

NeCS Winter School, Fai della Paganella, Thursday, January 23, 2020

A Practical Introduction to the Blockchain and DLT World

Burkhard Stiller, Christian Killer, Bruno Rodrigues, Eder Scheid

With many thanks addressed to T. Bocek, M. Franco, S. Rafati, and Dr. E. Schiller

*Communication Systems Group CSG
Department of Informatics IfI
University of Zürich UZH*



**Universität
Zürich^{UZH}**

[stiller|killer|rodrigues|scheid]@ifi.uzh.ch



Outline

1. Background
2. Principles of Blockchains
3. Blockchain Operations
4. Consensus Mechanisms
5. Blockchain Eras
6. Electronic Payment Systems
7. Bitcoins
8. Ethereum
9. Ethereum Wallet – Practical Part I
10. Smart Contract – Practical Part II
11. To Blockchain or not to Blockchain?
12. A Critical Review

Coffee Break

1. Background

Distributed Systems to Remember

- ❑ Distributed systems face very large numbers of
 - Devices, users, applications, ...
 - which determine scalability challenges
- ❑ Any “participant” – hardware, software, individual – is operated or operates autonomously and transparently
 - There is no central control, no single, real-time, unique state
- ❑ Failures happen in a heterogeneous environment and need to be handled (application- or system-specific) such that systems runs securely

Characteristics of Distributed Systems

1. Transparency

- Single view of the system
- Hide numerous details

2. Heterogeneity

- Networks
- Computers (HW)
- Operating systems (SW)
- Programming languages
- Developers

3. Failure Handling

- Detecting
- Masking
- Tolerating
- Redundancy
- Recovery

4. Openness

- Extensibility
- Publication of interfaces

5. Scalability

- Controlling the cost of resources
- Controlling the performance
- Preventing resources from running out
- Avoiding performance bottlenecks

6. Security

- Secrecy, privacy, integrity
- Confidentiality
- Authentication, authorization
- Non-repudiation

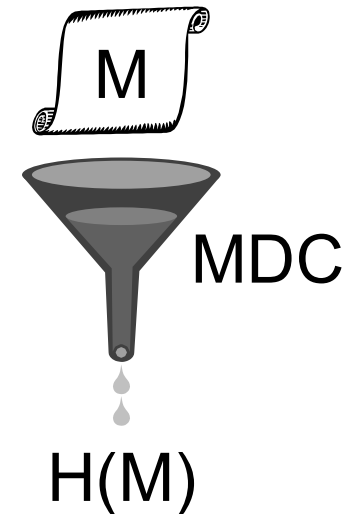
Overlay Networks to Remember

- ❑ ON operates on top of underlay
 - Typically the underlay is the Internet (IP-based)
 - ON is determined by informal “rules and regulations”
- ❑ P2P system instantiates a dedicated overlay
 - Using its own addressing scheme
 - Applying its own high-level communication protocol for peers
 - “Limited” to participating peers following the overlay rules
- ❑ Overlay application
 - Runs within a given P2P system exploiting its functionality

Hash Functions and Hashes

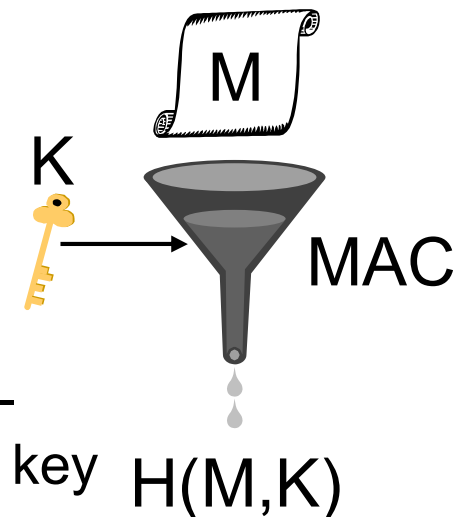
□ Message Digest Code, MDC

- Message M (arbitrarily long) $\rightarrow$ Hash $H(M)$
(hash with a minimum of 128 bit length)
- Note: “One-way” feature of hash function:
 - Efficient generation
 - Very low collision possibility: M, M' with $H(M)=H(M')$
 - Examples: (MD5), RIPEMD-160, SHA-256



□ Message Authentication Code, MAC

- Message M , key $K \rightarrow$ Hash $H(M, K)$
- May be constructed out of MDC
- HMAC (RFC 2104), e.g., HMAC-SH-256
- Cryptographic strength of underlying hash function depending on size of its hash output, size, and quality of key



HMAC: Hash-based MAC, RIPEMD: RACE Integrity Primitives Evaluation Message Digest, SHA: Secure Hash Algorithm

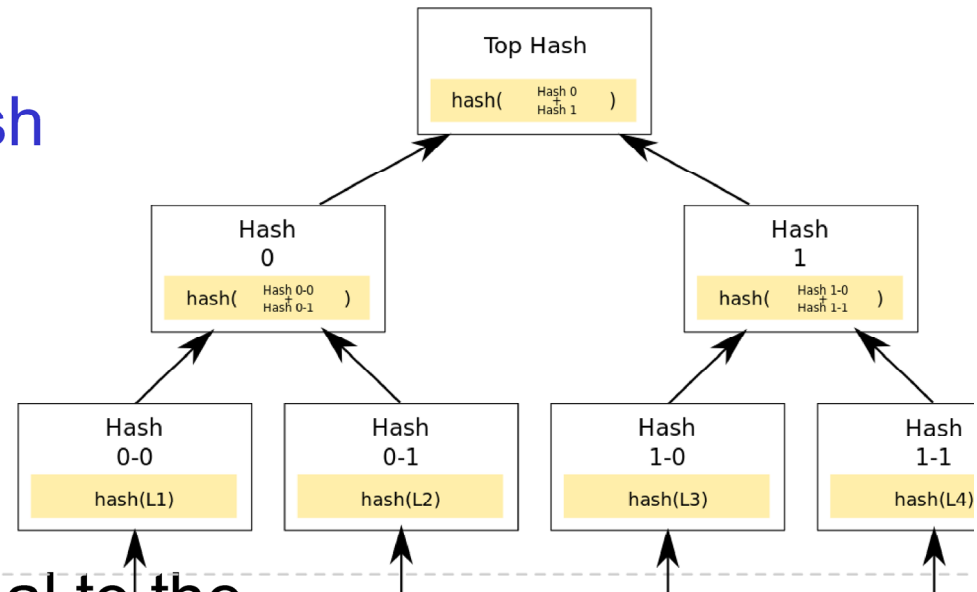
Merkle Trees

❑ Merkle (or hash) tree

- Every leaf node is labeled with the hash of a data block and
- Every non-leaf node is labeled with the cryptographic hash of the labels of its child nodes.
- Top of a tree there is a **root hash**

❑ Characteristics

- **Efficient and secure verification content of large data**
- For binary hash trees:
computing hashes is proportional to the
logarithm of the number of leaf nodes of the tree
- Usually **cryptographic hash functions** used (e.g., SHA-256)



https://en.wikipedia.org/wiki/Merkle_tree#/media/File:Hash_Tree.svg

Nonce

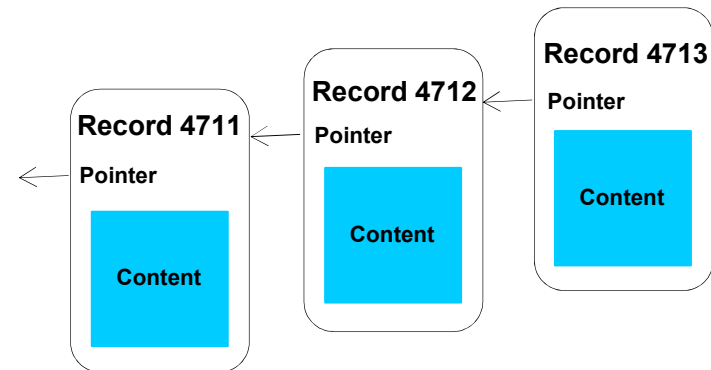
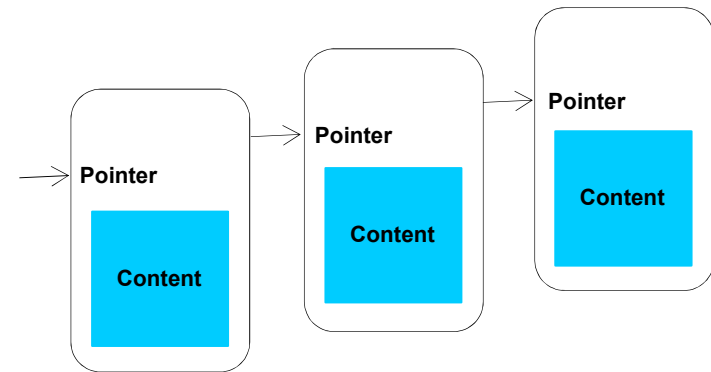
□ Nonce

- Arbitrary number that can be used just once in a cryptographic communication
 - Often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks
- A nonce may also include a timestamp to ensure exact timeliness
 - Prerequisite: clock synchronization between organizations
- Example:
 - **Salt**: random data that is used as an additional input to a one-way function hashing data (password or passphrase)

Data Type “Linked List”

❑ Linear collection of data elements (records)

- Linear order of records is given by pointers to the next record
- Data structure as a group of nodes represents an **implicit sequence**
- Example: **backward linked list**
- Data structure as a group of nodes represents an **explicit sequence** due to record identifiers added



Databases

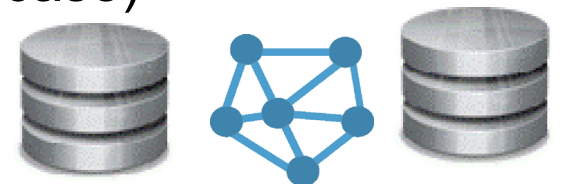
❑ Databases (DB)

- Access control for users and the “SysAdmin” (trusted root)
 - Centralized, physical, and trusted servers are maintained
- Potentially central point of failure, loss, or misuse
- Exchange of records between clients/database via commands
 - SQL statements include: INSERT, SELECT, UPDATE, DELETE, ...
- Databases typically maintain the ACID principle on transactions
 - Atomicity, Consistency, Isolation, Durability (ACID)
 - However, records updated are updated and records deleted are deleted → No storage of any history (standard case)



