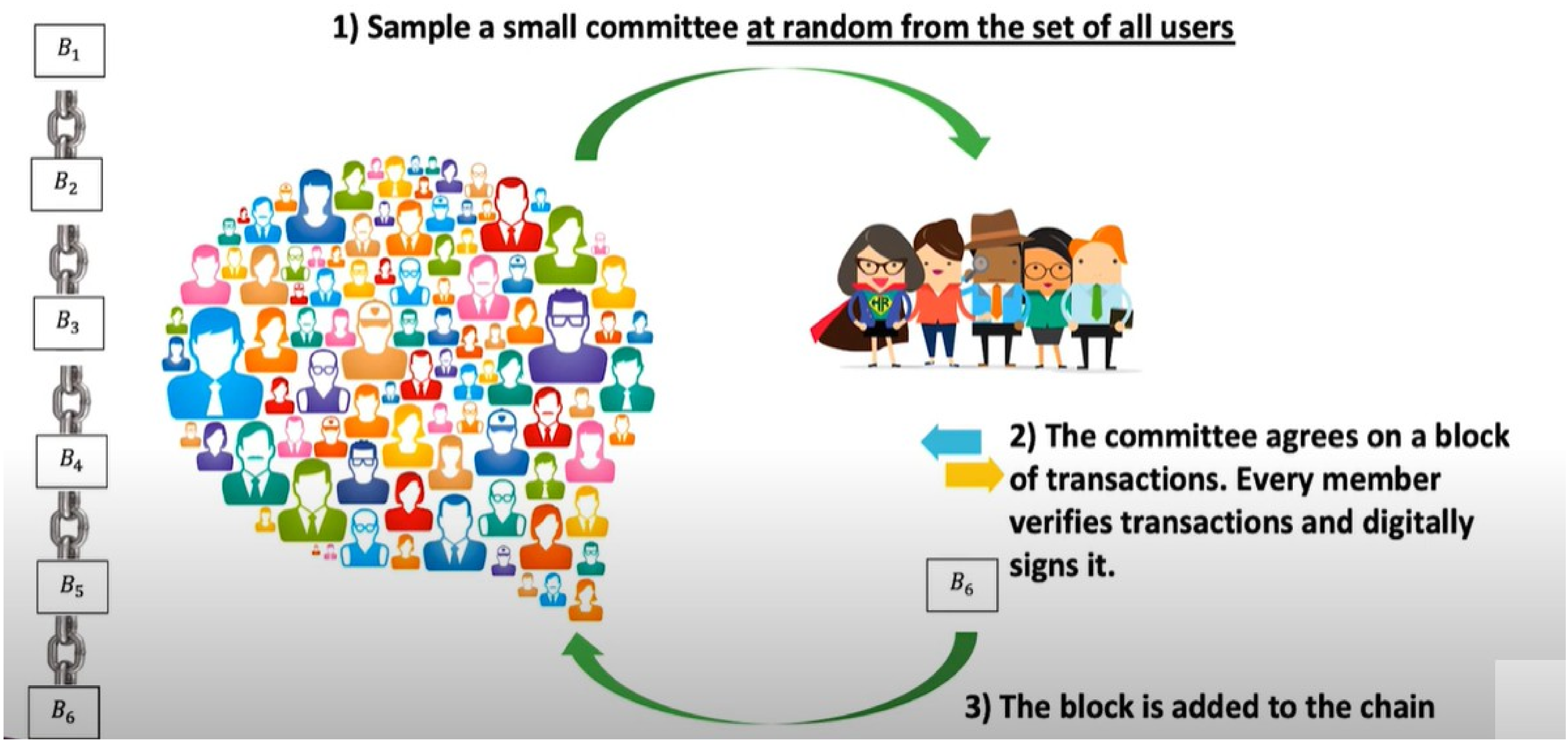
ALGORAND'S CONSENSUS: KEY IDEA



ALGORAND'S CONSENSUS: KEY IDEA



ALGORAND'S CONSENSUS: KEY IDEA



a new commette at each round of the protocol

ALGORAND'S CONSENSUS: KEY IDEAS

- any transaction is sent to all the other nodes of the network through a gossip protocol
- all the transactions propagate into the global peer-to-peer network
  - everyone sees each transaction appearing on the blockchain
- a node is chosen at random and assembles a block of transactions, propagates it to other nodes which vote for that block
  - both the proposer and the verifiers are from the elected commette
- then the block is added to the blockchain
- main assumption: honesty majority of money
- two basic technologies
  - random sampling using Verifiable Random Functions (VRF)
  - a variant of PBFT (seen in the previous lesson)
    - Byzantine Agreement* (BA*)

ALGORAND'S CONSENSUS KEY IDEAS

- starts from Byzantine Agreement in a permissionless setting
  - Sybil Attack what if a single user control a huge number of nodes?

- solution: weighted users
  - users are weighted by the money in their accounts
  - the attacker cannot amplify its power by simply using pseudonyms, needs stake
  - probability of forks is negligible
- honest majority of money
  - the attacker must control less than 1/3 of the monetary value of the system

VERIFIABLE RANDOM FUNCTIONS (VRF)

- developed by *Micali Rabin and Vadhan, 1999*

- VRF: cryptographic pseudo-random functions that process inputs and produce
  - a pseudorandom output
  - a verifiable proof of the correctness of its output
  - what does this mean?
    - the node which runs the function on its secret key can then later prove to some other node that the result of the function has been really generated by the running the function on the secret key, without revealing it.
    - the result is not a number generated "ad hoc" to cheat
- used to implement a lottery to choose the commette members
  - the leaders to propose a block
  - the commette members to vote on a block

