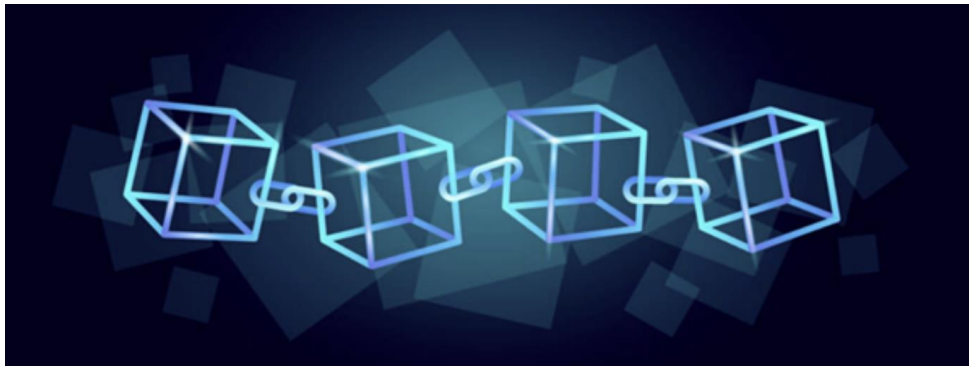
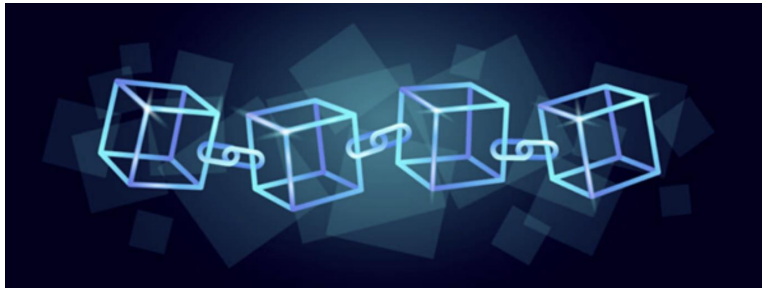


PoW vs PoS

Tal Rabin

University of Pennsylvania

Algorand Foundation



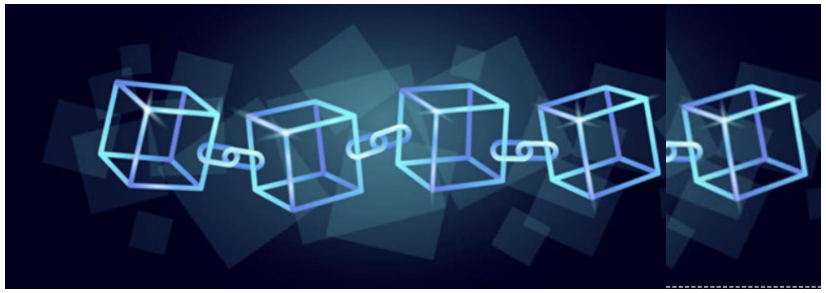
Leader elected and proves it is the leader (secret until block is proposed)

Today

Various techniques
Can mix and match

Proposes a blo

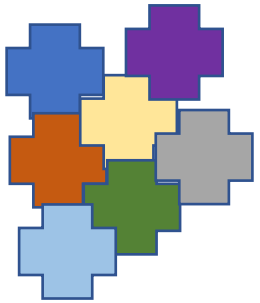
Block is agreed upon and added to the blockchain



