HS18

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

Consensus & Scalability

R. Blum

roman.blum@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Agenda

- **Consensus**
  - Definition
  - Consensus Types
  - Blockchain Consensus (PBFT, PoW, PoS)
  - Consensus in the Bazo Blockchain
- **Scalability**
  - Definition
  - Motivation and Problems
  - Solutions (Plasma, State Channels, Sharding)
  - Introduction to Sharding
  - Scalability in the Bazo Blockchain

CONSENSUS

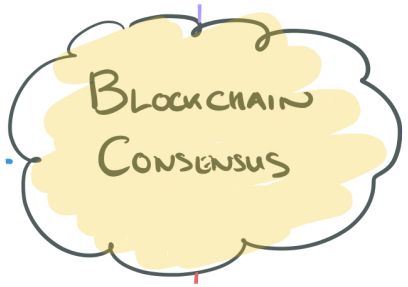
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

What is Consensus?

- **Definition: Consensus decision-making is a group decision-making process in which group members develop, and agree to support a decision in the best interest of the whole.**
- **Consensus may be defined professionally as an acceptable resolution, one that can be supported, even if not the "favourite" of each individual.**
- **Different ways to reach consensus:**
  - Person in-charge decides
  - Executive committee decides
  - Simple Majority (>50%)
  - Super Majority thresholds (90%, 80%, 75%, two-thirds, …)
  - ...

Blockchain Consensus

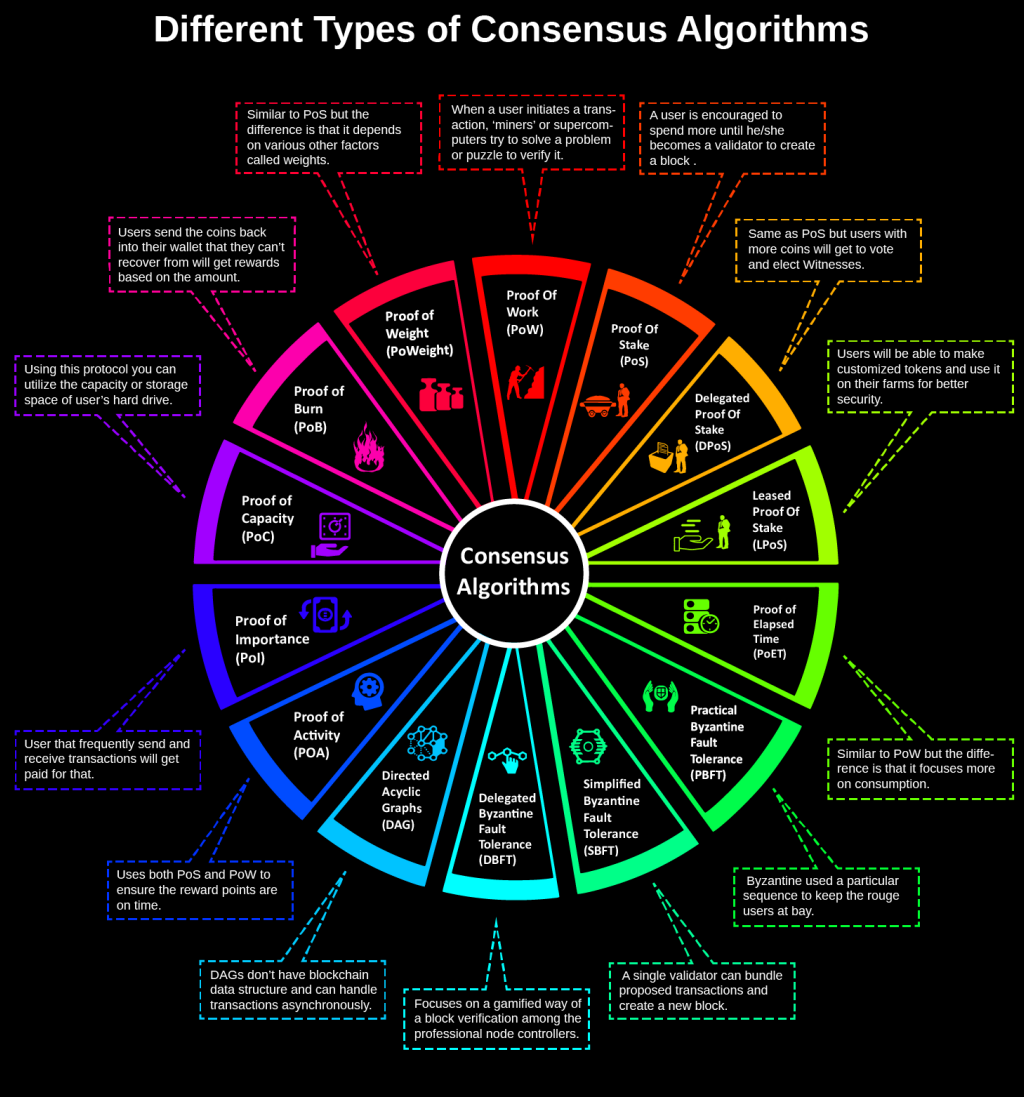


- **Classical Consensus Models**
  - Crash failure models → honest nodes failing
  - Byzantine failure model → able to tolerate a portion of malicious nodes
- **Elected Leader**
  - Probabilistic elected leader (e.g., who can find the hash first?)
  - Most known Proof-of-Work (PoW)
  - Also, Proof-of-Stake (value held on the chain and its variants), Proof-of-Capacity (PoC), Proof-of-Burn (PoB), Proof-of-Authority (PoA), etc.
- **Hybrid Consensus Models**
  - Single consensus has many limitations
  - Combine different consensus mechanisms
- **Scalability**
  - System is divided into shards (communities)
  - Cross-chain communications

Source

Blockchain Consensus

- **Recall: A blockchain is a public, decentralized ledger. Anyone can access it, and there is no single entity controlling it.**
- **Nodes must evaluate and agree on all addenda before they are permanently incorporated into the blockchain**
- **We cannot be sure of the other's trustworthiness, it is vital that all new information must be reviewed and confirmed before being accepted.**
- **Algorithms to discuss:**
  - Practical Byzantine Fault Tolerance (PBFT)
    e.g. Hyperledger, Stellar, Ripple
  - Proof of Work (PoW)
    e.g. Bitcoin, Ethereum
  - Proof of Stake (PoS)
    e.g. Peercoin, Ethereum's Casper, Bazo



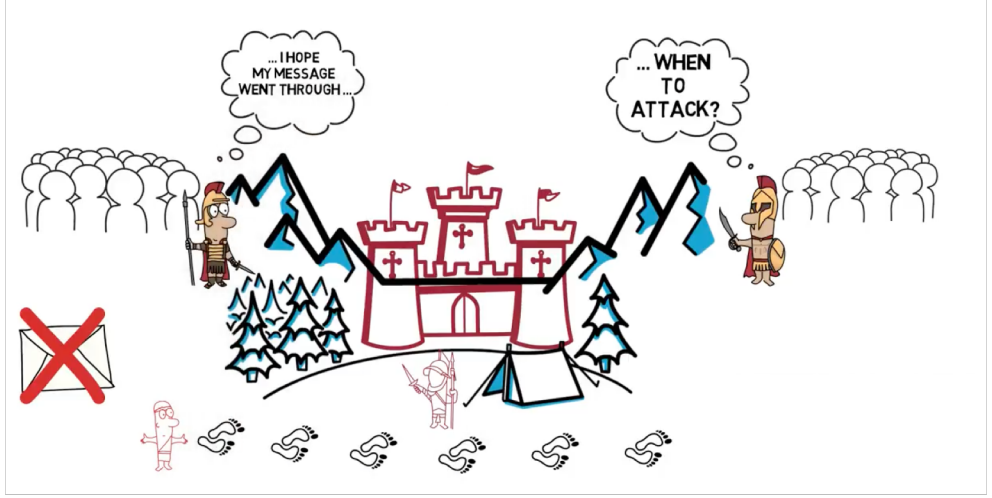
Source: 101blockchains.com

URL: b.socrative.com/student/

Room: HSR

The Two Generals Problem

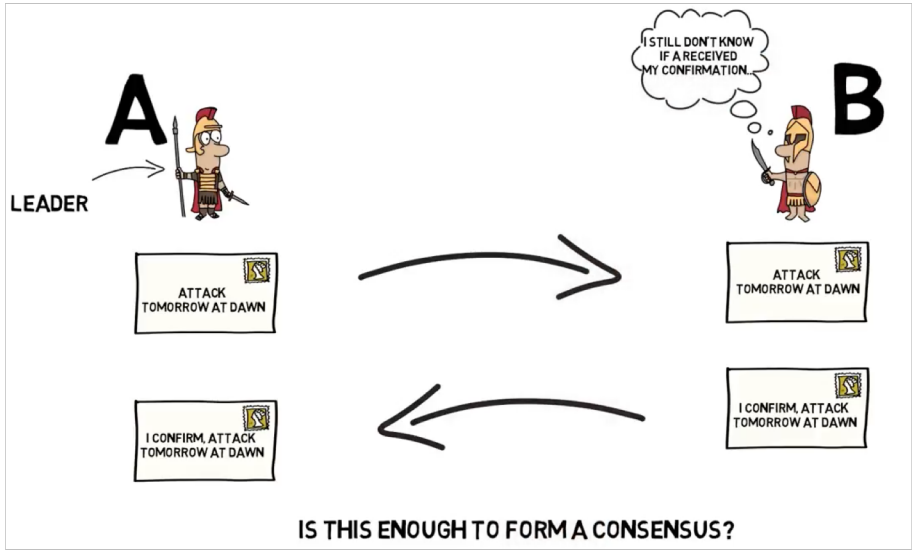
- **Two armies are preparing to attack a city from opposite sides.**
- **The General of army A is orchestrating the attack, and has decided that they must attack simultaneously at noon on March 3rd to succeed.**
- **He sends a sealed, encrypted letter by messenger across the valley to the General of army B, informing him of the plan.**
- **If the attack is not simultaneous, both attacking armies will be destroyed by the defenders.**
- **What could go wrong?**