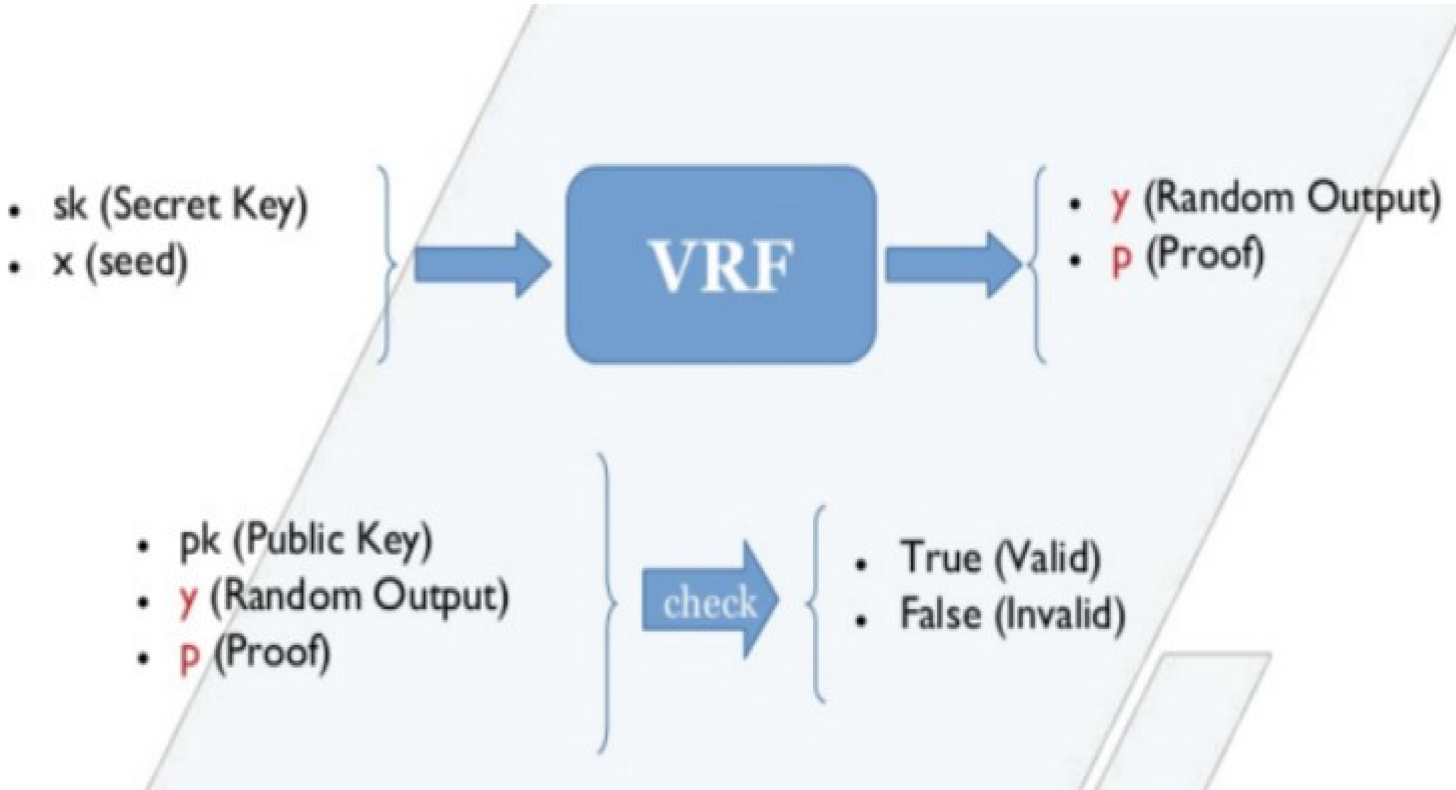
THE DISTRIBUTED ELECTION

- each node has a pair
  - secret key SK
  - verification key VK, publicly available
- a node wants to decide autonomously if it should be in the commette to run Byzantine Agreement*
  - before a round starts, every node autonomously calculates a VRF starting from its own secret key and from a seed
    - $VRF(S_k, seed) \rightarrow (Y, \pi)$ where the *seed* is a "*magic string*" which is available at every node in the system
  - checks if Y falls in a certain range [0,P] that depends on the stake the user holds in the system.

A DEEPER LOOK AT VRF

- a VRF is a triple of algorithms *Keygen*, *Evaluate*, and *Verify*.
- Keygen(r) → (VK, SK)
  - on a random input, the key generation algorithm produces a verification key VK and a secret key SK.
- Evaluate(SK, X) → (Y, ρ)
  - takes as input the secret key SK, a value X and produces a pseudorandom output string Y and a proof ρ.
- Verify(VK, X, Y, ρ) → 0/1
  - takes as input the verification key VK, the message X, the output Y and the proof ρ.
  - outputs 1 if and only if it verifies that Y is the output produced by the evaluation algorithm on inputs SK and X.
  - the proof ρ enables anyone who knows VK to check that Y corresponds to X, withouth knowing SK

VERIFIABLE RANDOM FUNCTIONS: RECAP



VERIFIABLE RANDOM FUNCTIONS

- the lottery
  - use the random number obtained by the VRF to sample from a binomial distribution
  - draw a number for every algo in a user's account: a lottery for each algo
  - the more algos in an account, the greater chance the account will be selected

- the user holds a proof (Y, π)
  - validates its commette membership for the block
  - given (Y, π) and the user's verification key Vk, anyone can verify that
    - Y is a valid output
    - has been really produced by the node claiming it
    - falls within the required range
    - so the node holding Vk can indeed serve in the commette

PROPERTIES OF VRF

- Time Complexity
  - execution time constant and independent of the length of input value
  - computation is cheap
- Uniqueness
  - impossibility to create two unique proofs that would verify the same set VK, Y, X
- Collision Resistance
  - impossible to find two value X1 and X2 that produce the same output Y
- Random uniqueness
  - impossibility to predict the output of the function
- similar to signature schemes, but
  - some of the signature bits may be predictable, VRF satisfy a stronger pseudo-randomness property
  - in a signature scheme, many valid signatures may exist for the same input

VRF ARE AVAILABLE?