❑ Distributed DBs

- Every copy runs on a trusted node



2. Principles of Blockchains

The Digitization of Society

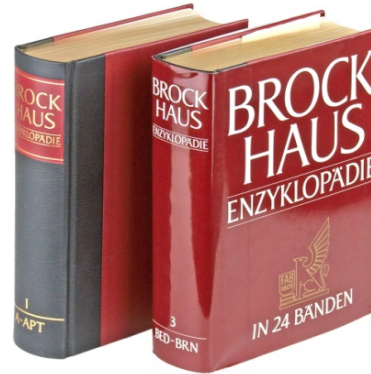
Physical
Objects



Telegram

*Ended Dec 29, 2017
in Belgium*

Since mid 70's, RFC 524



Encyclopedia



Since 2001



Money



Since 2009

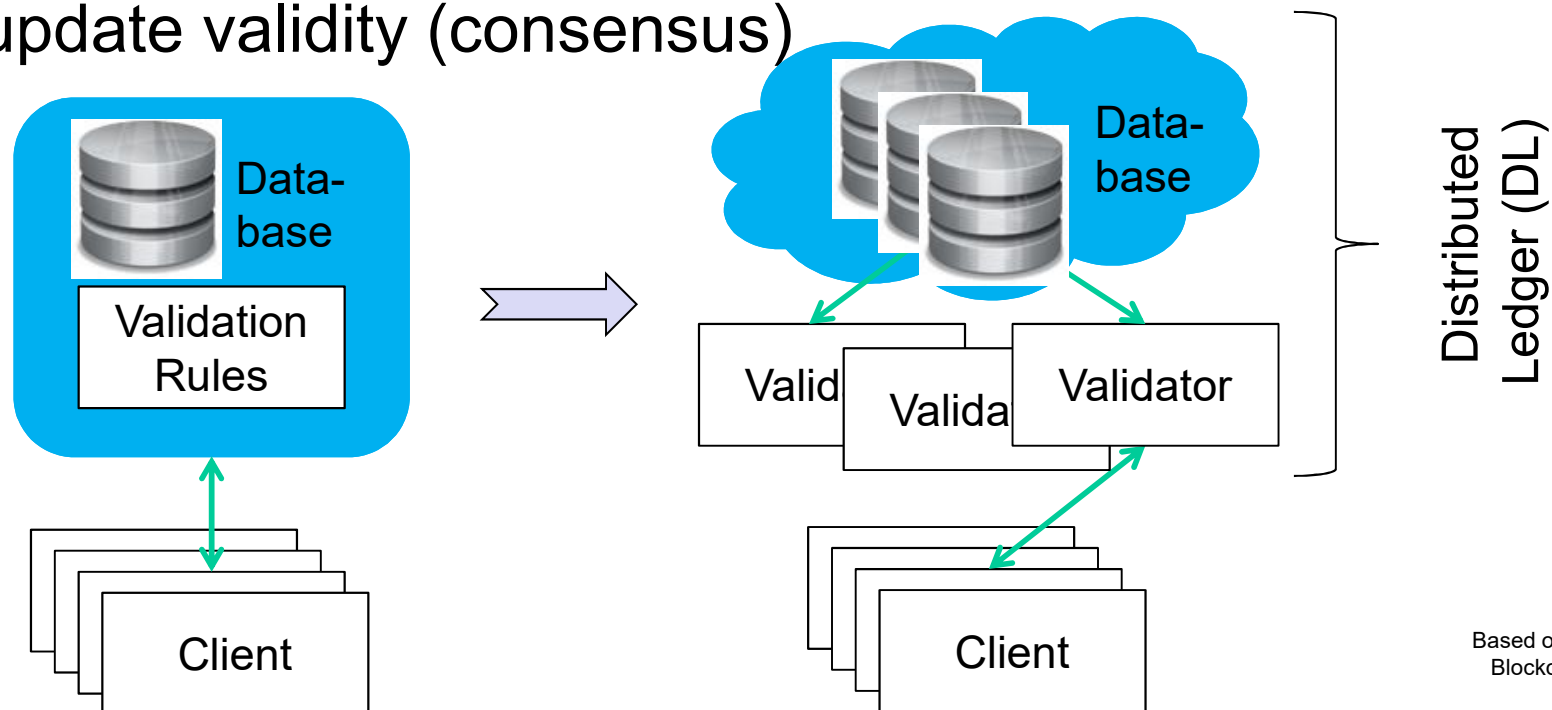
Bitcoins

Digitized
Representations

Systems operated as decentralized and distributed systems!

Key Idea: “Replacing” (Central) Databases

- ❑ Distributed Ledgers **replace** clients' access-protected writes to an authoritative database via validation rules **by** a distributed consensus of decentralized validators
 - where the database's state depends on majority agreement of update validity (consensus)



Based on Terence Spies:
Blockchain Mechanics

DL/BC Types and Terminology – Simplified

❑ Private permissioned



- Read/write/consensus restricted to authorized nodes (pre-defined stakeholders)

– “Enterprise-grade” DL

❑ Public permissioned



- Write/consensus restricted to authorized nodes (pre-defined stakeholders)
- Read open to everyone

– “Controlled collaborative” DL

❑ Private permissionless

- Write/consensus restricted to authorized nodes (pre-defined stakeholders)
- Read partially open

– “Consortium-grade” DL



❑ Public permissionless

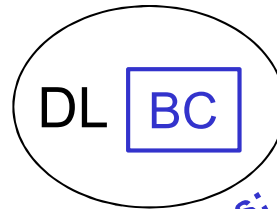


- Read/write/consensus open to everyone
- No restrictions and full transparency

– “Public” BC, the BC

The real and only
blockchain (BC)!

No real blockchains:
“restricted” stakeholders
capabilities!



Definition

- [Distributed Ledgers (DL) or] ^{“Private” BC} **Blockchains** ^{“Public” BC} (BC)
 - Digital records of who-owns-what w/o a central storage



- Consensus Mechanism (CM) ensures that each node's copy of the ledger is identical to every other node's copy



- Write access to BCs by miners or validators (with data from any asset owner) for transactions via CM and cryptographic signatures, read access at no “costs”



- Key **advantages** of (public) BCs
 - Immutable, traceable, and preventing “double spending”



“Who-owns-what” Records

- ❑ A **digital asset** is determined by an electronic representation, such as a source file, which inherently bears the **right of use** of this source.
 - ❑ A **token (digital token)** can be a digital asset, issued by a stakeholder, which gives the holder the right to participate within the network of stakeholders
 - It may allow for “payments” inside that network
 - ❑ A **coin (digital coin)** is a digital representation of value, specifically designed to represent digital “money” within a network of stakeholders, typically the BC, and beyond
 - Counterfeiting and double-spending prevented by cryptography
- ⇒ One can buy a token with a coin, but generally not a coin with a token.

Token Types

- ❑ **Utility tokens** provide access digitally to an application or service by means of a blockchain
- ❑ **Asset tokens** represent assets such as (a) a debt or (b) an equity claim on the issuer
 - *E.g.*, they promise a share in future company earnings or future capital flows – analog to equities, bonds, or derivatives
- ❑ **Payment tokens** are used (a) as a means of payment for acquiring goods or services or (b) as a means of money or value transfer
 - Synonymous to “cryptocurrencies” (thus, a dedicated “coin”)
 - Note: Cryptocurrencies give rise to no claims on their issuer

3. Blockchain Operations

Blockchain Ingredients

❑ Public key cryptography and hashes

– Asymmetric approach for arbitrary users

- Ensures validation and authentication (in turn authorization)



❑ Internet

– Networked infrastructure for everyone

– Distributed system with arbitrary users and devices (nodes)

- Peer-to-peer (overlay network) communication paradigms: protocol
- Storage capabilities for “any”-sized data volumes



❑ Incentives

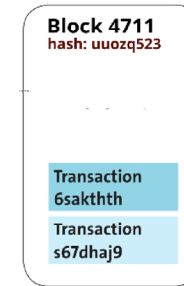
– Supporting rewards for participants’ tasks performed within an overlay network by a “protocol” enabling communications

- Ensures participation of anyone (potentially non-trusted stakeholders)



Blockchain Operations

- ❑ Transactions (content) collected in blocks
 - New blocks created regularly
- ❑ A block contains a hash of and a pointer to the previous block ...
- ❑ Consensus mechanism required to determine the block to be integrated into this blockchain
 - Public blocks contain, e.g., solved crypto puzzles (PoW)
 - E.g., a form of partial hash collisions (SHA-256)
- ❑ Creation of valid blocks performed by anyone (incentive)
 - Solving crypto puzzles $\equiv$ confirmation of blocks $\equiv$ Mining
 - Computational expensive $\rightarrow$ Avoids double spending

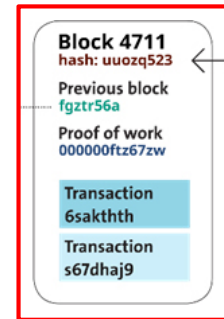


Blockchain



Blockchain Data Structure in Detail (1)

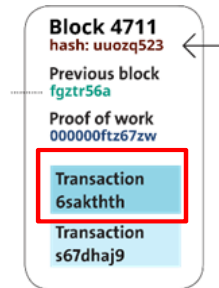
- ❑ BCs are a **backward-ordered, linear list** of blocks
 - Chain starts with **genesis block** to which others are back-linked
- ❑ Blocks **contain** (at least)
 - Transaction (tx) data (content, payload)
 - Pointer to and a hash of the previous block
 - Cryptographically hashed value of crypto puzzle (result of PoW)
 - Time stamp
- ❑ BC's structural and technical **characteristics**
 - Chain may show **side chains**, but only one valid branch finally
 - Chronological order guaranteed by previous block's hashes
 - A BC network is organized as a **peer-to-peer network**
 - Overlay topology may change, replicas of BC are hold on multiple nodes, exchanges of new blocks performed within that overlay, anyone to join



Blockchain Data Structure in Detail (2)

□ Transactions (tx/Tx)