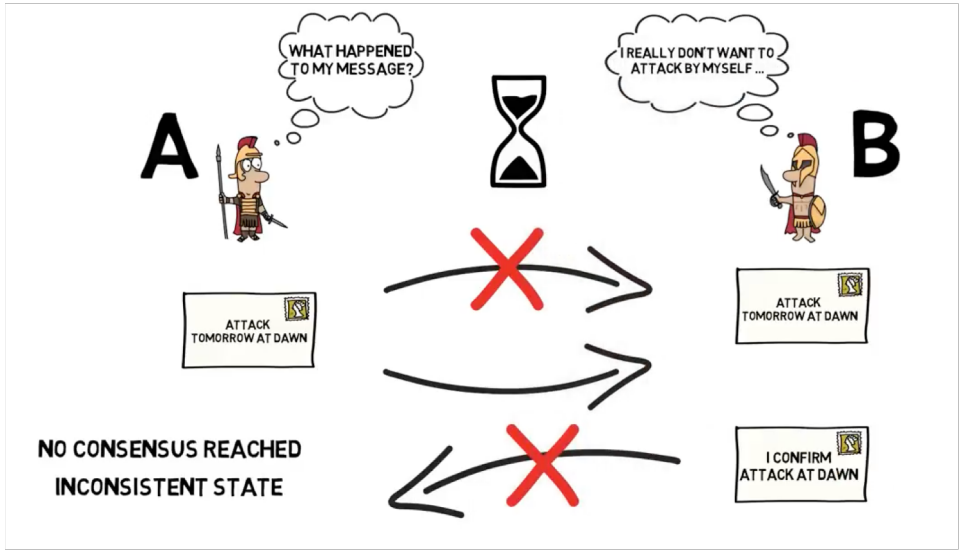
Source

The Two Generals Problem

Case 1: No guarantee of acknowledgment



Case 2: Faulty communication channel

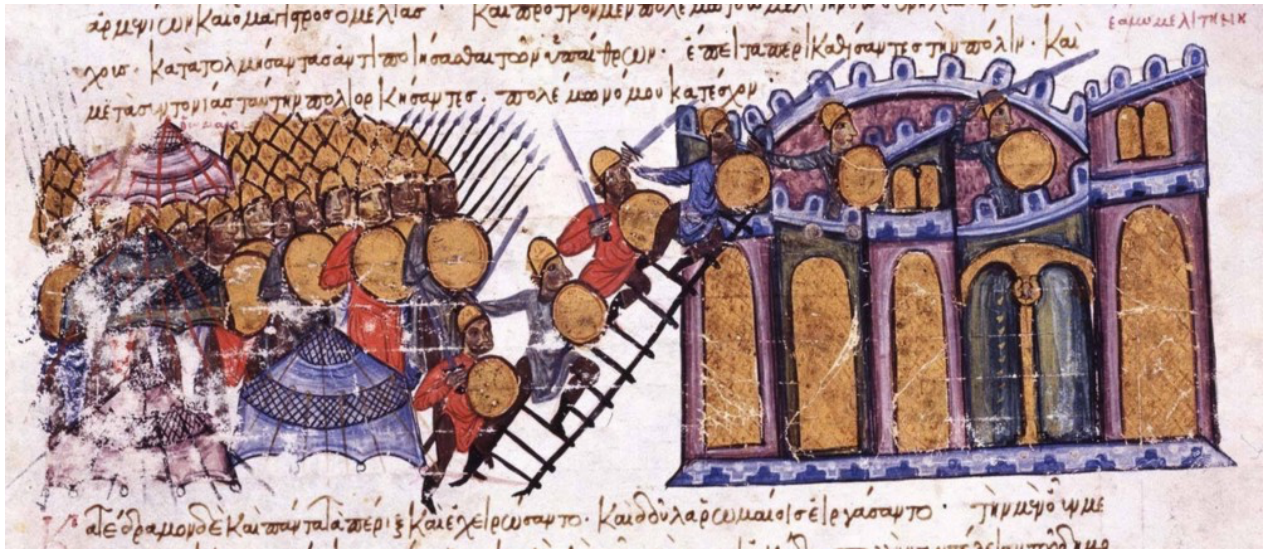


- **The Two Generals Problem has been proven to be unsolvable.**

Source

The Byzantine Generals Problem

- **First referenced in the paper "The Byzantine Generals Problem", published in 1982**
- **How do you make sure that multiple entities, which are separated by distance, are in absolute full agreement before an action is taken?**



The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Categories and Subject Descriptors: C.2.4. [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.5 [**Operating Systems**]: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

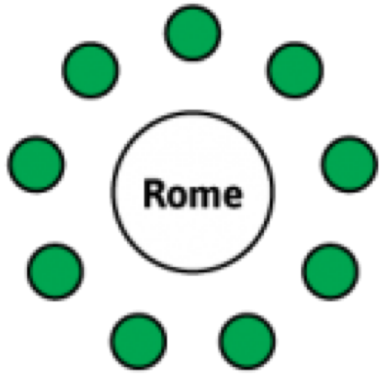
Additional Key Words and Phrases: Interactive consistency

1. INTRODUCTION

A reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked—namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. We devote the major part of the paper to a discussion of this abstract problem and conclude by indicating how our solutions can be used in implementing a reliable computer system.

We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals

The Byzantine Generals Problem



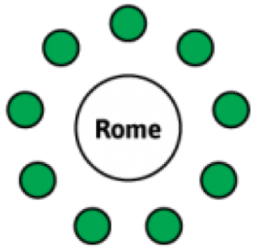
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.

In order to launch a succesful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.

The decision to either **attack** or retreat is put up to a **vote**. Whichever option receives **more than 50%** of the votes, that's what the Generals will do (**retreat** in the example above).

Source: CryptoGraphics.info

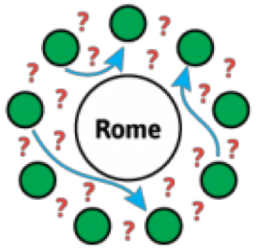
The Byzantine Generals Problem



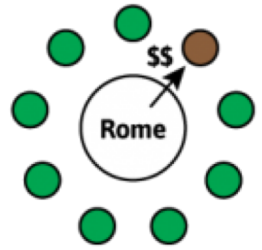
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.

In order to launch a succesful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.

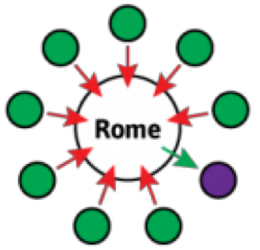
The decision to either **attack** or retreat is put up to a **vote**. Whichever option receives **more than 50%** of the votes, that's what the Generals will do (**retreat** in the example above).



Problem 1
The generals communicate by using **couriers**, who have to cross unknown areas controlled by the Romans, risking capture or their message becoming corrupt.



Problem 2
Each of the generals could be bribed by the Romans: **Traitorous Generals**.



Problem 3
Any of the Generals can make the wrong decision, regardless of the vote: **Improperly Functioning Generals**.

- "The proof-of-work chain is a solution to the Byzantine Generals' Problem." – Satoshi Nakamoto
- It's a probabilistic solution to the Byzantine Generals Problem, which means the confidence that a consensus is reached is growing with every block added to the chain, but it never reaches 100%.
- Practical Byzantine Fault Tolerance (PBFT): "The algorithm offers both liveness and safety provided at most $\left\lfloor \frac{n-1}{3} \right\rfloor$ out of a total of n replicas are simultaneously faulty.

Source: CryptoGraphics.info

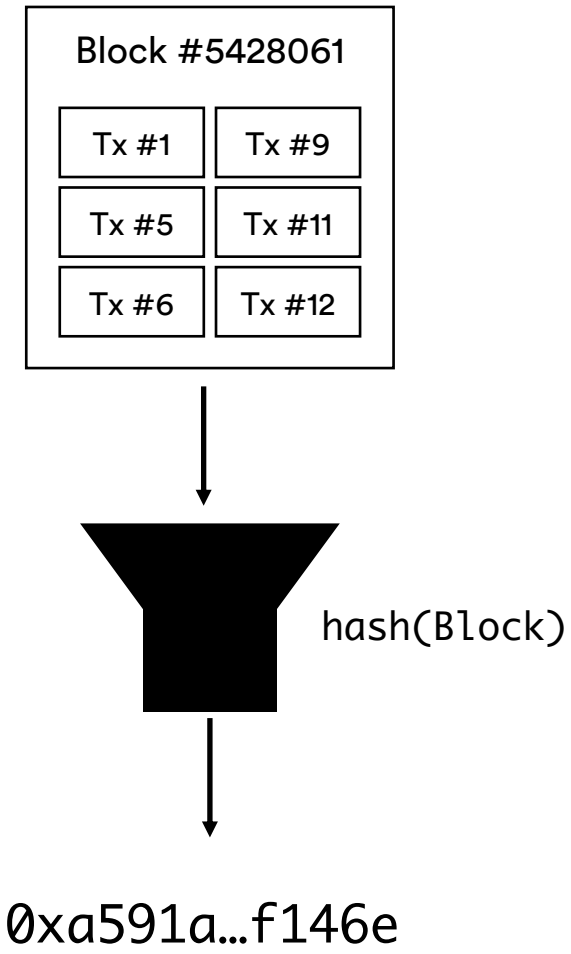
Blockchain and the Byzantine Generals Problem

1. **A General solves the PoW problem, creating a block that is broadcasted to the network**
2. **Following receipt of this block, each General verifies and works on solving the next PoW problem, incorporating the prior solution into it, so that their plan adds on to the previous resolution**
3. **Each time a General solves a PoW problem, a block is generated and the chain begins to grow. In time, any General working on a different solution will switch over to the longest chain. This is the one most Generals are contributing to and therefore has the greatest chance of success**
4. **As the Generals know roughly how long a PoW solution takes to solve, after a set amount of time they will know if enough of the other Generals are also working on the same chain**

Through this process, the Generals can arrive at a consensus of when to attack, can estimate their chances of successfully doing so, and can prevent multiple different signals to attack being sent simultaneously.