- effective implementation of Verifiable Random Functions (VRFs) exist
- a VRF implementation exists in Ethereum and can be used in Solidity

```
pragma solidity ^0.5.0;

import "vrf-solidity/contracts/VRF.sol";


contract MyContract is VRF {

  function functionUsingVRF(
    uint256[2] memory _publicKey,
    uint256[4] memory _proof,
    bytes memory _message)
  public returns (bool)
  {
    bool isValid = verify(_publicKey, _proof, _message);
    // Do something...
    return isValid;
  }
}
```

CRYPTOGRAPHIC SORTITION

- an algorithm to choose a random subset of nodes according to their weigths
- given the weights w_i of each node i and the total weight of all users in the system W, the probability that a user is selected must be proportional to w_i/W
- implemented through VRFs
  - provides a source of verifiable randomness
- the sortion algorithm requires
  - a role parameter distinguishing the different roles that the user may be selected for (proposer or commette member)
  - a threashold τ determining the expected number of users elected in that role

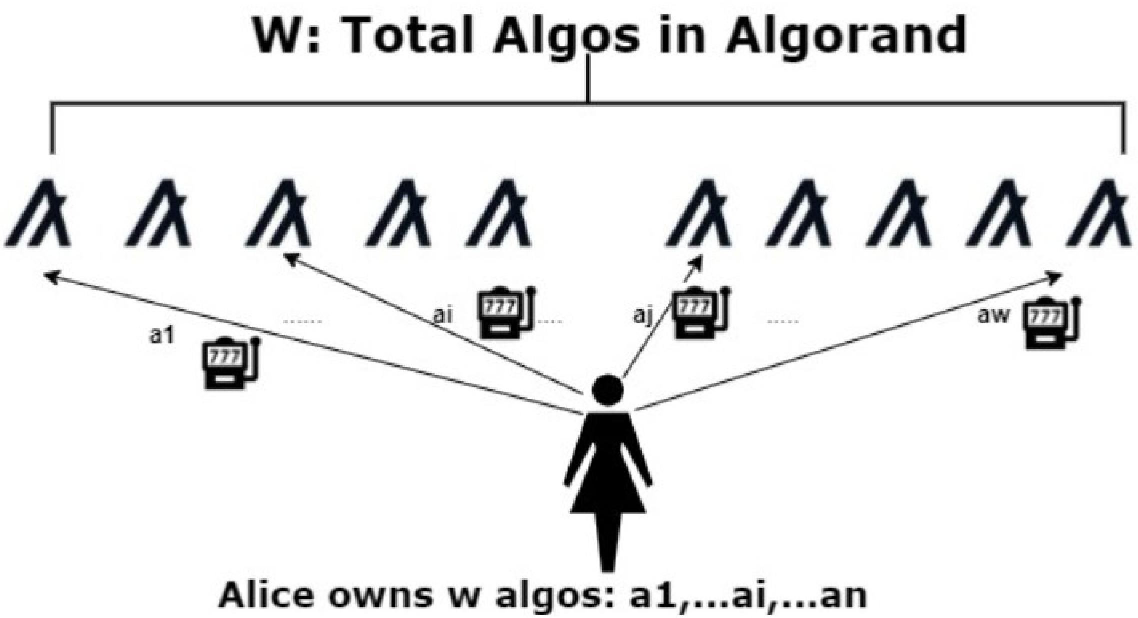
CRYPTOGRAPHIC SORTITION

procedure Sortition($sk, seed, \tau, role, w, W$):

$$\langle hash, \pi \rangle \leftarrow \mathrm{VRF}_{sk}(seed || role)$$
$$p \leftarrow \frac{\tau}{W}$$
$$j \leftarrow 0$$
$$\textbf{while } \frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right) \textbf{ do}$$
$$\quad j{+}{+}$$
$$\textbf{return } \langle hash, \pi, j \rangle$$

- every node autonomously runs the code that will decide if it will be elected
- cheating is infeasible, because of the crypthographic mechanisms
- election
  - does not depend on trusted third parties to elect the commette
  - can be parallelized and does not require interaction, so it is quite fast