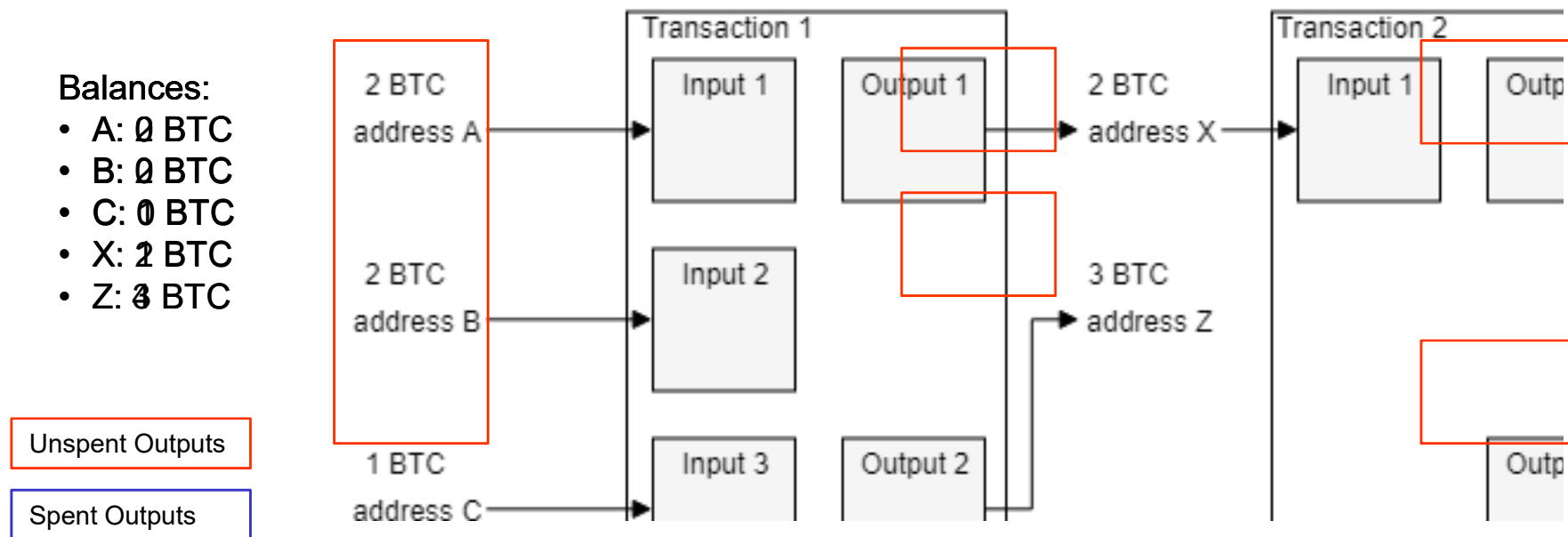
- Data structure encoding the transfer of “value”
 - From a source (“input”) to a destination (“output”)
- Structures are typically not related to accounts or identities
 - Chunk of “value” locked with a specific secret known by the owner
- Example tx content
 - Version of tx
 - Input counter – The number of separate input values
 - Inputs – The values
 - Output counter – The number of separate input values
 - Outputs – The values
 - Locktime – Earliest time at which tx will be valid (to be relayed on the BC network or added to the BC), typically a Unix timestamp



UTXO or account

Unspent Transaction Output (UTXO)

- ❑ Concept based on physical money, e.g., bills
 - Balance is calculated by total of unspent outputs
 - Each transaction has inputs (**unspent outputs**) and outputs
 - Possible to combine one or more outputs as inputs to form a transaction



Account-based Model

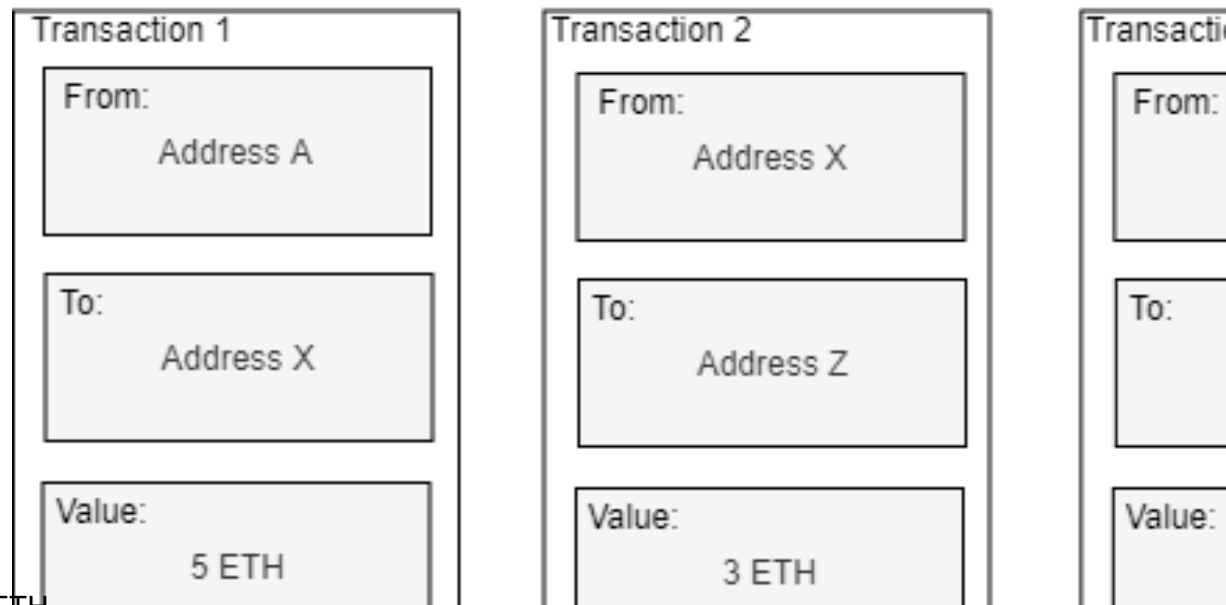
- ❑ Account maintained by a global state
- ❑ Similar concept to a debit card
 - Each transaction **modifies the state** of accounts
 - Balance should be larger or equal than spending amount



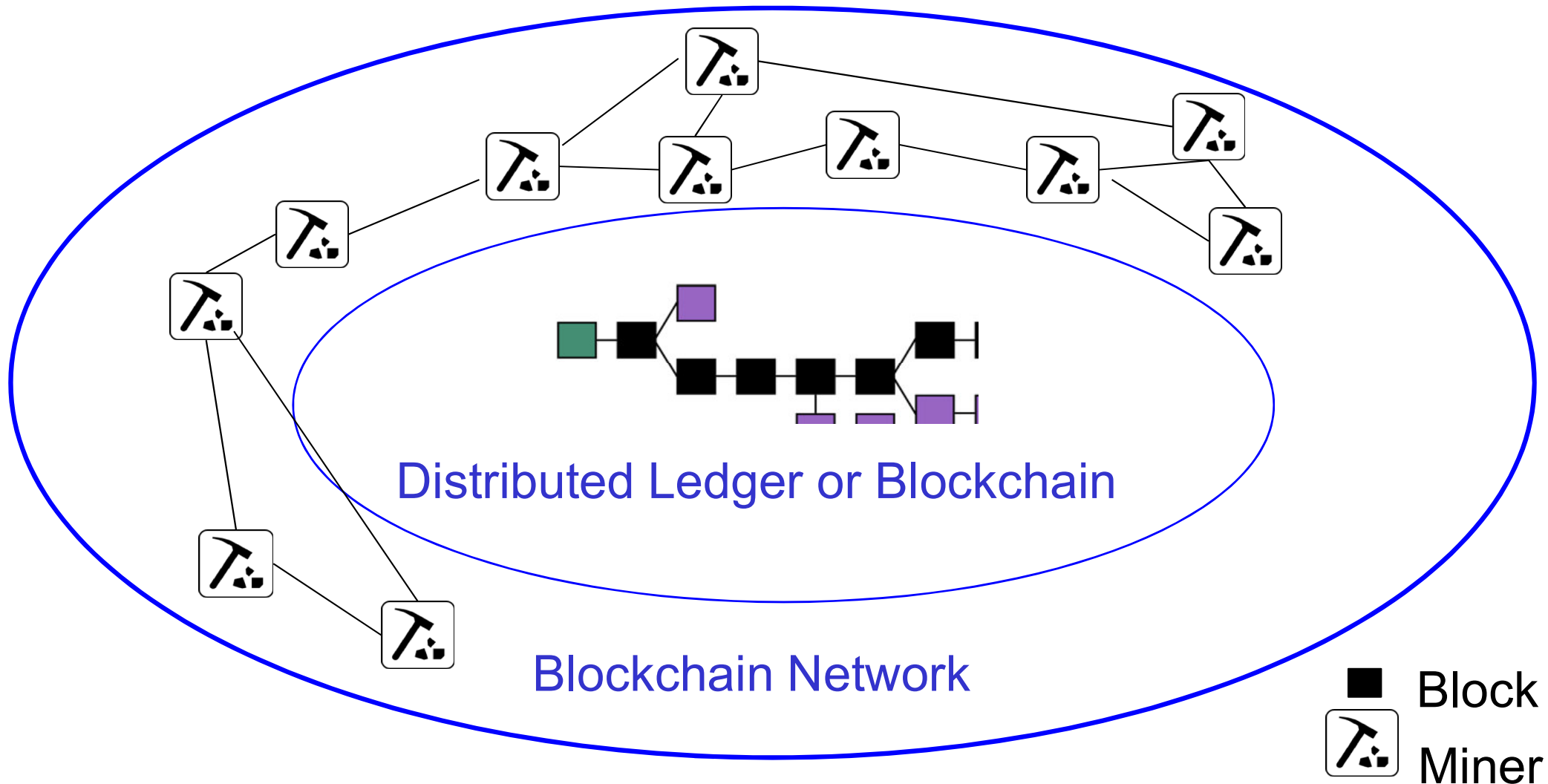
Balances:

- Account A: 0 ETH
- Account X: 3 ETH
- Account Z: 3 ETH

Account A's initial balance: 5 ETH



Blockchain Terminology (1)



Chain: Backward-linked list of cryptography-based (hash-secured) pointers to previous blocks

Blockchain Operations in Detail

For illustrative reasons, the Bitcoin BC (BTC) is used here.