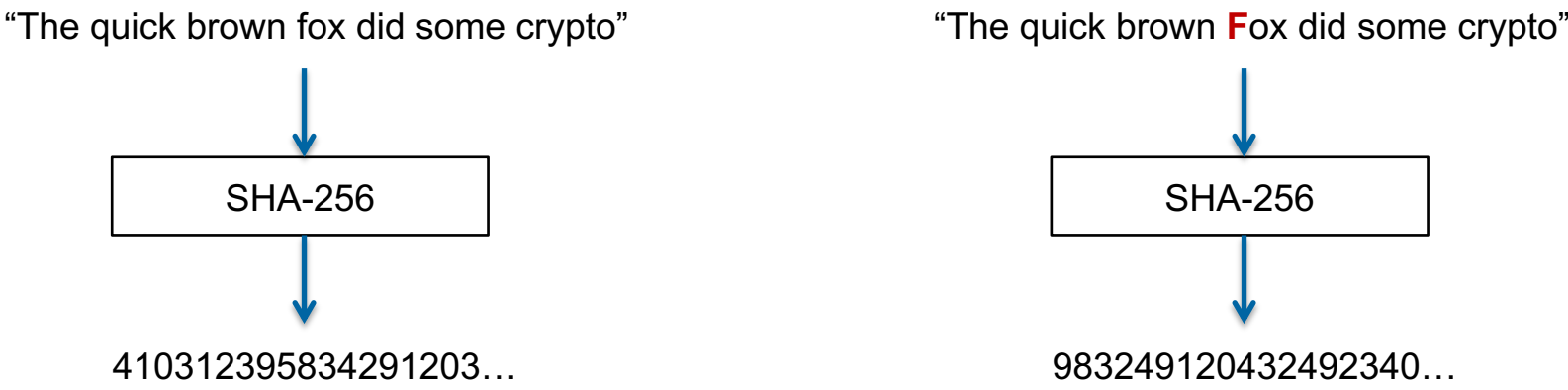
Proof of Work

- **Each node of the network has to solve a difficult mathematical problem to propose a block**
- **Since it takes significant computing power to solve this problem, it proves that the node has done sufficient work to produce the block**
- **Bitcoin: The digest of the hash function must start with a number of zeroes (simplified!)**



Recall: Hash Functions

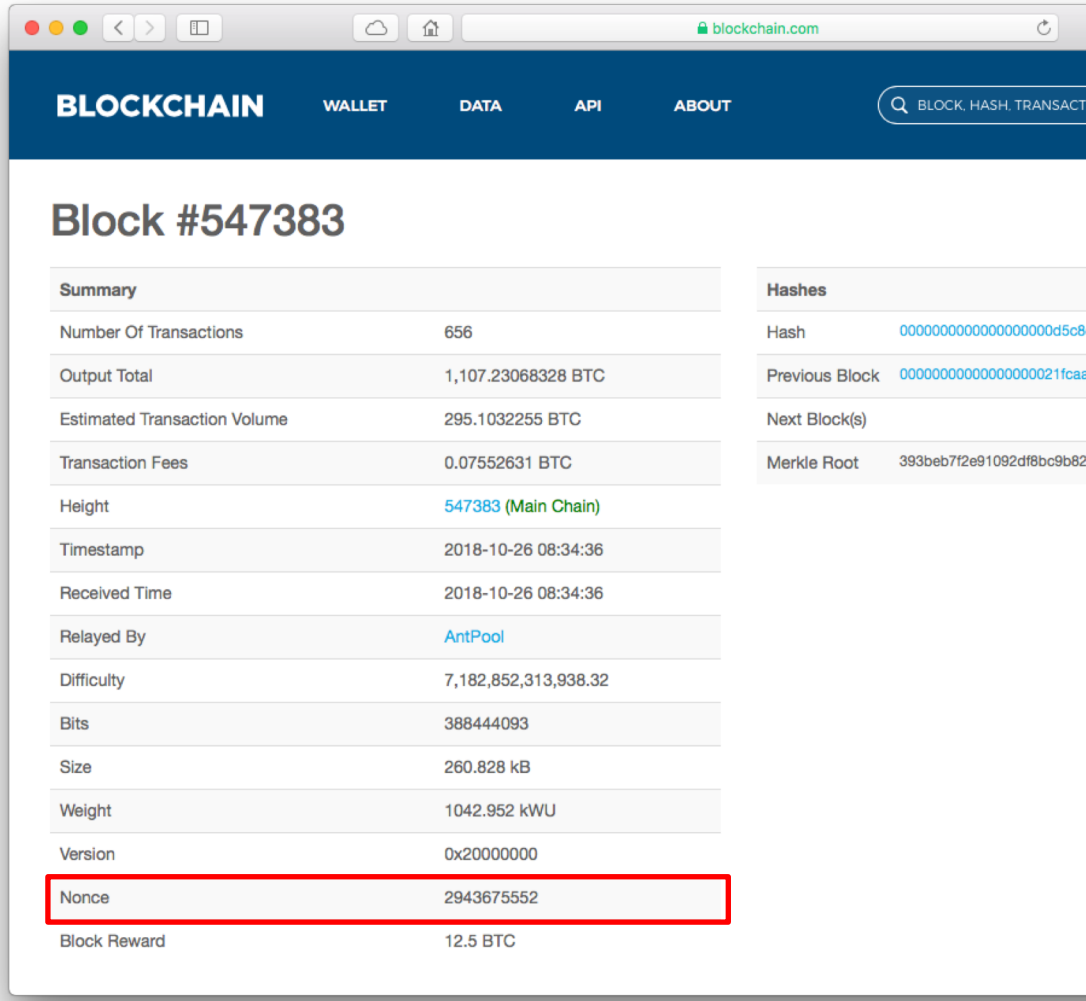
- **A hash function (like SHA-256) takes a block of data in and produces an effectively random fixed size integer**
- **For the same input, the hash function always yields the same output.**
- **Any change to the input randomizes it**
  - “Simple” input changes, e.g., one char, lead to “dramatic” result changes



Recall: Bitcoin Header

- **Version number: Used to keep track of upgrades and changes in the protocol**
- **Previous block header hash: Linkage to the previous block, secures the chain**
- **Timestamp: Number of seconds since the first of January 1970**
- **Merkle Root: Root hash of the generated Merkle tree**
- **Nonce: Used to vary the output of a cryptographic function. It is the value that is altered by the miners to try different permutations to achieve the difficulty level required**

```
hash(block w/ nonce = 1)  => a80a8…8d732
hash(block w/ nonce = 2)  => f7bc9…5345f
hash(block w/ nonce = 3)  => ea758…ae629
hash(block w/ nonce = 13) => 0ebc5…37a66
```



URL: b.socrative.com/student/

Room: HSR

Hash-based Proof of Work

- **To find a hash with N zeros (in hex) at input start, it requires $2^{4^N} = 16^N$ computations, which proves computational work performed**

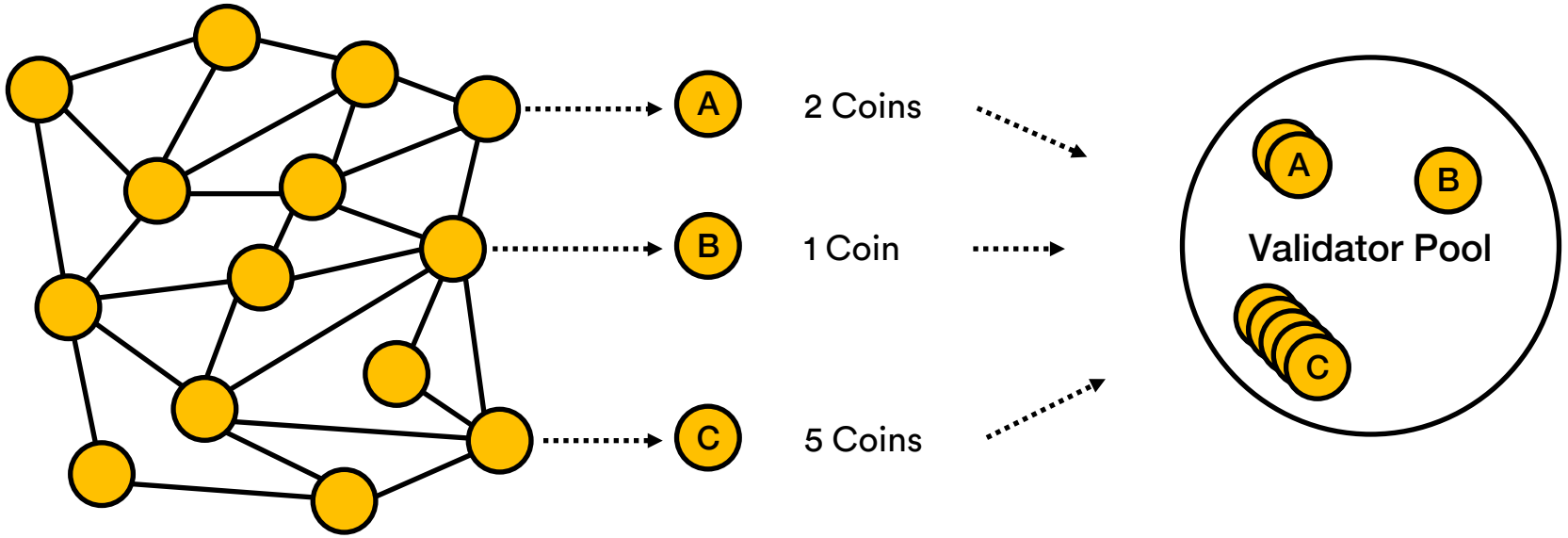
```
in 3e-05 seconds, nonce = 0 yielded 0 zeros.          value = 4c8f1205f49e70248939df9c7b704…
in 0.000138 seconds, nonce = 12 yielded 1 zeros.      value = 05017256be77ad2985b36e75e486a…
in 0.000482 seconds, nonce = 112 yielded 2 zeros.     value = 00ae7e0956382f55567d0ed9311cf…
in 0.014505 seconds, nonce = 3728 yielded 3 zeros.    value = 000b5a6cfc0f076cd81ed3a606820…
in 0.595024 seconds, nonce = 181747 yielded 4 zeros.  value = 0000af058b74703b55e27437b89b1…
in 3.491151 seconds, nonce = 1037701 yielded 5 zeros. value = 00000e55bd0d2027f3024c378e0cc…
in 32.006105 seconds, nonce = 9913520 yielded 6 zeros. value = 000000 77a77854ee39dc0dc996dea…
in 590.89462 seconds, nonce = 186867248 yielded 7 zeros. value = 0000000225060b16117b23dbea9ce…
in 4686.171007 seconds, nonce = 1424462909 yielded 8 zeros. value = 00000000 2dd743724609a9f57260e…
```