Leader Election Proof of Work -- Nakamoto

Puzzle

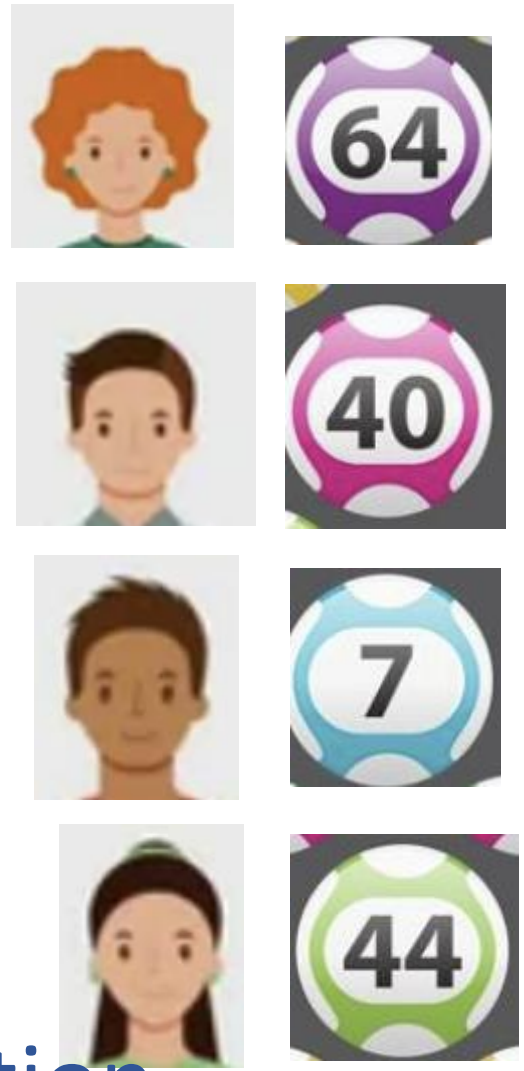
Proof



Proof of work – you work until you solve the puzzle

Proof-of-Stake

Put your hand in the hat according to how many tokens you have -- your stake



Mathematical computation – exponentiation

Proof of Work – Leader Election

Goal: computational problem that

- takes time $\Omega(D)$ to solve, but (D is called the **difficulty**)
- solution takes time O(1) to verify

How? $H: X \times Y \to \{0,1,2,\ldots,2^n - 1\}$ e.g. $n = 256$

- puzzle: input $x \in X$, output $y \in Y$ s.t. $H(x,y) < 2^n/D$
- verify(x,y): accept if $H(x,y) < 2^n/D$

Time for Choosing Leader

- Bitcoin 

- Time of computation continuously maintained:
  - Time is getting longer, need to make easier -- reducing D
  - Time is getting shorter, need to make harder -- increasing D

Verifiable Random Function (VRF)

- Signature Scheme
  - Gen() → (SK,PK)
  - Sign, S(SK, m) → σ
  - Verify, V(PK, m, σ) → accept or reject
- Properties:
  - V(PK, m, S(SK, m)) = accept
  - If (SK,PK) is fixed then given a value r, the value S(SK,r)= σ is random
  - If SK is not known, then the value σ is secret
  - Once σ is announced easy to verify that it is the correct value: V(PK, m, σ) = ?

How to Use VRF for Leader Election




SK_1 PK_1

SK_2 PK_2

SK_3 PK_3

SK_4 PK_4

PKI

PK_1

PK_2

PK_3

PK_4

r

$=\sigma_1$
$=S(SK_1,r)$

$V(PK_1,r, \sigma_1)=acc$

$=\sigma_2$

$V(PK_1,r, \sigma_2)=rej$

$=\sigma_3$

$V(PK_1,r, \sigma_3)=acc$

$=\sigma_4$

$V(PK_1,r, \sigma_4)=acc$

Why is this a good leader election mechanism?

Time for Choosing Leader

- Bitcoin

- PoS -- Algorand

Comparison of PoW and PoS – Electricity

- PoW: Average years of household-equivalent electricity to mine one Bitcoin using the most efficient hardware available – Aug 2021*:

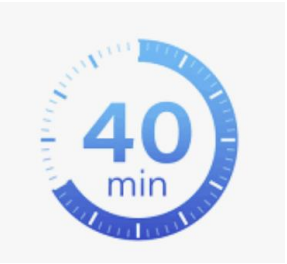
9 Years

- PoS: Negligeable

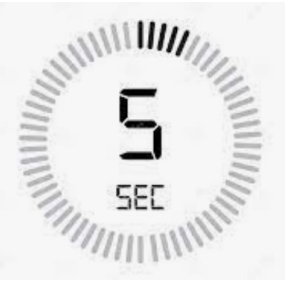
*NY Times: Jon Huang, Claire O'Neill and Hiroko Tabuchi, Sept 3 2021