- ❑ Overall operations consists out of 4 steps (in principle)
 1. User A wants to pay user B the amount of 1 BTC (bitcoin)
 - This intent is broadcast as a transaction via the P2P protocol into the BC network
 2. Any miner interested, adds this transaction to his/her respective current block
 3. Any of these minors wins a lottery and “mines” the block
 - That miner solved the crypto puzzle at first
 - Such a block may contain multiple transactions and is mined in well-defined periods of time
 4. The “definitive block “– including the transactions and the new hash – is broadcast to the BC network again as well and added to everyone’s BC copy

Hash Computation and Checks (1)

- ❑ Once a set of transactions becomes available, a **block is created** by utilizing the following data
 - Transaction(s)
 - Hash of previous block
 - Nonce (arbitrary number, used only once)
 - Other information (depending on the BC)
- ❑ The **hash of new block is calculated** meeting the “target”
 - **Target hash** is a number (absolute value) that a hashed block header must be less than or equal to in order for a new block to be awarded
 - Value used to determine the “difficulty of the input”
 - Value adjusted to ensure that blocks are processed “in time”
- ❑ Once the hash was computed, the hash is **broadcast to the BC network** and **checks are performed**

Hash Computation and Checks (2)

- Accepted blocks require the miners to solve a crypto puzzle (PoW): hash
 - Difficult to produce (time-consuming, energy), but easy to verify
 - Validity is ensured by checking if a block's hash value is less than “current target”
 - Hash is above the target value → Another miner may have found a suitable hash (not known yet to others), block attached to local BC only, but miner lost the distributed lottery, nonce will be incremented and a retry starts
 - Hash is below the target value → This miner won the distributed lottery and the new block's hash determines the crypto puzzles' result (PoW result $\equiv$ hash)
- Since each block contains the preceding block's hash, a sequence of those determines a larger amount of work
 - Changing any block would require the regeneration of all successors and redoing the work on the data they contain
 - Thus, at least 6 successors are required to consider a block being valid
- The PoW (hash, result of the crypto puzzle) is validated by other members of the BC network by confirming the new hash before adding the block to their local copy, at 51% of the network hashing power' ok, the block is “definitive”

Blockchain Terminology (2)

❑ Mining (Process)

- The process of BC members trying to solve the crypto puzzle and adding the respective new block onto the BC

❑ Miners

- Those BC members, who run machines to solve crypto puzzles
- Their reward in case of a successful inclusion are tokens of BC
 - In case of the bitcoin BC the reward are BTC (incentives to participate)

❑ Checks

- Verification of hashes broadcast to the BC network

Target Value and Adaptation Algorithm

- 
- ❑ Mining is rewarded → Likely more miners join
 - Higher processing capacity increases likelihood on finding hash earlier
 - ❑ Block creation rate increases, average mining time decreases
 - To maintain the ideal goal of 10 min mining per block: change difficulty, which effects the “target value”
 - ❑ BC network decreases target value to increase difficulty
 - Decreasing target values increases difficulty to find the hash
 - ❑ Block creation rate decreases and average mining time increases again
 - System stabilizes itself again, and continue as of above
 - ❑ Reverse of cycle applies, if block creation rate decreases

Consensus: Hash-based Proof-of-Work (PoW)

Based on Terence Spies:
Blockchain Mechanics

- Key: One cannot compute an input from an output
 - To find a **hash with N zeros at input start**, requires 2^N computations, which proves computational work performed
 - Hashing an incrementing “nonce” as hash input, leads to zeros

in 3e-05 seconds, nonce = 0 yielded 0 zeros. value = 4c8f1205f49e70248939df9c7b704ace62c2245aba9e81641edf...

in 0.000138 seconds, nonce = 12 yielded 1 zeros. value = 05017256be77ad2985b36e75e486af325a620a9f29c54...

in 0.000482 seconds, nonce = 112 yielded 2 zeros. value = 00ae7e0956382f55567d0ed9311cfd41dd2cf5f0a7137...

in 0.014505 seconds, nonce = 3728 yielded 3 zeros. value = 000b5a6cfc0f076cd81ed3a60682063887cf055e47b...

in 0.595024 seconds, nonce = 181747 yielded 4 zeros. value = 0000af058b74703b55e27437b89b1ebcc46f45ce55d6....

in 3.491151 seconds, nonce = 1037701 yielded 5 zeros. value = 00000e55bd0d2027f3024c378e0cc511548c94fbed0e....

in 32.006105 seconds, nonce = 9913520 yielded 6 zeros. value = 00000077a77854ee39dc0dc996dea72dad8852afbde6....

in 590.89462 seconds, nonce = 186867248 yielded 7 zeros. value = 0000000225060b16117b23dbea9ce6be86ac439d....

in 4686.171007 seconds, nonce = 1424462909 yielded 8 zeros. value = 000000002dd743724609a9f57260e2492908d....

- The **distributed game sets the difficulty N** of the game
 - Players get a list of blocks and their content (see slide before)
- Players **accumulate “points”** by creating blocks
 - Hashing the previous block, finding a hash of the new block with enough zeros, and transmitting this block to everyone

Blockchain Terminology (3)

□ Difficulty

- Measure of how difficult it is to find a new block compared to the easiest it can ever be
- Recalculated every 2016 blocks in the BTC BC such that the last 2016 blocks would have been calculated within two weeks
 - On average, one block is mined every 10 min

□ Consensus

- State reached where the majority of members of the same P2P network agrees on the same mining output (51% agree)
- This state of the consensus is secure and tamper-resistant, immutable with respect to the blocks
- Respective block data is persisted to the BC network's nodes

Communications – The Flood Protocol

- ❑ New transactions and mined blocks are broadcast to the BC network
 - Minimum of 8 neighbors maintained in Bitcoin P2P network
- ❑ Temporary BC splits possible
 - 2 miners arrive at 2 different, but valid hashes at the same time
 - P2P network resolves that split in short time to 1 valid branch
 - Clients only accept “longest chain of blocks” as valid (pruning)
 - Length is determined by the most combined difficulty, not #blocks
- ❑ As soon as block gets “solved” as valid, every miner in the BC network must stop the current mining process on that block and can start with the next one

Transaction (tx/Tx) Fees

- ❑ Mining and consensus finding require resources
 - Specifics depend on the dedicated approach used
- ❑ Generally, tx fees are part of the tx
 - Compensation for miner to mine
 - Tx fees are collected by that miner
 - Tx fees are set by the BTC's market forces
 - Tx fees incentivize a miner to include a tx into a block
 - High fees: early selection; low fees: delayed selection
 - No requirement for tx fees, but mining may be delayed “forever”
 - Tx fees are countermeasures against “spam” as providing no tx fees inside a tx may not lead to its selection
 - Tx fees are typically calculated by the size of the tx (in Byte)
 - A calculation of the tx fees based on the tx values is not applied

Blockchain Transaction Types

❑ On-chain tx

- Available on the blockchain visible to all nodes on the BC
- Tx valid when a communicated that tx across the network
 - Tx times may vary depending on the network load or tx queued

❑ Off-chain tx

- Value resides outside of the blockchain
- Tx executed instantly

❑ On-chain tx with off-chain storage of data

- The same is valid as for on-chain tx
- Additionally, the tx does not contain the full data, but only a hash of the data, which is stored off-chain
 - Reduces storage size of BC for many tx or large volume

Blockchain Addresses (1)

- ❑ Transactions on a blockchain require the knowledge of the **blockchain address of the sender and recipient**
 - Address as **27-34 digit code** consists of letters and numbers
 - Generated by the wallet, where public-private key pairs are stored, from public key via cryptographic hash function (“finger print”)
 - Usually currency-specific, addresses belong to a certain blockchain
 - Example Bitcoin
 - $A = \text{RIPEMD160}(\text{SHA256}(K))$
with A Bitcoin address and K public key
 - Results in a 160 bit (20 Byte) number
 - **Note:** A public key **IS NOT** the same as a Bitcoin address, but this is derived from a key with a one-way hash function!

Blockchain Addresses (2)

□ Address representations

– Base58 encoding (subset of Base64)

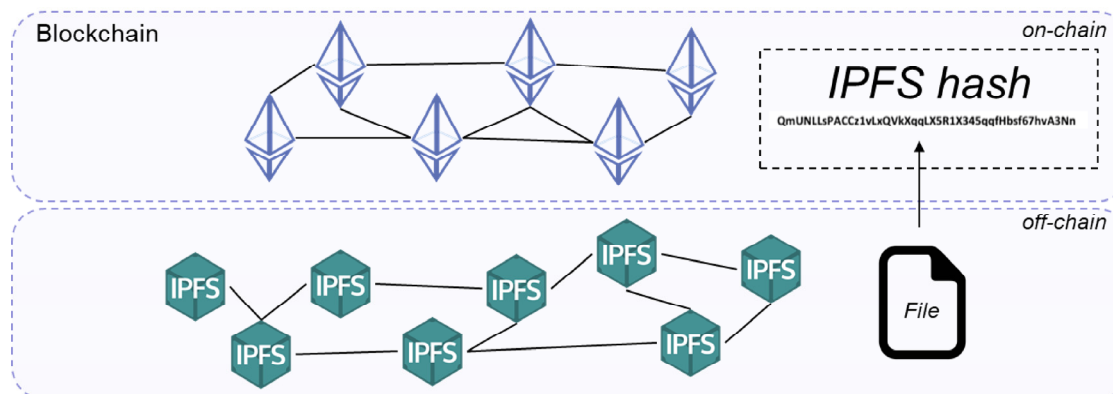
- Text-based binary-encoding format, using upper- and lowercase letters, but omitting “0” and “O”, “l” and “I”, and “\”, “/”, and “+”
- Balancing compact representation, readability, and error detection

– Base58Check encoding

- Adds 4 Byte checksum (error-checking over encoded data) to the add
 - Used for BTC, prevents transcription and typing errors
- Example:
 - Add a *prefix* to *data* (version byte), 0x00_h in the Bitcoin address case 0x80_h in case of a private key to be encoded
 - Compute double SHA checksum (CS):
 $CS = \text{SHA256}(\text{SHA256}(\text{prefix} + \text{data}))$, with “+” concatenation, and use of 4 first bytes of this 32 Byte hash

Off-chain Signaling of Addresses

- ❑ **Problem:** BCs typically show limited storage capacity
- ❑ **Solution**
 - Digital representation of a file (“hash reference” != “PoW result”) is stored on-chain and the file itself is stored off-chain
- ❑ **Practice**
 - Peer-to-peer network stores and shares hypermedia (e.g., graphics, audio, video, plain text) in a distributed file system



IPFS: Inter-Planetary File System
<https://ipfs.io/>

Wallets

❑ Container for private keys

- Structured files, databases
- Wallets contain keys (as keychains of private/public keypairs) not coins (which are stored on the BC as tx outputs), thus, keys prove the ownership of a coin!