Proof of Stake

- **Motivation: Mining requires a great deal of computing power to run different cryptographic calculations to unlock the computational challenges**
  - Bitcoin's current estimated annual electricity consumption* (TWh): 73.12
  - Country closest to Bitcoin in terms of electricity consumption: Austria
- **Proof-of-Stake algorithms achieve consensus by requiring users to stake an amount of their tokens so as to have a chance of being selected to validate blocks of transactions, and get rewarded for doing so.**
  - First cryptocurrency to implement PoS was Peercoin
  - Ouroboros was the first scientifically peer-reviewed PoS protocol
- **The more a user stakes, the better their chance of being selected since they'd have more skin in the game—acting maliciously would see them set back by a greater amount than someone who stakes less.**

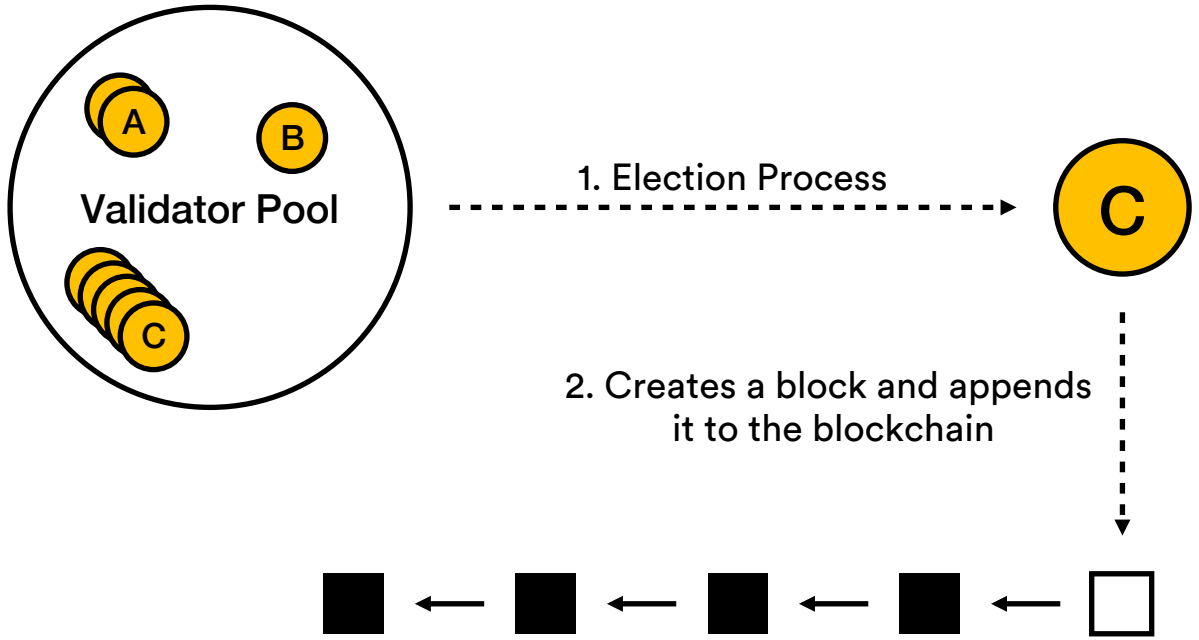
Proof of Stake

- **Nodes wishing to participate in the validation process put money on *stake*, i.e., joining a set of validators ("lottery").**



Proof of Stake

- **The more coins a node invests, the higher its stake; and the higher the chance of winning the "lottery".**
- **If the validator turns out to be adverse, and other validators notice it, his or her money on stake will be slashed, i.e., the adversary's money is gone**

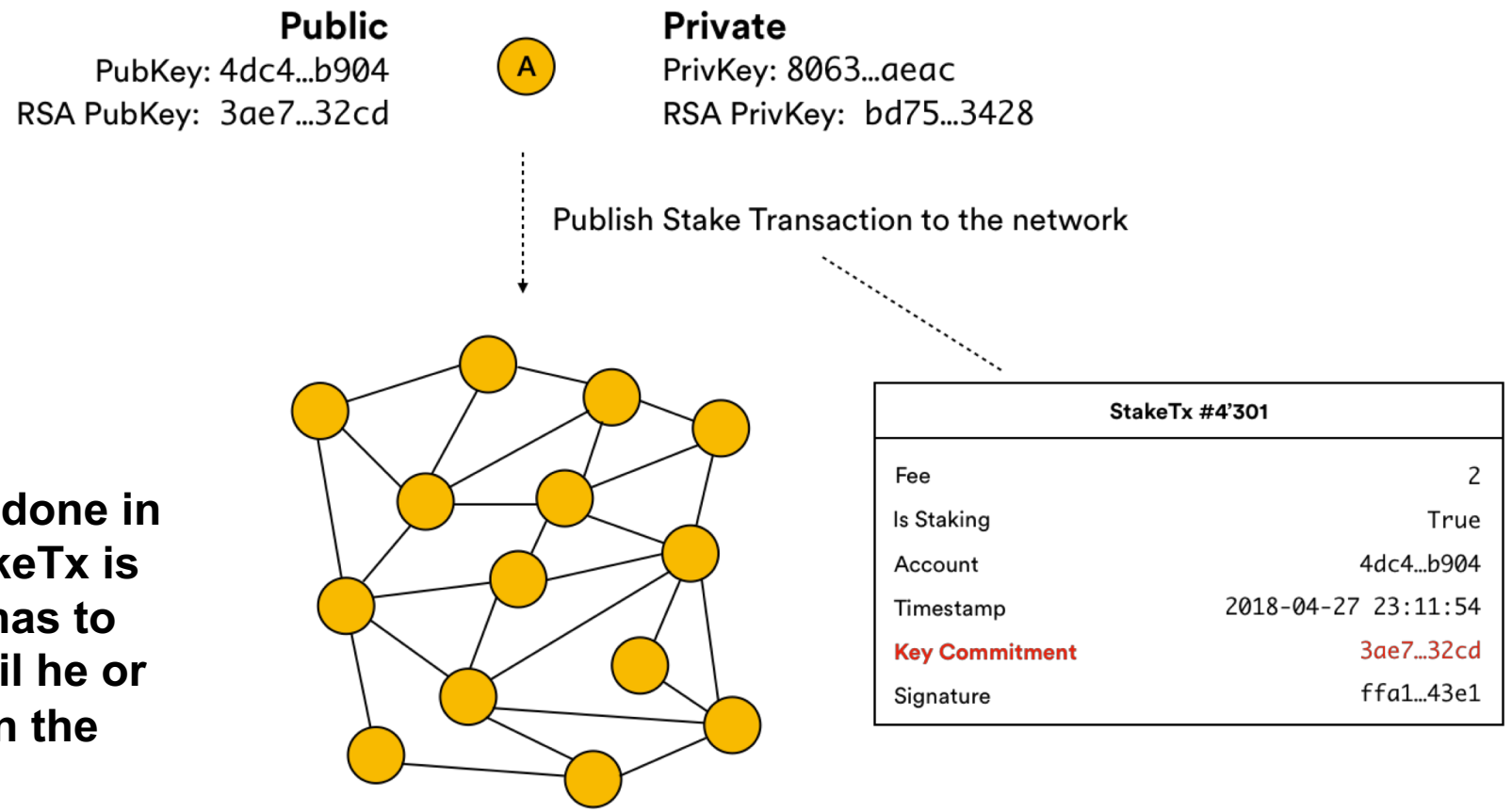


Note: In some blockchains, after node C got elected and appended a block, it must wait for a certain amount of time.

Proof of Stake in Bazo

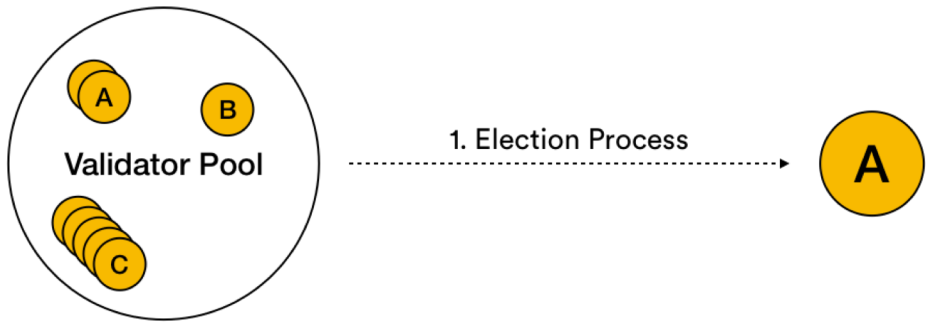
- Bazo uses a chain-based PoS approach, that is, the algorithm randomly selects a validator and assigns him or her the right to append a new block to the blockchain
- In order to become a validator, a user generates an RSA keypair (pk_{comm}, sk_{comm}) and publishes a stake transaction (StakeTx) containing the public key pk_{comm} to the network
- A transaction of this type is accepted by the network if the issuer fulfills the minimum staking amount that is needed
- After the minimum waiting time elapsed, each validator participates in the election process in a non-interactive way

Proof of Stake in Bazo



Note: This has to be done in advance. After a StakeTx is published, the user has to wait for X blocks until he or she can participate in the validation process.