CRYPTOGRAPHIC SORTITION



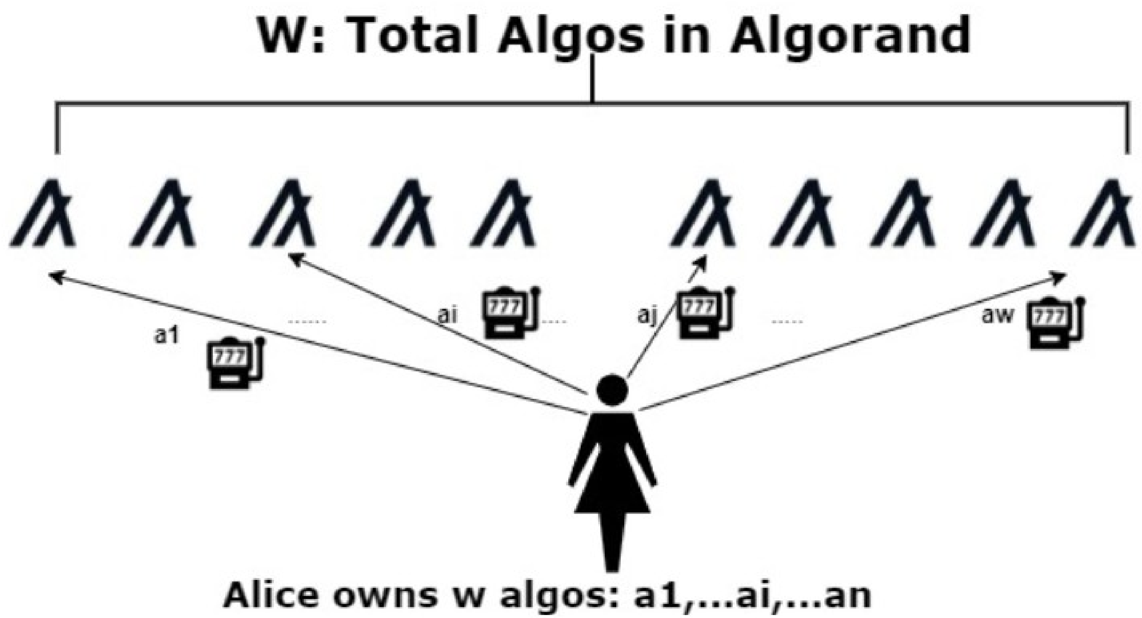
the main ideas

- place all the W coins existing in Algorand on a table
- there is a slot machine for each coin: pull the lever
- the result of each game is independent from the other ones
- winning probability: $p=\tau/W$