❑ Wallet types

- Non-deterministic (random): just a collection (hard to maintain)
- Deterministic (seeded): keys derived from a common seed (random number+index) via hash-functions
 - Hierarchical deterministic (HD) – BIP0032/BIP0044 standard – with keys in a tree structure: parent $\Rightarrow$ children $\Rightarrow$ grandchildren
 - Structure organizes inputs, branches with categories (e.g., departments)
- Paper: character strings, 2D codes, BIP0038 encrypted print

Smart Contracts (1)

- ❑ A Smart Contract (SC) resides inside transactions
 - Executed and validated on every node
 - In Bitcoin (a blockchain-based cryptocurrency), SCs specify how to withdraw, escrow, refund, or transfer BTC from A to B
- ❑ Smart contracts first mentioned in 1994/96

A smart contract is a **computerized transaction protocol** that executes the terms of a contract. The general objectives of [a] smart contract[’s] design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and **minimize the need for trusted intermediaries**. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs. [N. Szabo]

<http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>

http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

- ❑ Smart contracts alone are not “smart”
 - They need an infrastructure
 - A blockchain is the ideal infrastructure

Smart Contracts (2)

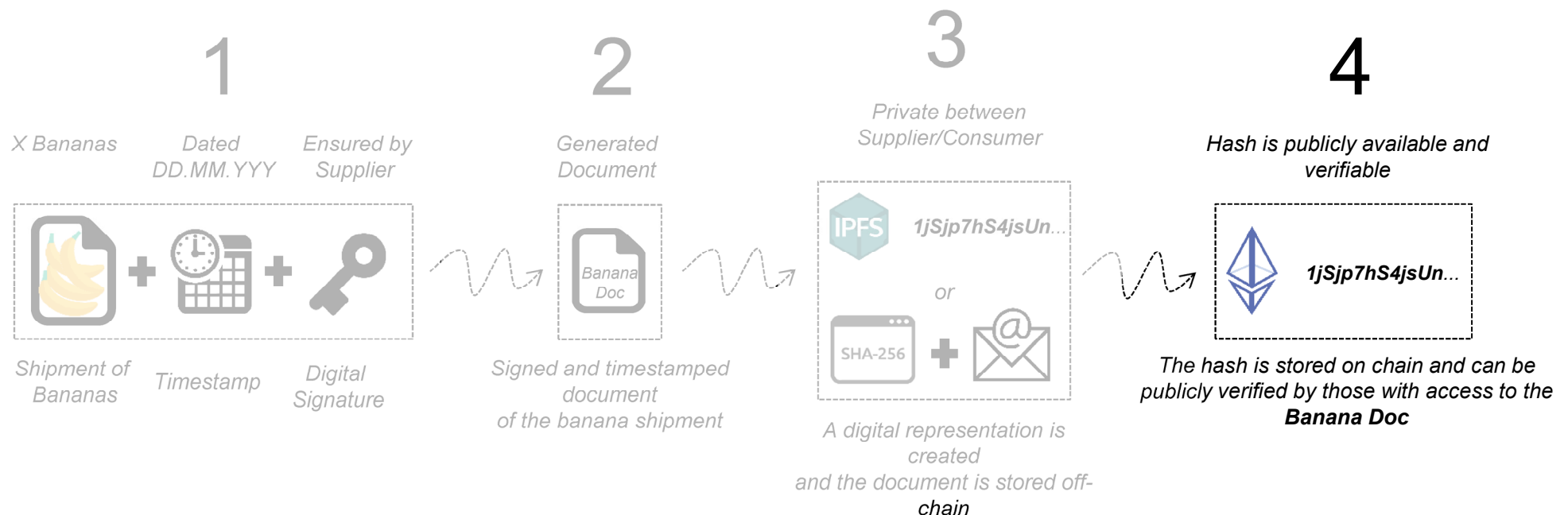
- ❑ SCs can be **exchanged** (not only transactions)
 - This is the code or program to be executed
 - “Programs and APIs” on the BC
 - Thus, SC are programs that encode conditions and outcomes

- ❑ *E.g.*, SCs can define **secure escrow services**
 - Ready to use program working on pre-determined conditions between any two users (*e.g.*, a provider and a customer)
 - “Real-time” operations
 - Near zero marginal cost

- ❑ Note: **Smart Contracts are neither “smart” nor “legal”!**

Blockchain's Immutability Exploited

- Time stamping proves existence, integrity, and creation/exchange of digital assets
 - Keeping track of the creation and modification time



Supplier can use a blockchain-based time stamping solution to ensure the quality of his products

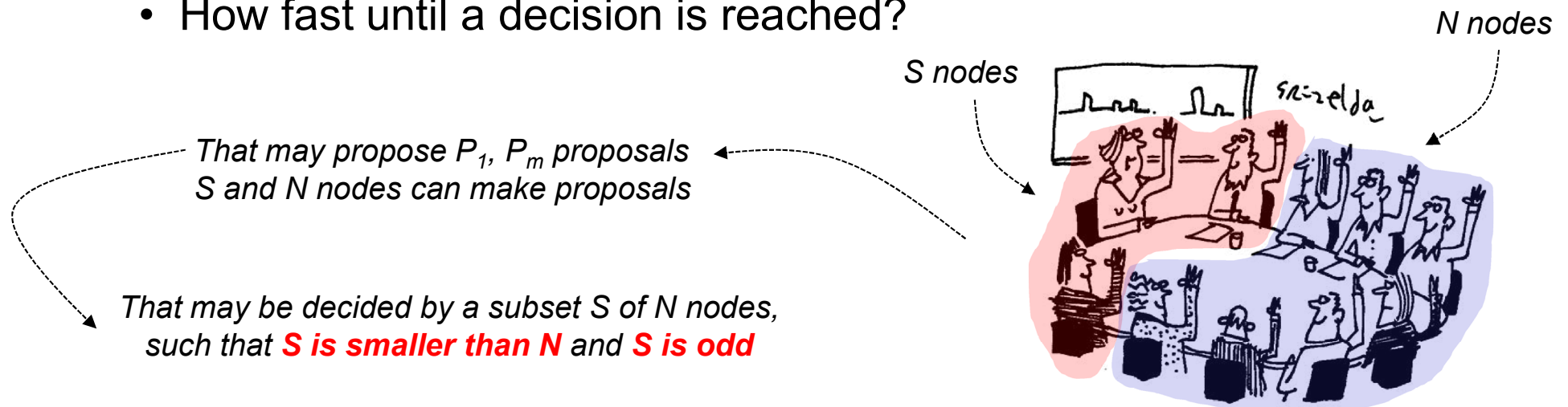
4. Consensus Mechanisms

Mechanisms for Distributed Agreement

- ❑ Also called “Distributed Consensus” algorithms
- ❑ The 4 key characteristics <https://pradeeplogathan.com/blockchain/consensus/>
 - Uniform agreement: No two nodes decide differently
 - Integrity: No node decides twice
 - Validity: If a node decides on value v , then v was proposed by some node
 - Termination: Every node that does not crash eventually decides on some value
- ❑ Given a cluster of N nodes and a set of proposals P_1 to P_m , every non-failing node will eventually decide on a single proposal P_x without the possibility to revoke that decision. All non-failing nodes will decide on the same P_x .

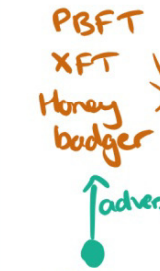
Mechanisms for Distributed Agreement

- ❑ How to decide on a single consensus proposal P_x ?
- ❑ There are many aspects a consensus mechanism has to address:
 - How **many** of the non-failing nodes are **entitled** to vote?
 - How to decide who is entitled to vote?
 - How to ensure a decision in a **deterministic** manner?
 - How fast until a decision is reached?

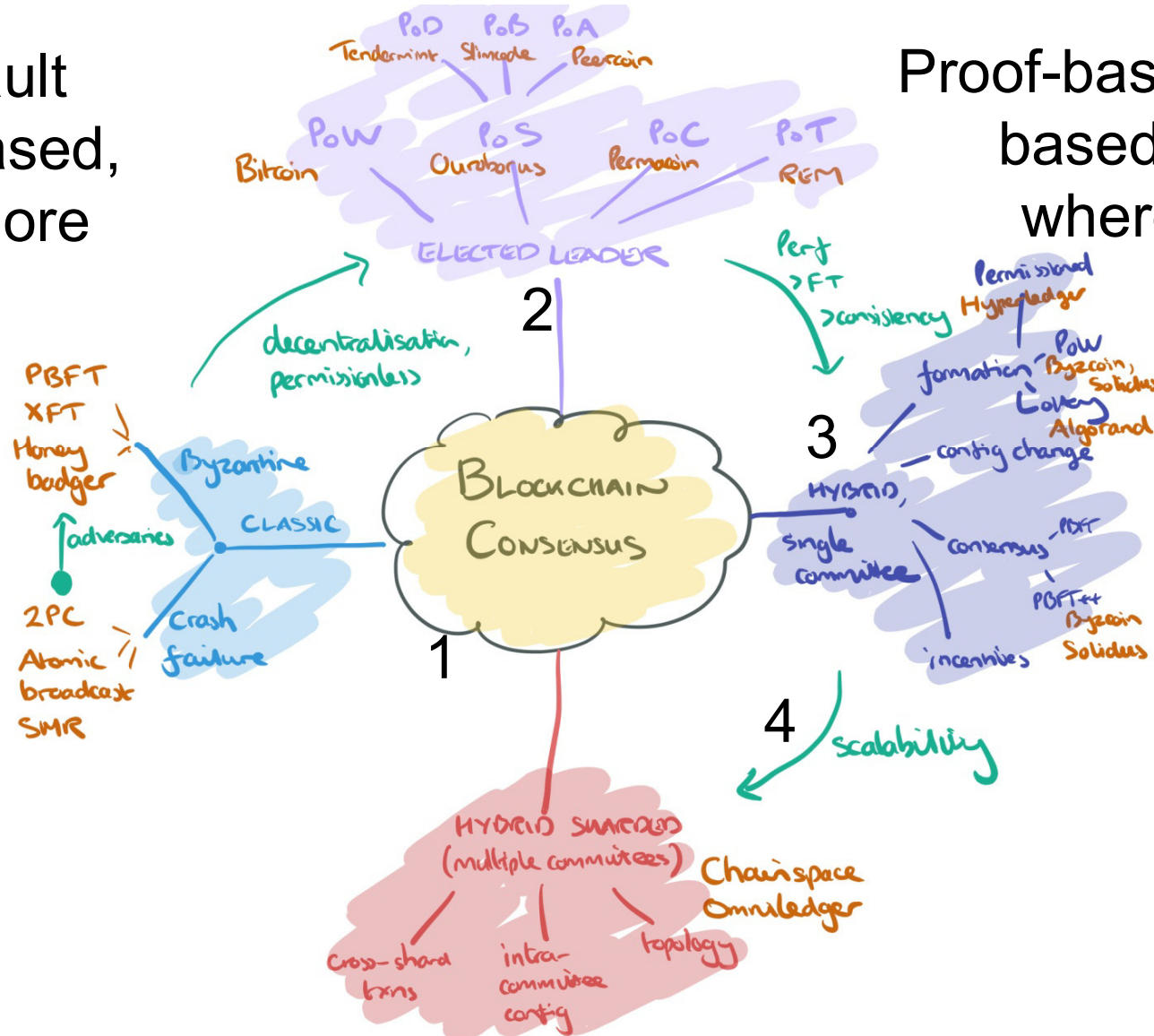


Evolution of Consensus Mechanisms

Byzantine fault tolerance-based, which is a more traditional approach based on rounds of votes.