Proof of Stake in Bazo



4.3 PoS Condition

The introduced PoS condition in Section 2 is updated to

$$\frac{\text{SHA-256}([Proof_{PrevBlocks}] \cdot Proof_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target. \quad (15)$$

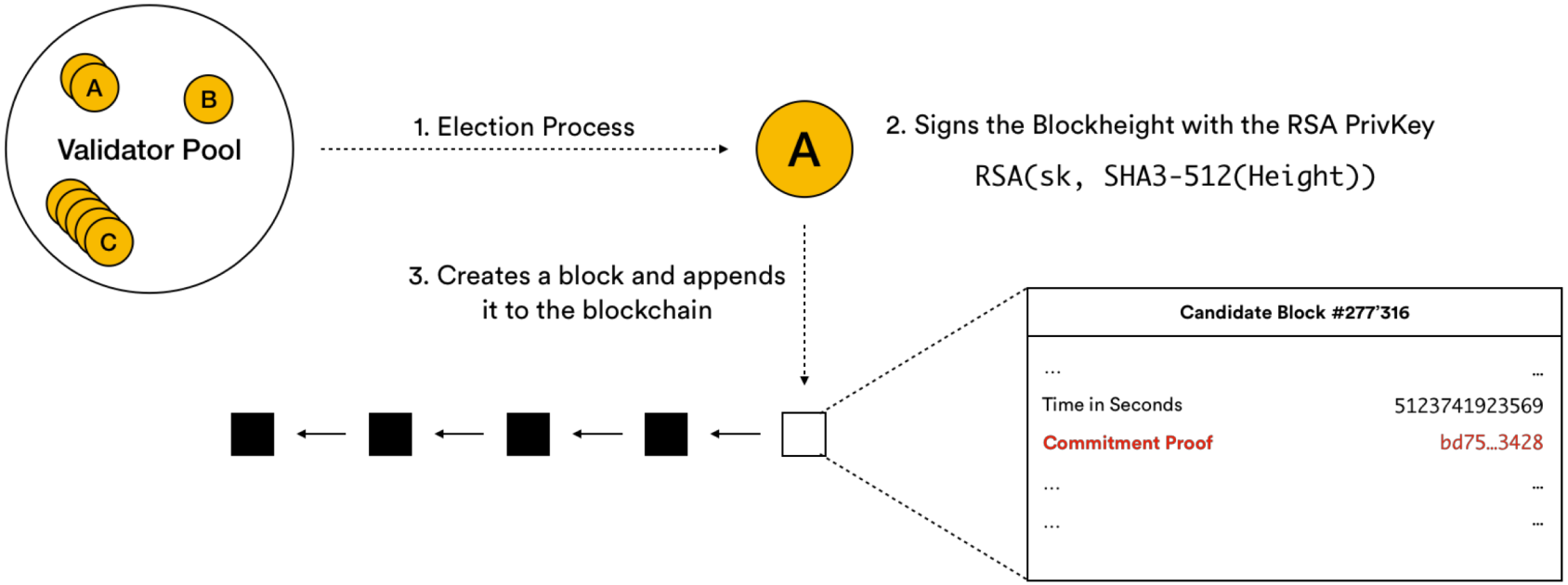
The following parameters changed as opposed to Equation 1:

List of the Previous Proofs ($Proof_{PrevBlocks}$) By including a list of the previous proofs a stake grinding attack becomes infeasible.

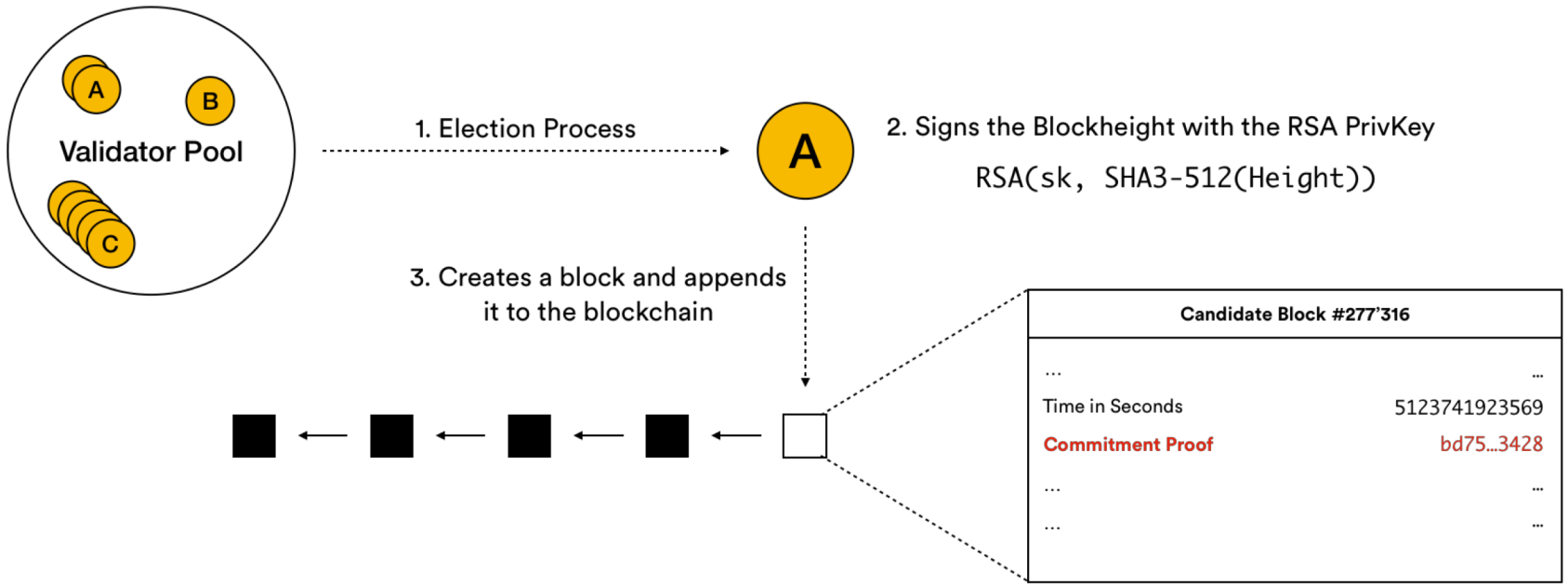
Local Proof ($Proof_{Local}$) All parameters in Equation 15 are the same for each validator in the network except the local proof. The local proof individualises the PoS condition to each validator and therefore, a validator with a low amount of coins also has the possibility to append a block to the blockchain.

Note: Contrary to PoW, a validator is limited to exactly 1 H/s (hash per second) by providing a valid TimeInSeconds to fulfill the PoS condition

Proof of Stake in Bazo



Proof of Stake in Bazo



Note: A validator who appended a block to the blockchain has to wait for X blocks until he or she can take part in the election process again.

SCALABILITY

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

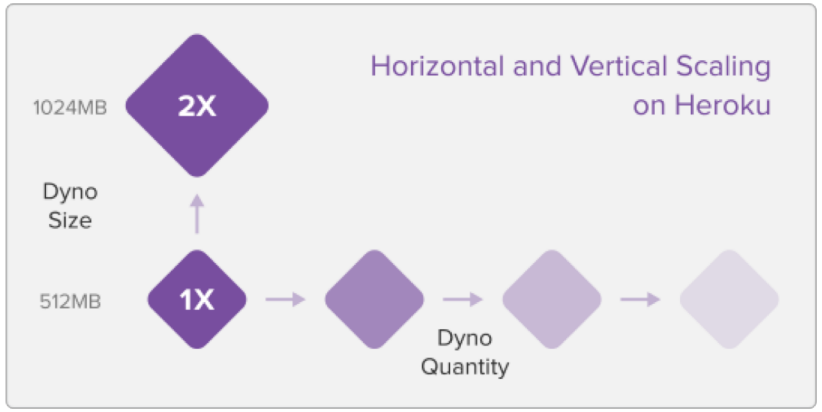
FHO Fachhochschule Ostschweiz

What is Scalability?

- **“A system is said to be *scalable* if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity.”**
- **If a system is expected to grow, its ability to scale must be considered when the system is designed (naming, authentication, authorization, accounting, communication, use of remote resources)**

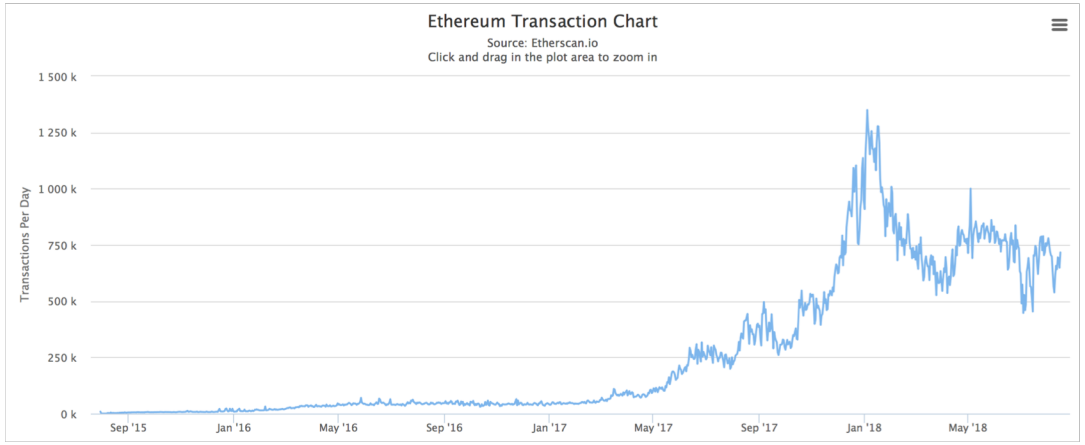
Two ways to scale computer systems:

- **Scale up (vertical): add more resources to a single computation unit, i.e., better CPU, more RAM**
- **Scale out (horizontal): add additional computation units to split workload across units**



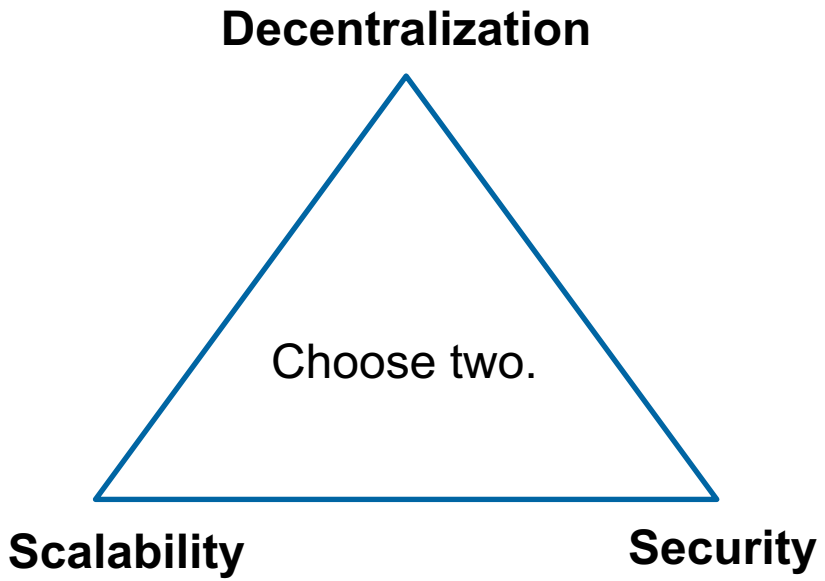
Motivation and Problems

- **Situation: Bitcoin can process roughly 8 transactions per second (tps), Ethereum around 15 tps**
  In comparison, Visa could process up to 65k tps
- **More nodes = more power. So more speed, right? Not exactly.**
- **Every single node must process every single transaction.**
  - Higher security and decentralization, less scalability
- **Divide transactions to different groups of nodes**
  - Smaller group of nodes are more vulnerable
  - Increases scalability, decreases security and decentralization