  τ is a parameter of the system

  note: W is very large, p is very small

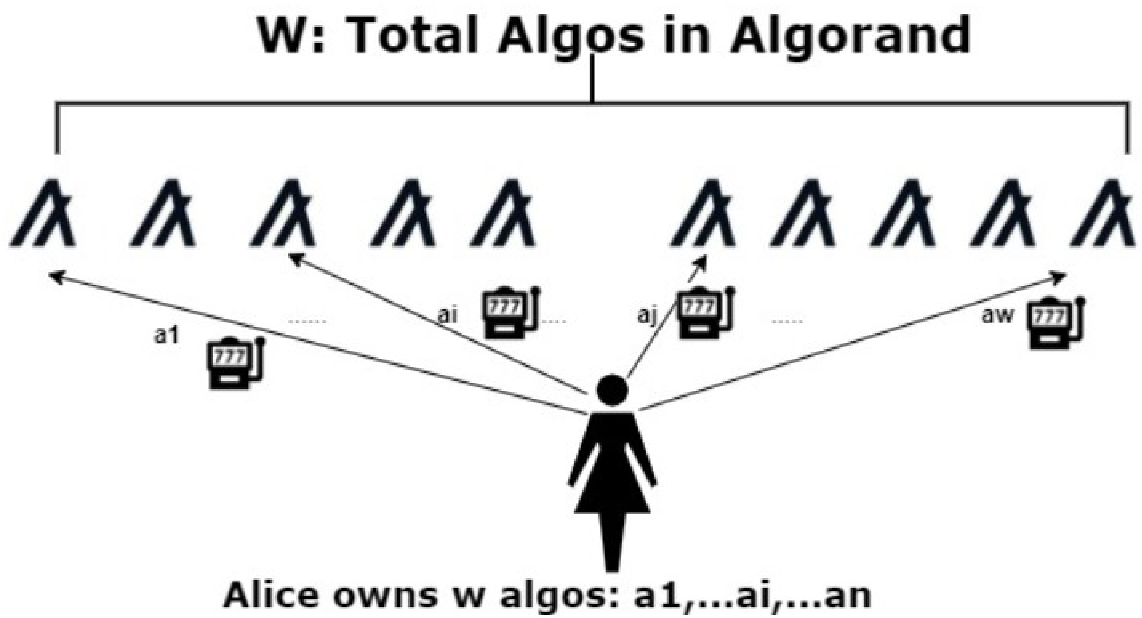
CRYPTOGRAPHIC SORTITION



the main ideas

- like flipping a biased coin
- label each coin as success or unsuccess
- expected number of successes is $W * p = W * \tau / W = \tau$
- τ can be used to control the expected amount of coins that are labelled success
  - control the commettee size

CRYPTOGRAPHIC SORTITION



the main ideas

- coins have an owner
- what does it happen if you consider ownership?
  - nothing changes, because all coins are equal
  - however, we have to consider how many coins each user owns

CRYPTOGRAPHIC SORTITION

- suppose the user owns w_i coins
- pull the lever for each of the coins
- each experiment is indipendent from the others
  - Bernoulli trials: Binomial distribution
- X random variable representing how many coins are labeled as success
- X ≈ Binomial (w_i,p)
- P(X=n) = B(n, w_i, p)

$$P(x) = \frac{n!}{(n-x)!x!} \cdot p^x \cdot q^{n-x}$$

$$\text{for } x = 0,\ 1,\ 2, \ldots, n$$

CRYPTOGRAPHIC SORTITION

- consider a user owning 2 coins
- the probability p of marking the coin as successful is very small, so it is very likely that neither of the two coins is a winning one
- the probability of having one winning coin is smaller
- two winning coins only if you are very lucky
- draw the probabilities on a line, of lenghth 1 (sum of the probabilities must be 1)

$B(k,w_i,p)$

CRYPTOGRAPHIC SORTITION

$B(k, w_i, p)$

- the length of the segments correspond to the probabilities
- throw a dart in the segment [0,1]
- the probability that the dart hits the segment k corresponds to the length of the segment
- dart is like the random number obtained by the VRF
- equivalent from sampling from distribution using the cumulative distribution function

CODING CRYPTOGRAPHIC SORTITION

procedure Sortition($sk, seed, \tau, role, w, W$):

$\langle hash, \pi \rangle \leftarrow \text{VRF}_{sk}(seed || role)$

$p \leftarrow \frac{\tau}{W}$

$j \leftarrow 0$

while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right)$ **do**

$\quad j{+}{+}$

return $\langle hash, \pi, j \rangle$

- reduce the random hash obtained by the VRF to the interval [0,1] by dividing by 2^{hashlen}
- the while loop finds out in which segment the random hash lies
  - $\sum_{k=0,j} B(k, w_i, p)$ cumulative size of the first k segments
  - the returned value j is the number of succesfull tickets
    - the bigger is j, the more voting power

CRYPTOGRAPHIC SORTITION: SECURITY ANALYSIS

procedure Sortition($sk, seed, \tau, role, w, W$):

$\langle hash, \pi \rangle \leftarrow \mathrm{VRF}_{sk}(seed || role)$

$p \leftarrow \frac{\tau}{W}$

$j \leftarrow 0$

while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right)$ **do**

$\quad j{+}{+}$

return $\langle hash, \pi, j \rangle$

- the bigger is j, the more voting power
- a malicious user can try to fake the value returned by the VRF
  - can manipulate the input of the hashing function, but...
  - the seed is a system wide value, difficult to manipulate
  - SK: the consensus protocol forces to choose the secret number before the seed was agreed