Proof-based or leader-based consensus, whereby a leader is elected and proposes a final value



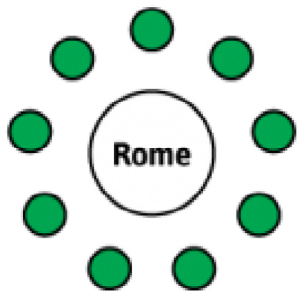
Figure

Consensus Mechanism Types (1)

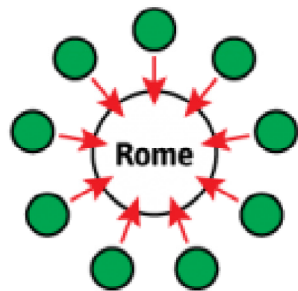
□ Classical Consensus Models

- Crash failure models → honest nodes failing
- Byzantine Failure Tolerance (BFT)
 - State machine replication
- BFT General's Problem

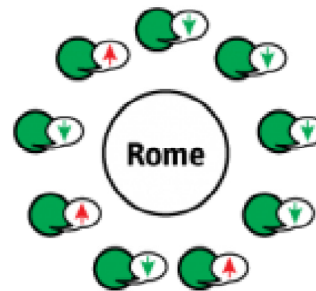
HyperLedger (SOLO, Kafka mechanisms), Stellar



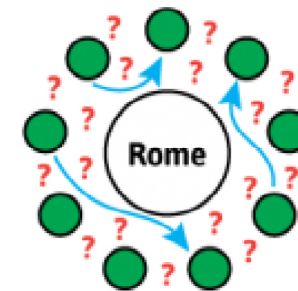
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.



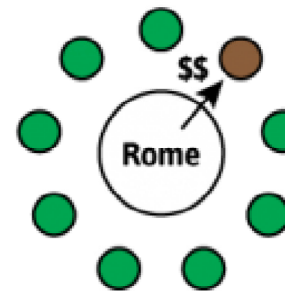
In order to launch a successful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.



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



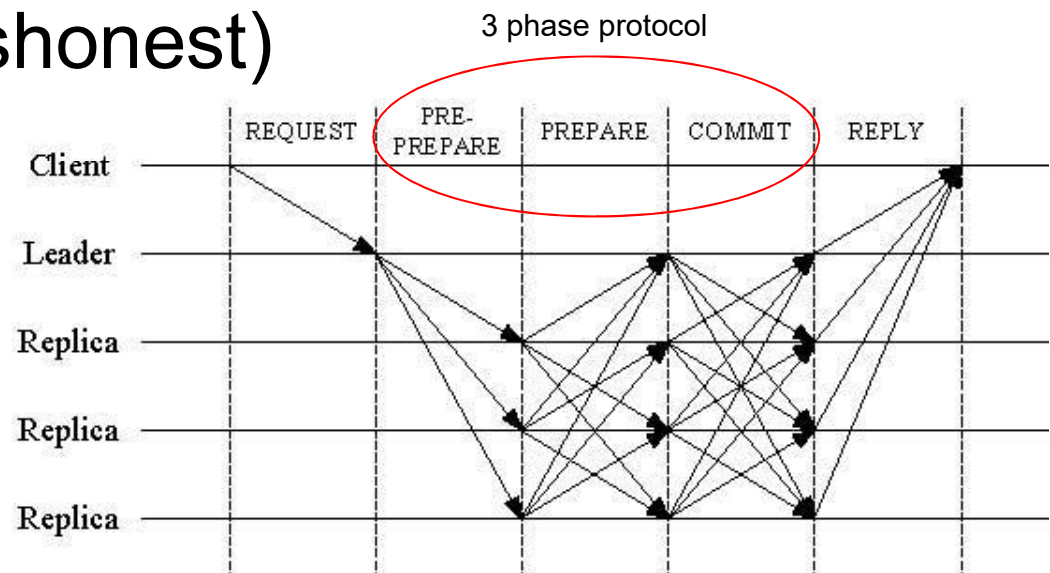
Problem 2
Each of the generals could be bribed by the Roma **Traitorous Generals**.

Byzantine Fault Tolerance (BFT)

- ❑ Described as the capacity of a system to **handle or survive unreliable situations** and (all kinds of) failures
- ❑ **Practical BFT (PBFT)**: assume a small fraction of nodes as Byzantines (dishonest)

1. A **client** sends a request to invoke a service
2. The primary **leader** multicasts the requests to the replicas
3. Replicas execute the request and send a reply to the client
4. The client waits for $F+1$ replies from different replicas with the same result

- ❑ Other examples: XFT, HoneyBadger

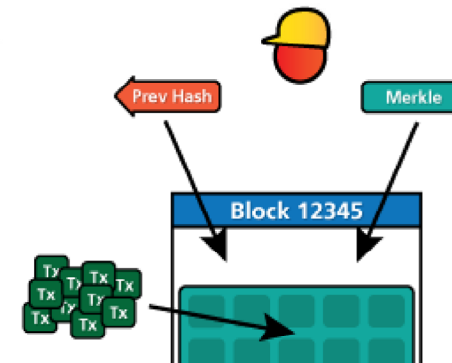
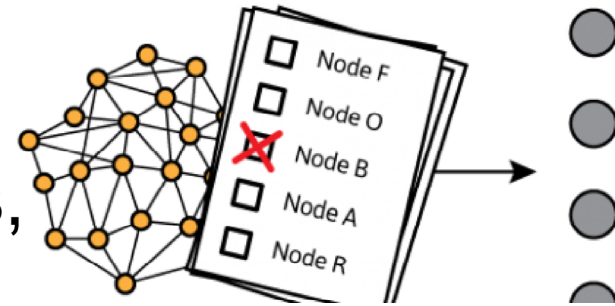


n = Total # of nodes in network

$f = \frac{n-1}{2}$ (Max # of faulty nodes)
PBFT property

delegated BFT (dBFT)

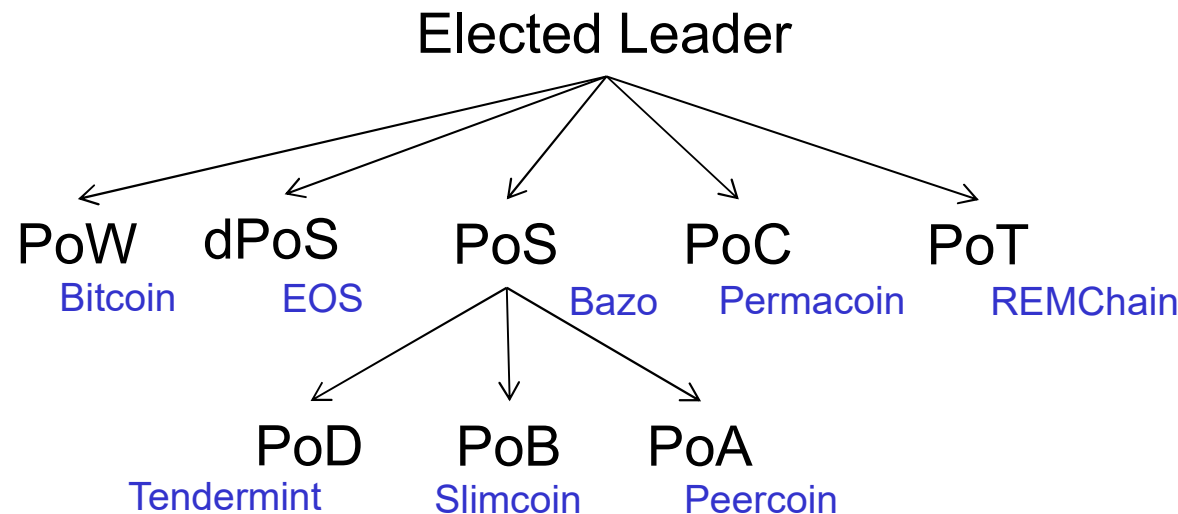
- ❑ Nodes in the network elect a group of consensus nodes (e.g., CoZ)
- ❑ Leader/speaker randomly chosen from consensus nodes, remainders are delegates
- ❑ Leader/speaker creates new block, needs to be positively checked by 2/3 of all delegates
- ❑ If 2/3 agree, block is added to the chain
- ❑ Countermeasures for dishonest leader/speakers or delegates



Consensus Mechanism Types (2)

□ Elected Leader Models

- Probabilistic elected leader in a
 - Lottery-like
 - Competition, or
 - Probabilistic algorithm



PoX: Proof-of-X, where X=

A: Age

B: Burn

C: Capacity (storage)

D: Deposit

S: Stake

T: Trust

W: Work

d: delegated

Proof-of-Work (PoW)