Agreeing on the Block

- Wait until enough time has elapsed, say 40 minutes
  - danger of fork and double spending



- PoS: execute Byzantine Agreement protocol
  - Eliminates the danger of a fork



Why Can’t Bitcoin Run a Byzantine Agreement

- Byzantine Agreement is a protocol that requires multiple rounds of interaction between parties
- Want to maintain the You Only Speak Once (YOSO) idea
  - Can’t choose parties quickly enough

Player Replaceability

- When a party can be chosen in a split second via VRF then a full committee can be chosen quickly

Step 1 Step 2 Step 3 and so on…..



Randomized Byzantine Agreement

- Setting: n parties $P_1,\ldots,P_n$, t might be faulty
- [PSL] Deterministic BA requires t+1 rounds
- [PSL, FLM] Without digital signatures number of parties $n \geq 3t+1$
- Can we do better? Go beyond the lower bounds?

Randomized Byzantine Agreement

- Setting: n parties $P_1,\ldots,P_n$, t might be faulty
- [PSL] Deterministic BA requires t+1 rounds
- [PSL, FLM] With digital signatures
  - Dolev-Strong can tolerate any number of faulty parties, but still t+1 rounds
- Can we do better? Go beyond the lower bounds?

Randomized Byzantine Agreement

- Setting: n parties $P_1,\ldots,P_n$, t might be faulty

- [PSL] Deterministic BA requires t+1 rounds
  Non-Deterministic BA requires constant expected number of rounds
- [PSL, FLM] Without digital signatures number of parties $n \geq 3t+1$

- Can we do better? Go beyond the lower bounds?

Randomized Byzantine Agreement

- Setting: n parties $P_1,\ldots,P_n$, t might be faulty
- Global clock, parties are synchronized
- Assume a beacon that omits a random bit at each clock tick
- Variant of the problem: Byzantine General has input {0,1}
- Validity: If the general is honest all (honest) parties output the general's input
- Agreement: All honest parties output the same bit

Randomized Byzantine General

- Round 0: General sends input v to all parties. Party P_i sets $v_i = v$

- Beginning of Epoch (repeat until instructed to terminate)
  - Round 1: Party P_i send (init, v_i) to all parties (including itself)
  - Round 2: If # of (init, v_j) received is $\geq$ 2t+1 for a single v, send (echo, v)

  - Decision :
    - IF # of (echo, v_j) received is $\geq$ 2t+1 for a single v, then output v, set $v_i = v$ and run for one more epoch and then terminate (do not change your output)
    - ELSE IF # of (echo, v_j) received is $\geq$ t+1 for a single v, set $v_i = v$
    - ELSE set v_i = bit of the beacon
- End of Epoch

Claim 1:

- If all honest parties start an epoch with the same v_i then they will all terminate during the next epoch and output the value v_i

- Corollary: Validity holds as the honest general will send the same value to all honest parties

Claim 2:

- If an honest party P_i outputs v then all other honest parties will, in the next epoch, output v

- From the protocol:
  - IF # of (echo, v_j) received is $\geq$ 2t+1 for a single v, then output v, set v_i = v and run for one more epoch and then terminate (do not change your output)
  - ELSE IF # of (echo, v_j) received is $\geq$ t+1 for a single v, set v_i = v

Claim 3:

- If an honest party P_i sets $v_i = v$ in the ELSE IF then any other honest party P_j that sets v_j to some value will set it to v as well
- From the protocol:
  - ELSE IF # of (echo, v_j) received is $\geq$ t+1 for a single v, set $v_i = v$
  - And
  - Round 2: If # of (init, v_j) received is $\geq$ 2t+1 for a single v, send (echo, v)

Claim 4:

- If all honest parties are in the ELSE IF or all honest parties are in the IF then all honest parties will output the same value v in the next epoch and terminate in the following one

Claim 5:

- If the honest parties are in the ELSE IF and ELSE steps with probability half, they will all set v_i to the same bit

Advantages of PoS relative to PoW

- Green
- Increased throughput
- Lower latency
- If use BA – no forking
- More aligned incentives