The (Blockchain) Trilemma

- **A trilemma between decentralization, scalability and security.**
- **Examples of distributed systems:**
  - Bitcoin: decentralized, secure, but not scalable
  - Google: scalable, secure, but not decentralized
  - BitTorrent: decentralized, scalable, but not secure



URL: b.socrative.com/student/

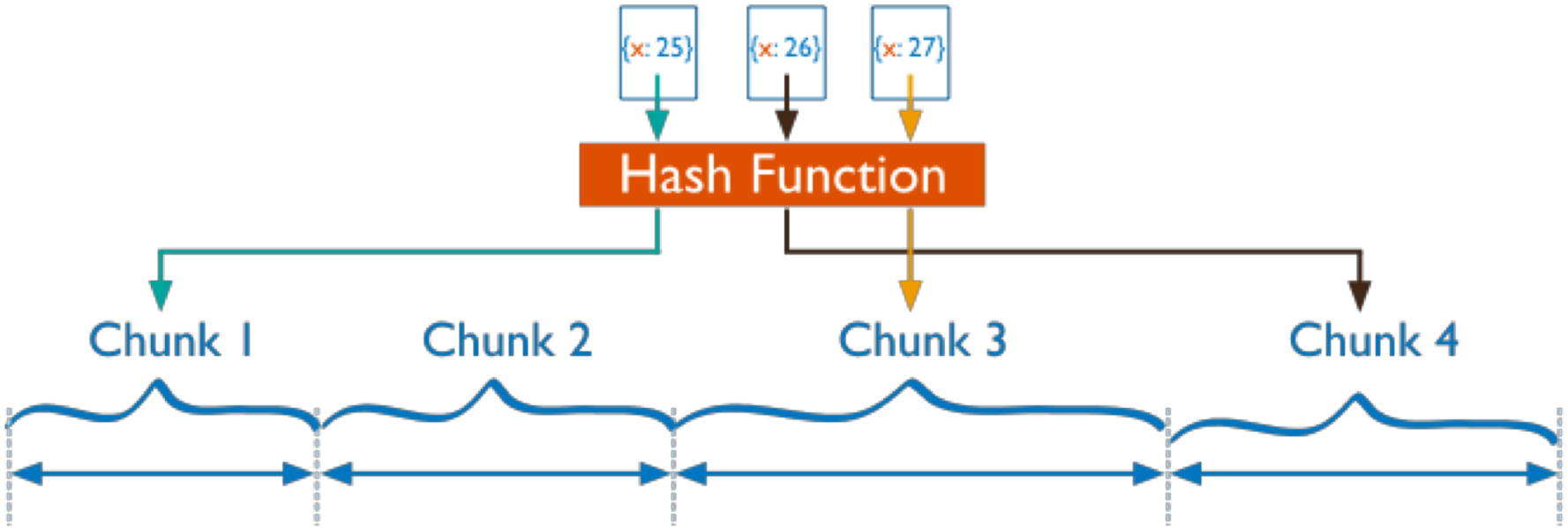
Room: HSR

Blockchain Scalability Solutions

- **Plasma**
  Connects parent and child chains to the main blockchain and transforms it into a tree-structure
- **State Channels**
  A two-way communication channel to send transactions off-chain
- **Sharding**
  Divides the network state into partitions, where each node is responsible for a single partition

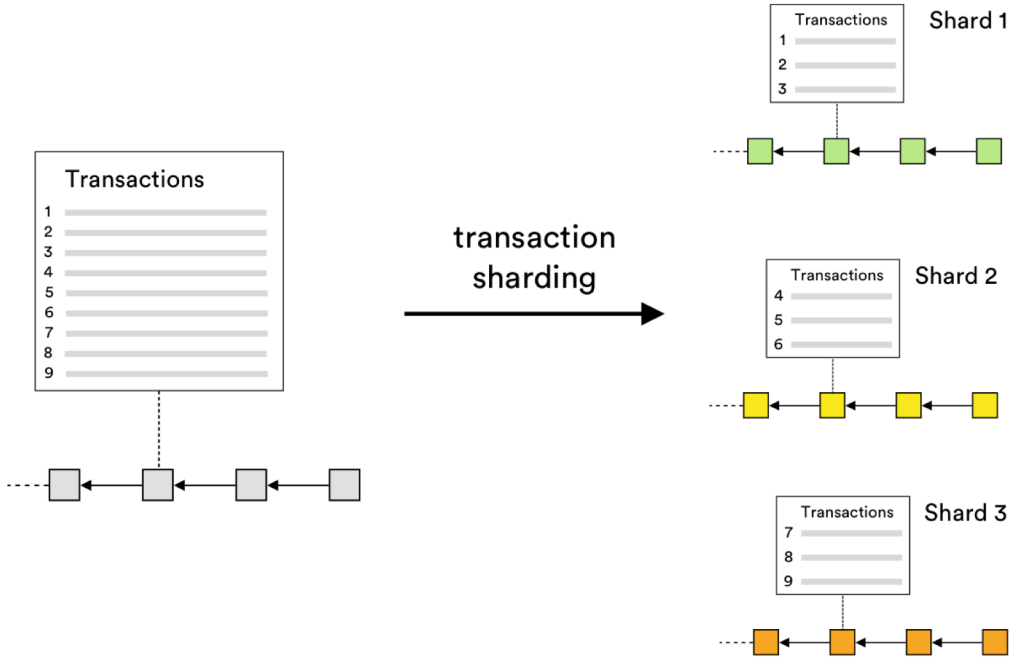
Database Sharding

- **Compute a hash of the shard key field's value**
- **Each chunk is assigned a range based on the hashed shard key values**
- **Each chunk is held on a separate database server instance**



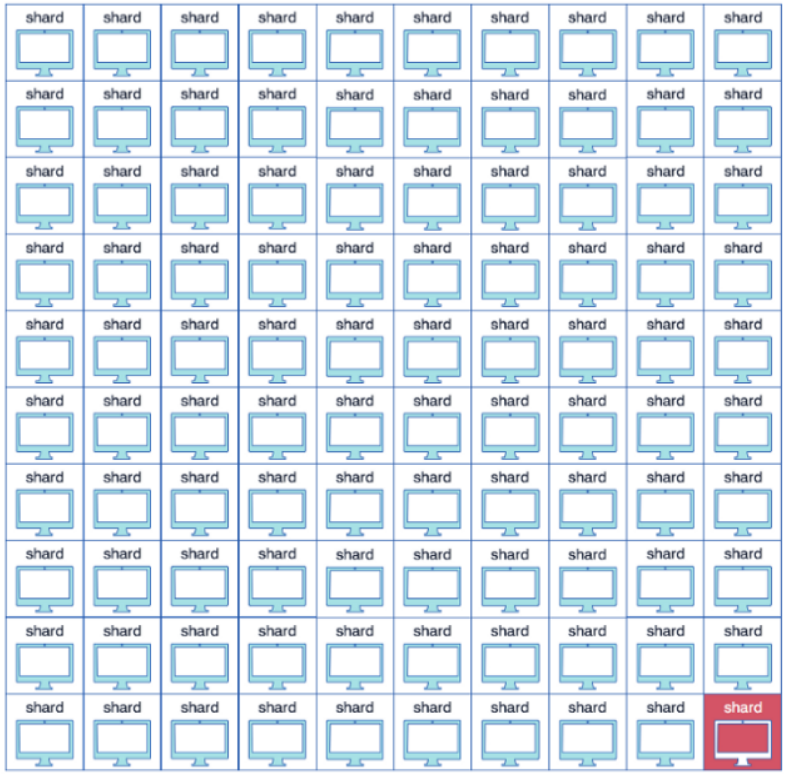
Blockchain Sharding

- **The transaction load on the network is divided into different shards**
- **Only designated nodes validate the transaction, and not the entire network**
- **The blockchain should scale linearly with every new node added to the network.**
- **Higher scalability without sacrificing security or decentralization**



Single-Shard Take-Over Attack

- **Problem: Traditional sharding protocols do not take into account a Byzantine setting**
  Nodes must be resilient against malicious attackers and independent nodes have to reach consensus over the current state of the network.



1% Attack

> In 100 shards system, it takes only 1% of network hash rate to dominate the shard.

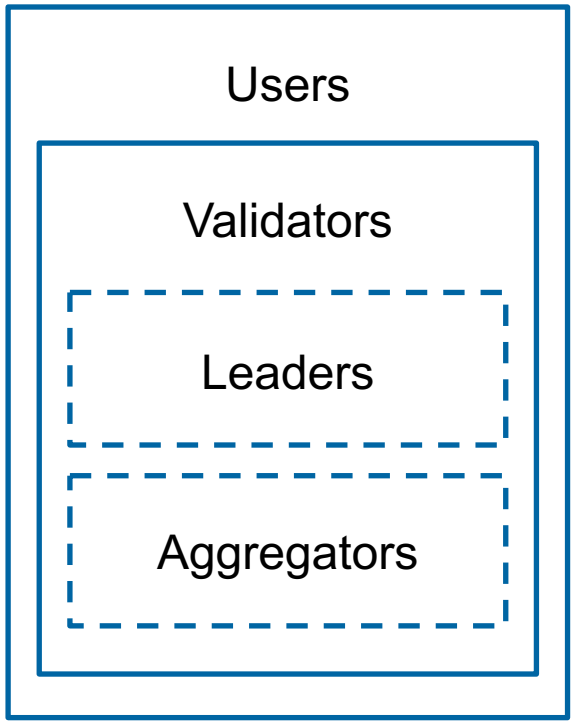
Sharding in Bazo

A conceptual overview on how to achieve scalability for the Bazo Blockchain with Sharding:

- **Dynamic Load-Balancing**
  Computational work is equally divided by dynamically adjusting the number of partitions (shards)
- **Self-Contained Proofs**
  Miners can verify the validity of transaction without the blockchain history
- **Transaction Aggregation**
  Automatically reduce the overall size of the blockchain over time

Overview of Entities

- **Users**
  Uses Bazo's infrastructure to transfer funds and run smart contracts
- **Validators**
  Participates in Bazo's consensus protocol and validates blocks
- **Leaders**
  A special type of validator who has the right to append the next block to a random shard
- **Aggregators**
  A special type of validator who aggregates transactions of a shard