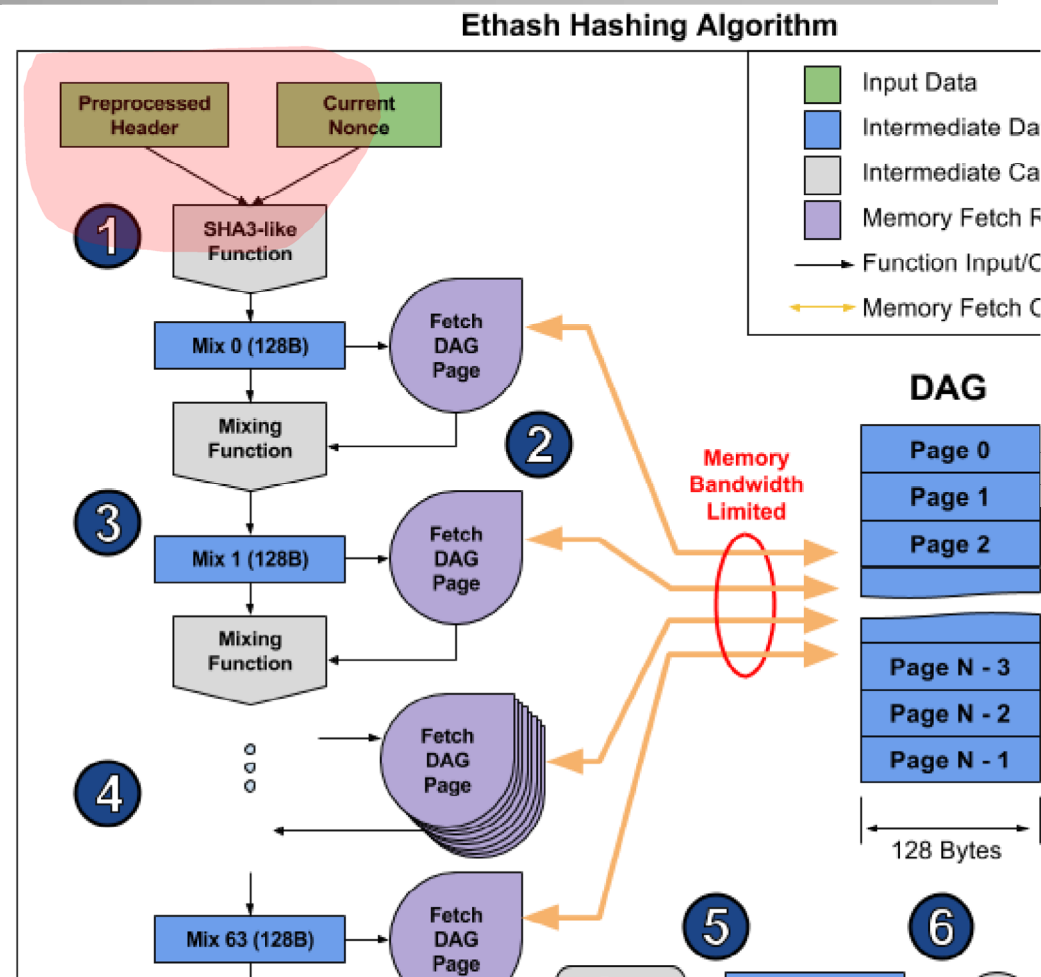
- ❑ Set of transactions becomes available, block is created, by utilizing the following data
 - Transaction(s), hash of previous block
 - Nonce (arbitrary number, can only be used once)
 - Other information (depending on BC)
- ❑ Hash of new block is calculated
- ❑ Checking performed once hash was computed
 - Hash is **above the target** value → Another miner may have found a suitable hash, block attached to local BC, but miner lost the lottery, otherwise nonce will be incremented, retry
 - Hash is **below the target** value → This miner won the lottery and the new block's hash determines the PoW result

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1

The **Pre-processed Header** (derived from the latest block) and the **Current Nonce** (the current guess), are combined using a SHA3-like algorithm to create our initial 128 byte mix, called **Mix 0** here.



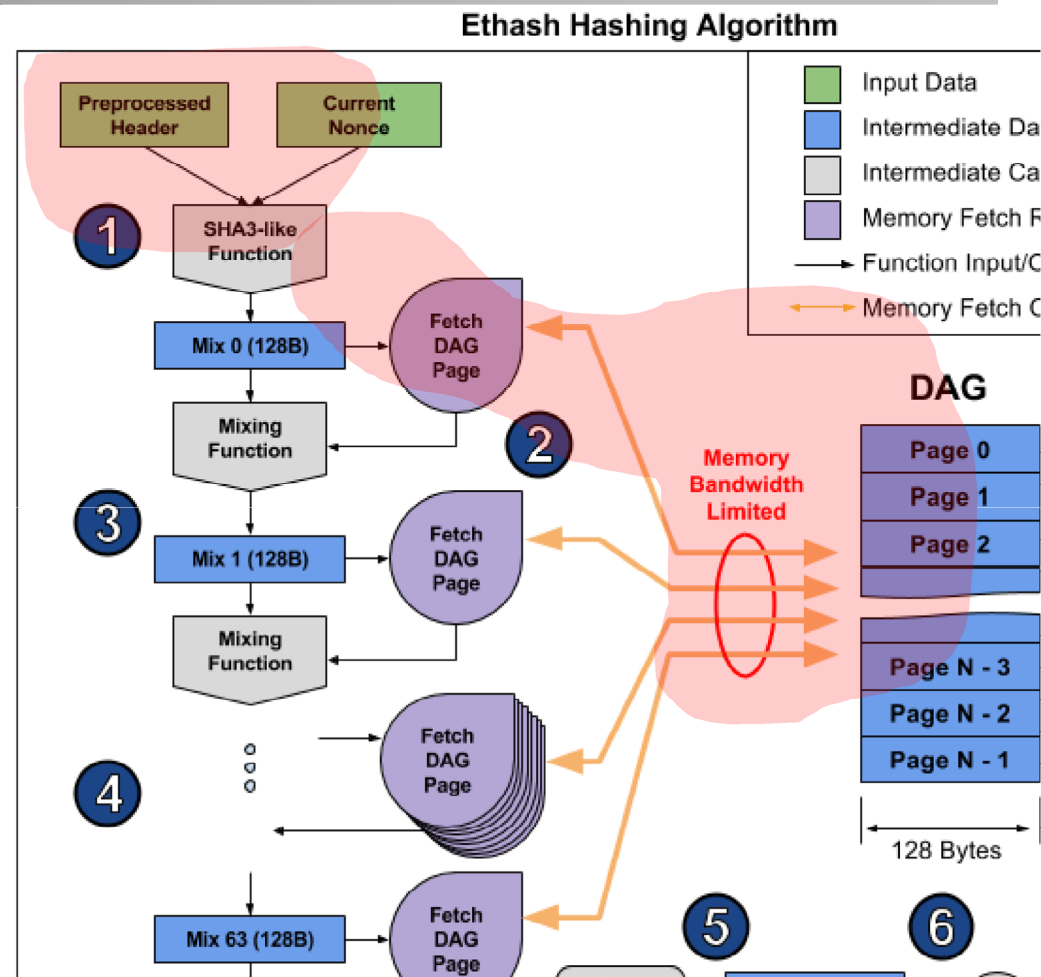
- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1 2

The **Mix** is combined with the retrieved DAG page. This is done using a ethereum-specific mixing function to generate the next mix, called **Mix 1** here.



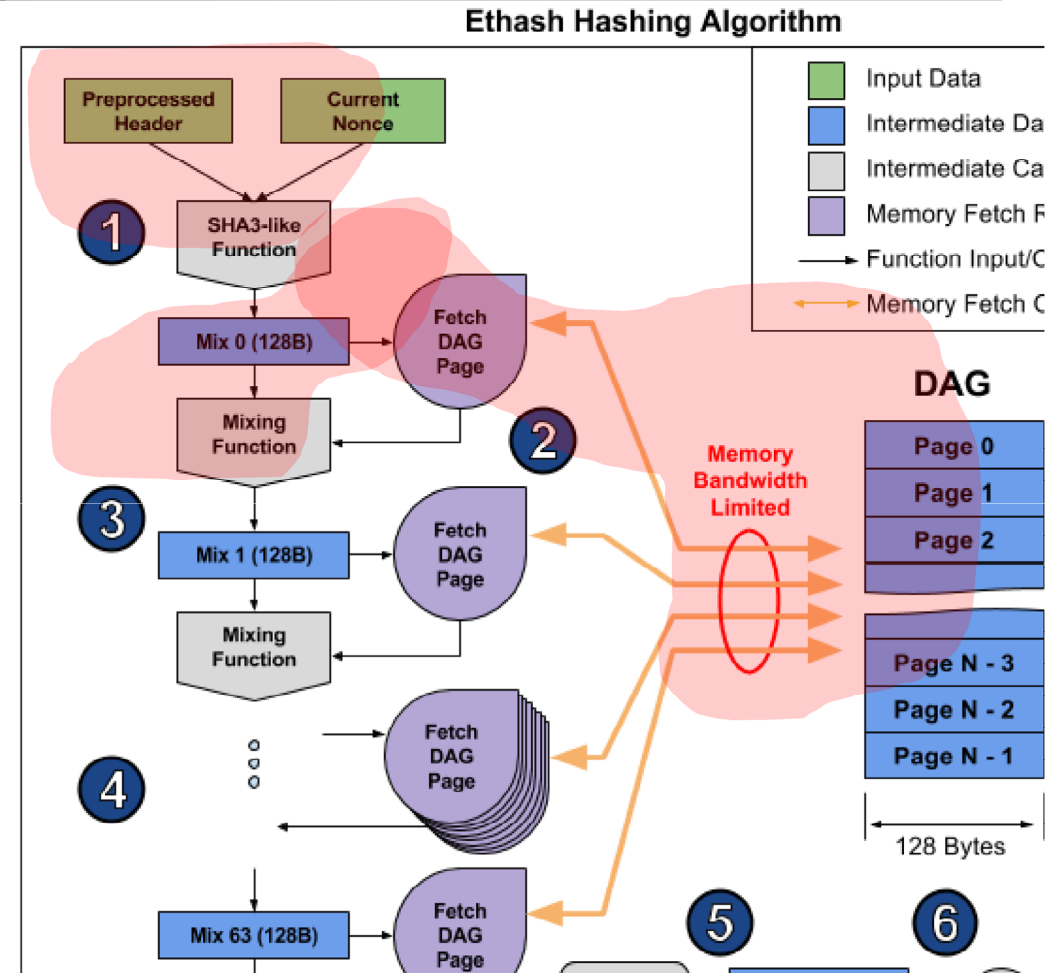
- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only requires large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1 2 3

The **Mix** is used to compute which 128 byte page from the DAG to retrieve, represented by the **Get DAG Page** block.