CRYPTOGRAPHIC SORTITION: SECURITY ANALYSIS

procedure Sortition($sk, seed, \tau, role, w, W$):

$\langle hash, \pi \rangle \leftarrow \mathrm{VRF}_{sk}(seed || role)$

$p \leftarrow \frac{\tau}{W}$

$j \leftarrow 0$

while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right)$ **do**

  j++

return $\langle hash, \pi, j \rangle$

- Sybil attacks: do you have advantages from having multiple accounts?
  - no, because each experiment is independent from the others
  - more formally
    - X ≈ *Binomial (k1,p)*
    - Y ≈ *Binomial (k2,p)*
    - X+Y ≈ *Binomial (k1+k2,p)*

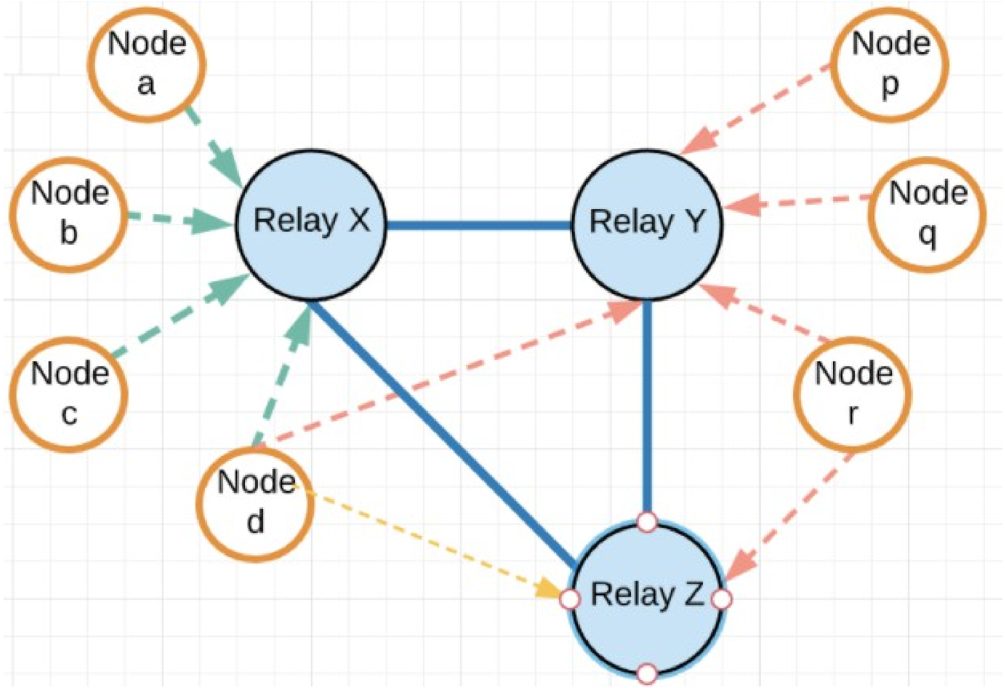
CHOOSING THE SEED AND THE SECRET KEY

- seed must be chosen at random and publicly known by every node
- the choice of the seed must not be controlled by an adversary
- at each round of the protocol, a new seed, is computed at each node
- as VRF of the previous seed and the round
- if the block proposed by the node is confirmed, the seed is stored in it
  - every node find the seed on the blockchain
- requires that user's secret key is computed well in advance with respect to the seed
- details in the paper
- "*Algorand: Scaling Byzantine Agreements for Cryptocurrencies*", Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, Nickolai Zeldovich, ACM SOSP '17.