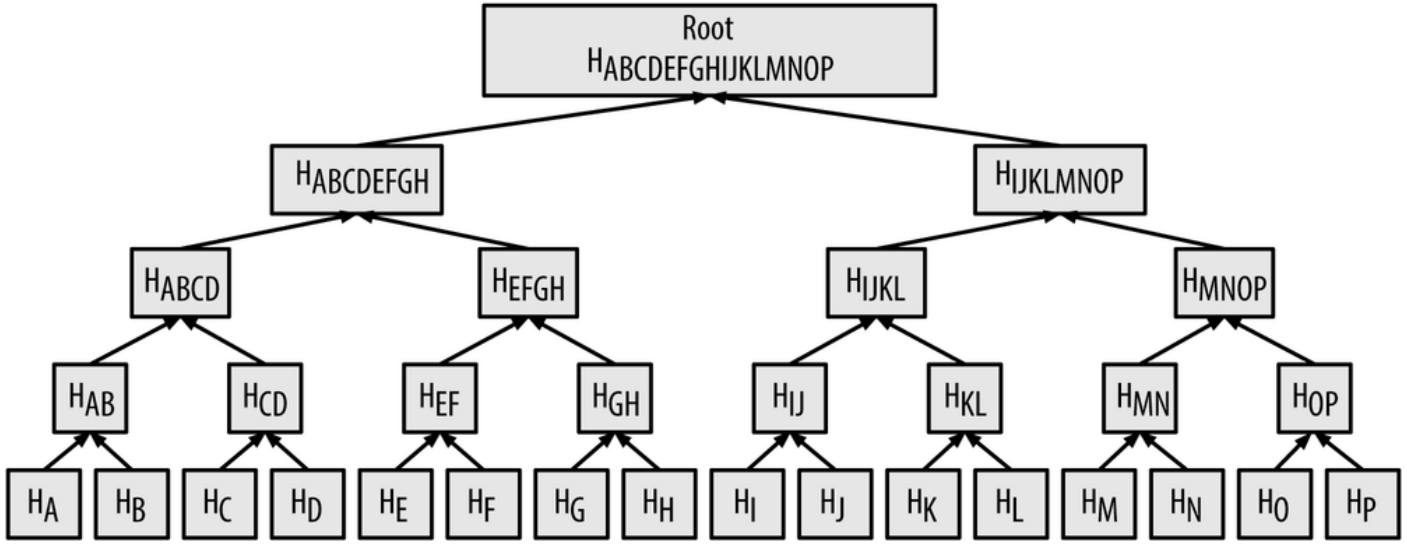


Users

Assume user *U*.

- ***U* uses Bazo's infrastructure to send/receive coins and run smart contracts.**
- ***U* has a wallet with a public-private keypair** (pk_{wall}, sk_{wall})**.**
- ***U* creates a transaction by including a self-contained proof.**

Recall: Merkle Trees



- **A Merkle tree is a *binary hash tree* containing N leaf nodes**
- **Constructed bottom-up, i.e., $H_{AB} = Hash(HA + HB)$**
- **Used to summarize all transactions in a block**
- **To prove that a specific transaction is included in a block, a node only needs to produce $log_2(N)$ hashes, constituting a merkle path connecting the specific transaction to the root of the tree.**

URL: b.socrative.com/student/

Room: HSR

Self-Contained Proofs (SCPs)

- **Traditionally, miners require the full blockchain to validate a transaction of a user**
  e.g. does the user have sufficient funds to execute the transaction
- **Instead of validators requiring the full blockchain to validate a transaction, a user has to provide all required proofs in the transaction in order for validators to verify a transaction, independent of the blockchain.**
- **A SCP is a list of Merkle proofs. A user has to provide one Merkle proof for each block that contains one or more transactions where he or she sent or received funds**

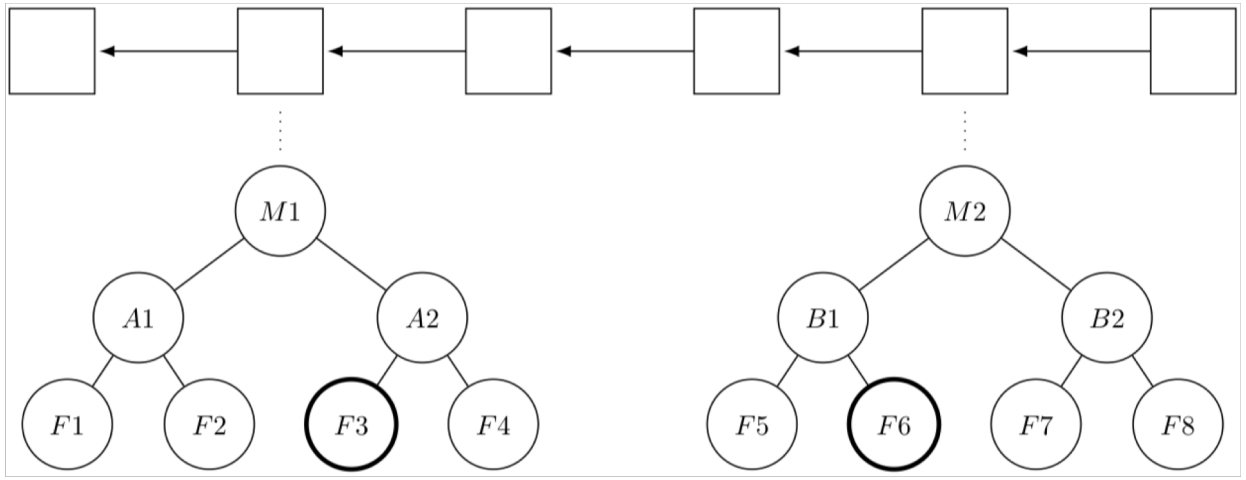
Example of a SCP

- **Assume a user *A* with past transactions *F3* and *F6***
- **A new transaction of user *A* has to include two Merkle proofs, i.e.,**

$$SCP = \{Proof_1, Proof_2\}$$

where

$$Proof_1 = \{F5, F6, B2\}$$
$$Proof_2 = \{F3, F4, A1\}$$

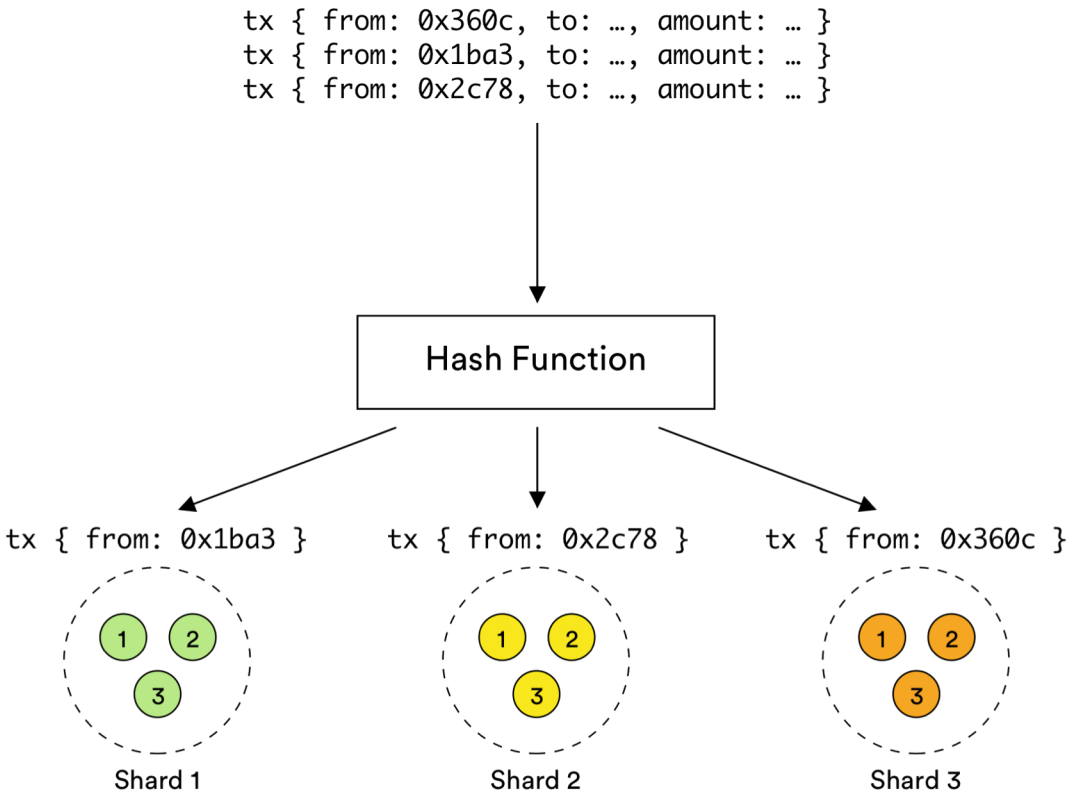


Validators

Assume validator *V*.

In addition to all properties of a user,

- ***V* participates in Bazo's consensus protocol and validates blocks.**
- ***V* has a commitment keypair (pk_{comm}, sk_{comm}) to join the set of validators.**
- ***V* validates blocks based on pk_{wall}.**
- ***V* stores block headers of all shards and block bodies based on pk_{wall}.**

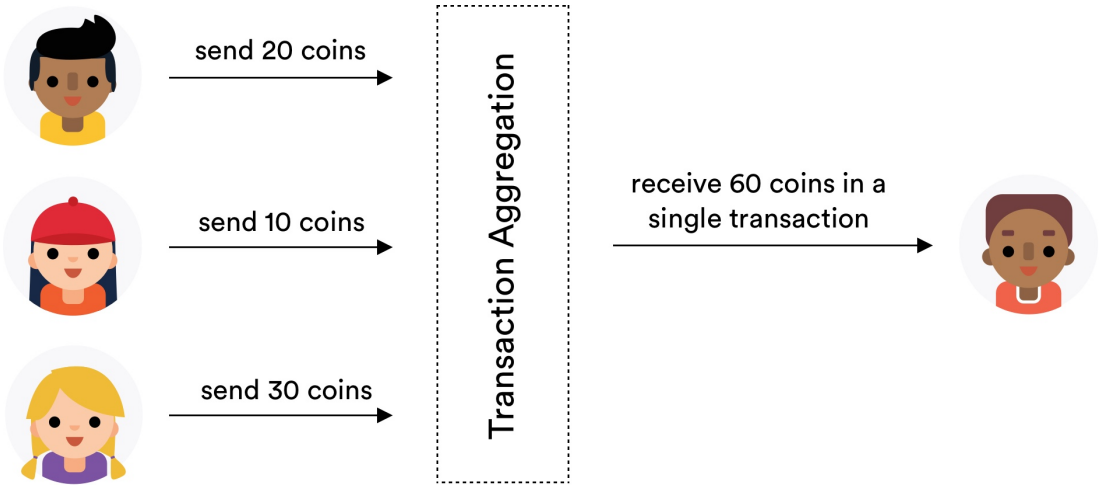


Aggregators

Assume aggregator *A*.

In addition to all properties of a validator,

- **an aggregator is a special type of validator who aggregates transactions of a shard.**
- ***A* creates a special type of transaction called "Aggregation Transaction" by summing up transferred amounts.**
- ***A* aggregates transactions based on $\texttt{pk}_{\texttt{wall}}$.**
- ***A* adds aggregation transactions to a special type of block called "Aggregation Block".**



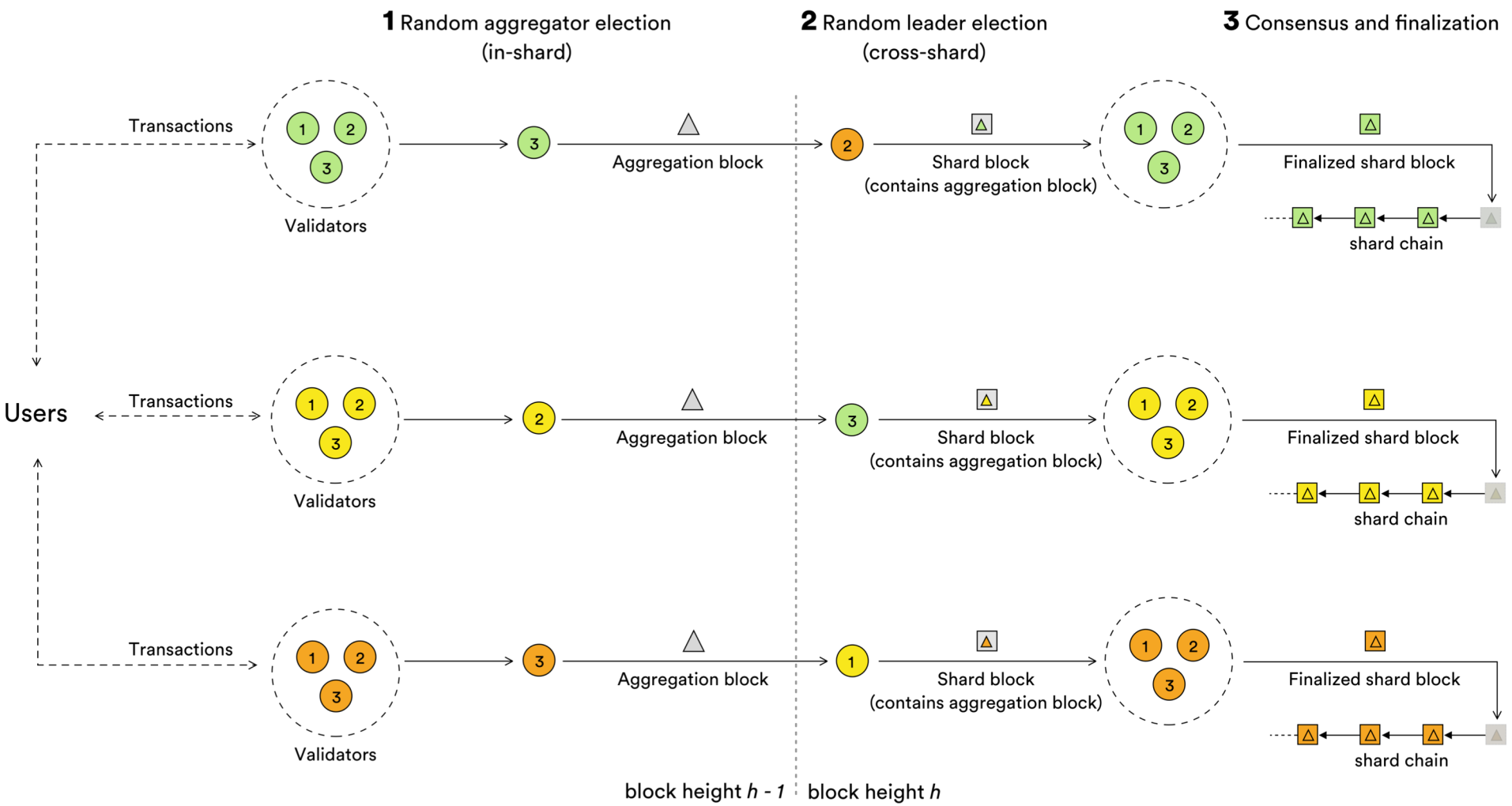
Leaders

Assume leader *L*.

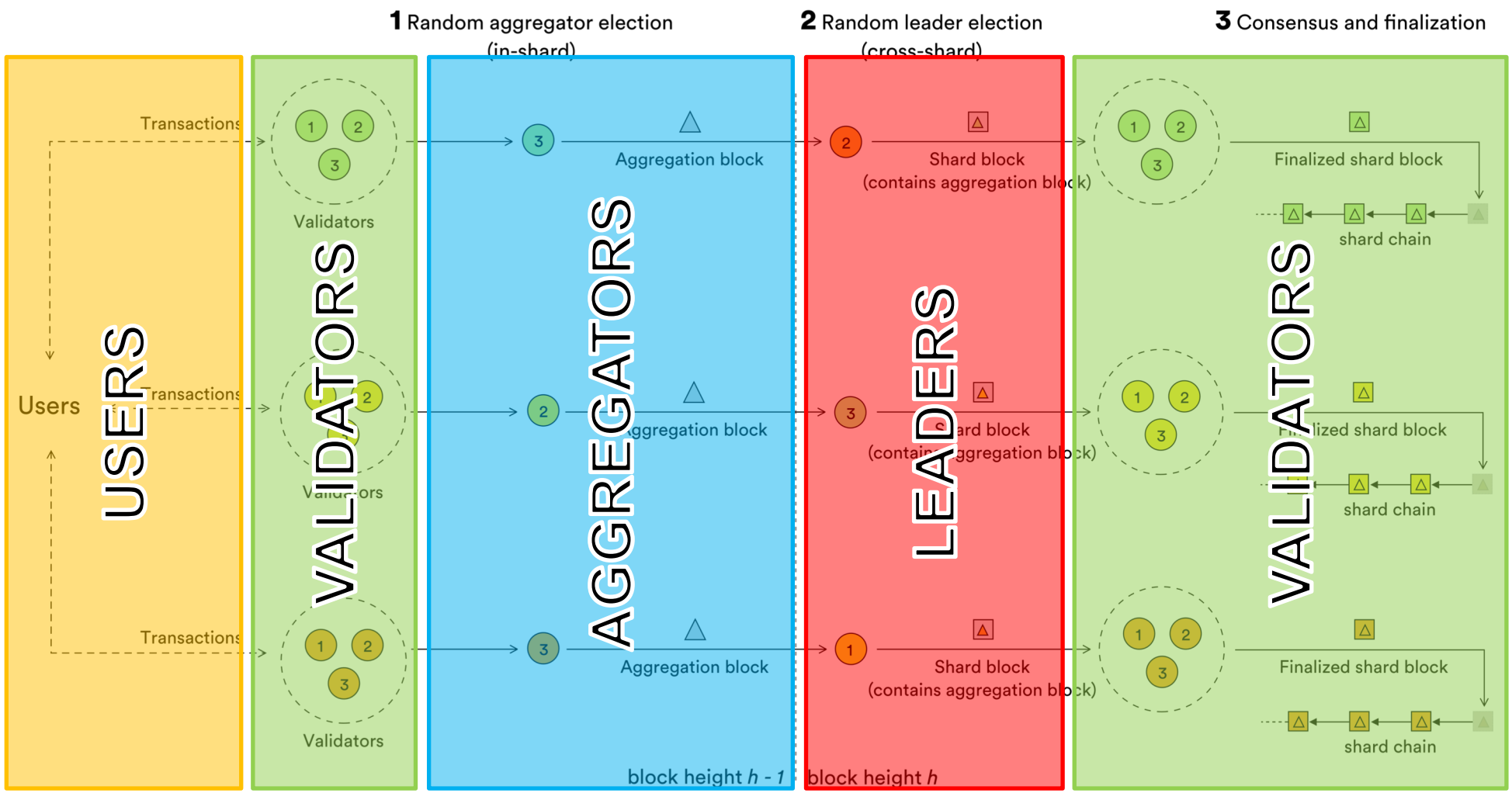
In addition to all properties of a validator,

- **a leader is a special type of validator who has the right to append the next block to a random shard.**
- ***L* participates in the election process to become a leader of a random shard *S*.**
- ***L* validates funds transactions and adds them to a new block *B*.**
- ***L* further validates an aggregation block and includes them in block *B*.**
- ***L* proposes block *B* to validators of shard *S*.**

Protocol Overview



Protocol Overview



But, does it scale?

- **It should.**
- **Transactions are distributed between shards**
  Increases scalability of the network.
- **Computational resources are distributed between nodes**
  Reduces system requirements and cost for running a node.
- **Leaders are elected randomly**
  Mitigate single shard attacks
- **Minimal cross-shard communication**
  Reduces overhead of communication and waiting times between nodes.

Questions?

Roman Blum
roman.blum@hsr.ch

References

- Scale in Distributed Systems
  B. Clifford Neuman, 1994
- Notes on Scalable Blockchain Protocols (version 0.3.2)
  Vitalik Buterin, 2015
- Introduction to Parallel Computing 2nd Edition
  A. *Grama, A. Gupta, G. Karypis, V. Kumar, 2003*
- The Byzantine Generals Problem
  L. Lamport, R. Shostak, M. Pease, 1982
- Cryptographic Sortition for Proof of Stake in Bazo
  R. Blum, 2018
- Scalability for the Bazo Blockchain with Sharding
  R. Blum, 2018
- Sharding in MongoDB, Link: https://docs.mongodb.com/manual/sharding/
  MongoDB Documentation, visited 14th Aug 2018
- Mastering Bitcoin
  Andreas M. Antonopoulos, 2014