THE BYZANTINE AGREEMENT PHASE

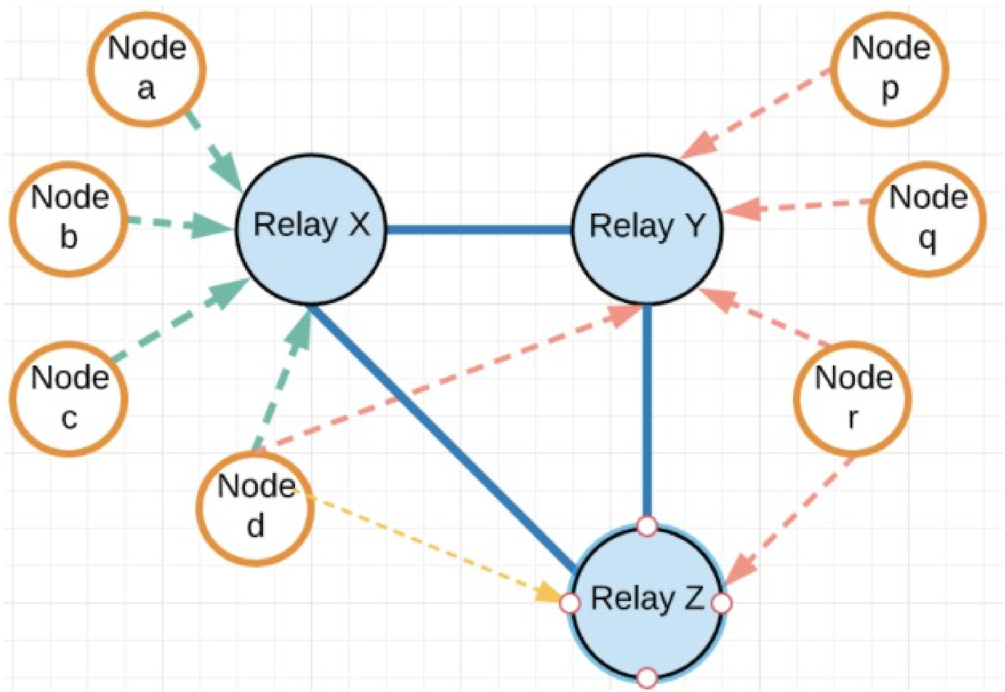
- Goals
  - Safety: all users agree on the same value
  - Liveness: the system makes progress
- Assumptions
  - weak synchrony assumption
    - the netwok can be asynchronous for a long but bounded period of time
    - after an asynchrony period, the network must be strongly synchronous for a reasonable period to ensure safety
    - strong synchrony assumptions: most of the honest nodes can send messages to other nodes within a known time bound
- organized in two phases: Reducion and BinaryBA()

ALGORAND P2P NETWORK



- a hierachical P2P network composed of
  - nodes: partecipate to the consensus and communicate with other nodes through Relays
  - relays: provide communication hubs for the nodes

ALGORAND P2P NETWORK



- nodes and relays form a star topology
- connections node-relays and relays-relays are continously updated
- relays
  - characterized by high bandwith
  - receive rewards for their services

ALGORAND: TOKEN SUPPLY

- the entire supply of algos minted at the genesis of the blockchain:10 billions of Algos.
  - initial auction June 2019
  - then by reward vesting, for Relay Nodes, grant funding, etc.
- this is the fixed and immutable supply of Algos.
- not all initial supply is liquid: preminted locked are gradually unlocked and distributed
- at November 2020 16% of the total Algo supply injected in circulation
- Algorand tokenomics
  - a detailed plan for the long term allocation of the rewards
  - see https://algorand.foundation/the-algo/algo-dynamics

ALGORAND: SMART CONTRACTS

- mainly a cryptocurrency
- smart contracts two-tier architecture
  - layer-1 smart contracts
  - layer-2 smart contracts
- written in TEAL, an assembly-like language
- layer-1 smart contracts, on-chain smart contracts
  - execute on-chain many common operations
    - atomic swaps
    - atomic transfers
    - multisignature wallets
  - for the moment, state-less programs, a stateful version is under development
- layer-2 off chain smart contracts