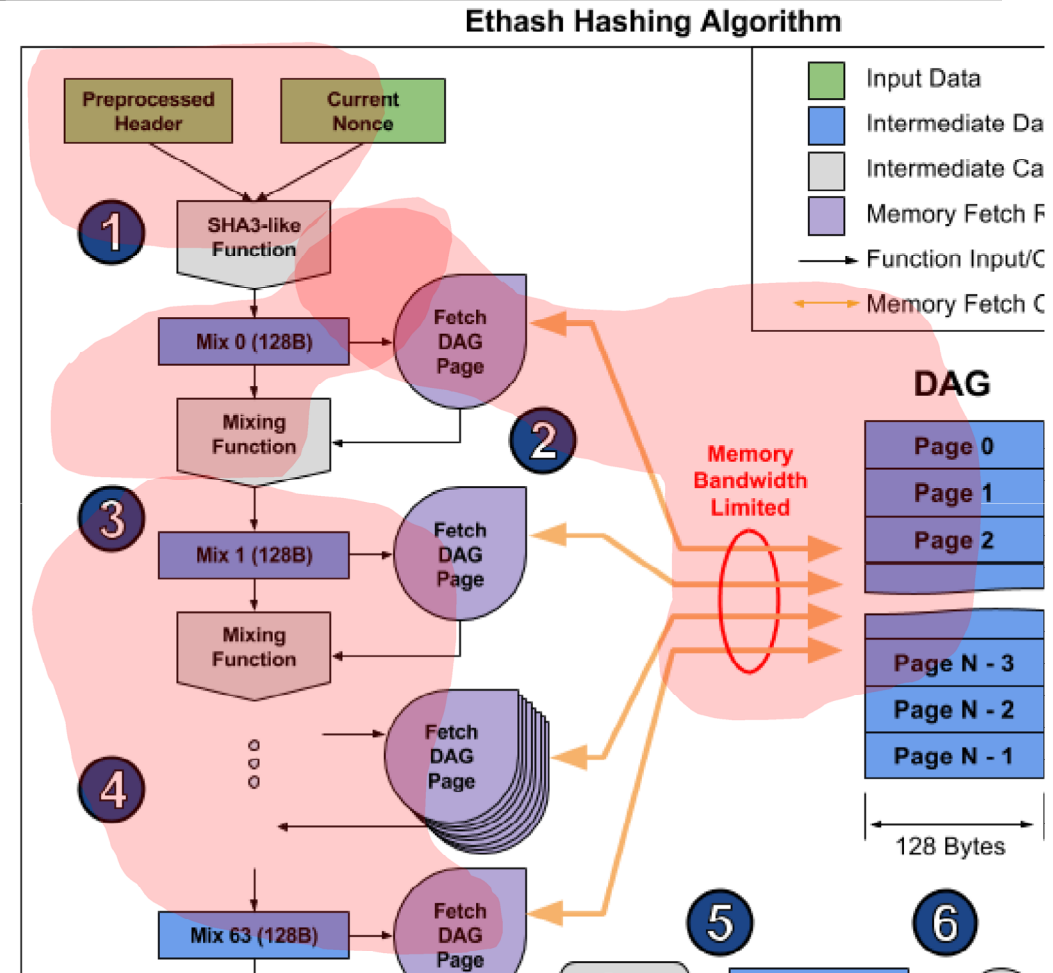
- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only requires large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1 2 3 4

Steps 2 & 3 are repeated 64 times, finally yielding **Mix 64**.



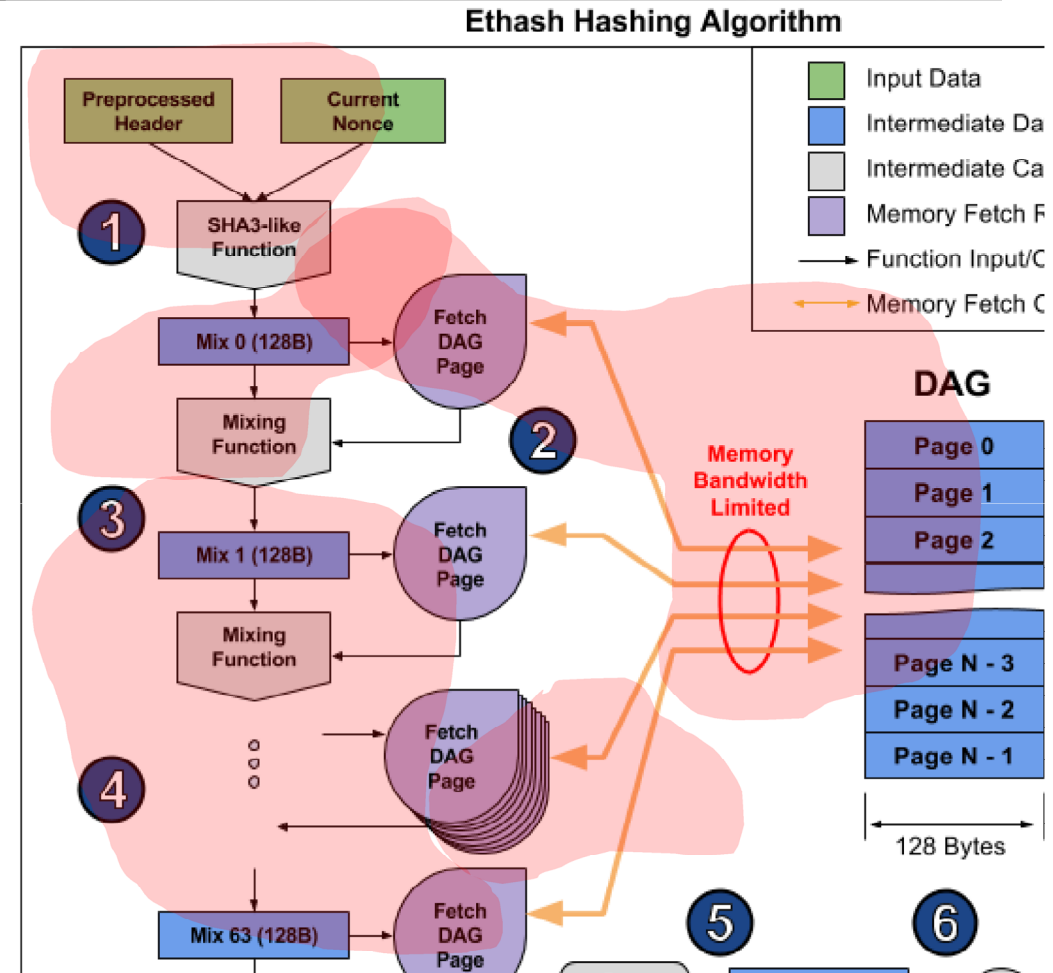
- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only requires large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1 2 3 4 5

Mix 64 is post processed, yielding a shorter, 32 byte **Mix Digest**.



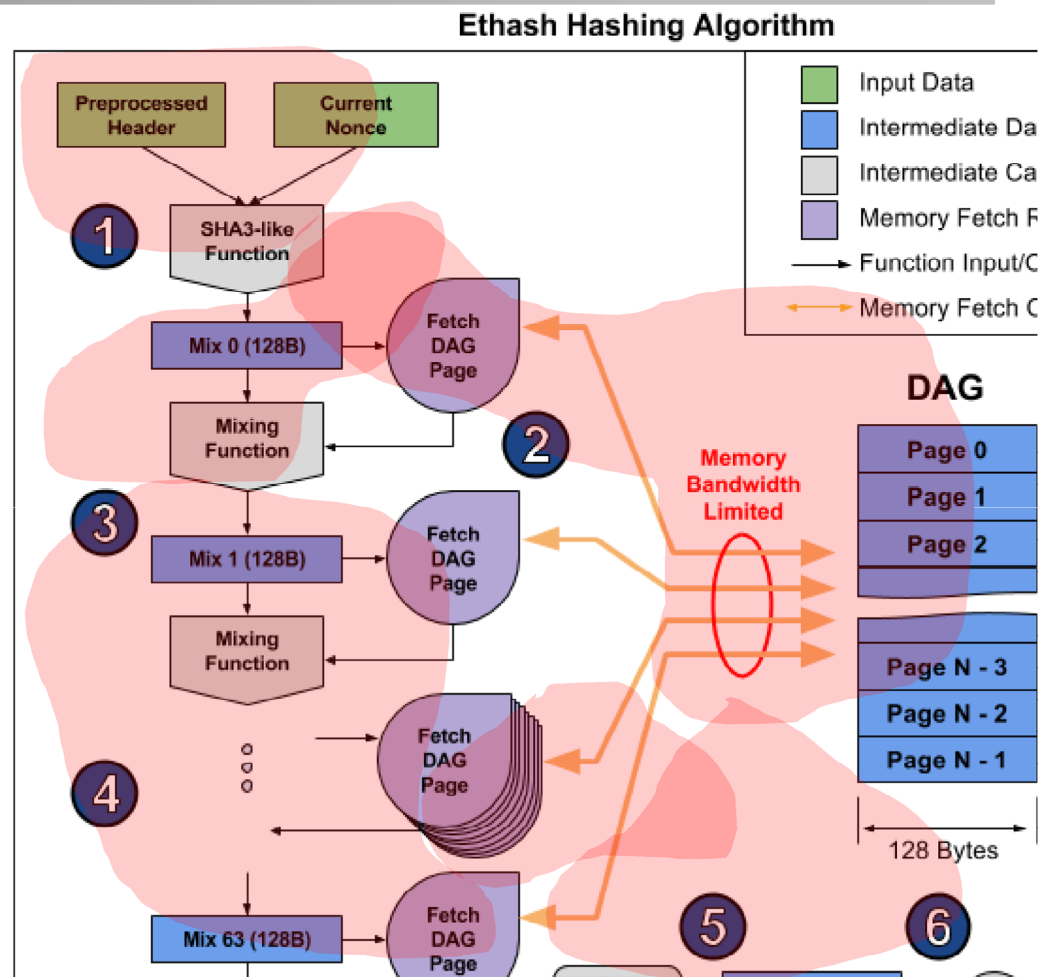
- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- ❑ Based on **partial hash collision** (target hash)
 - Works as a memory-bound cryptographic sponge
 - Requires heavy **fetching of data on the DAG** (RAM intensive process)

1 2 3 4 5 6

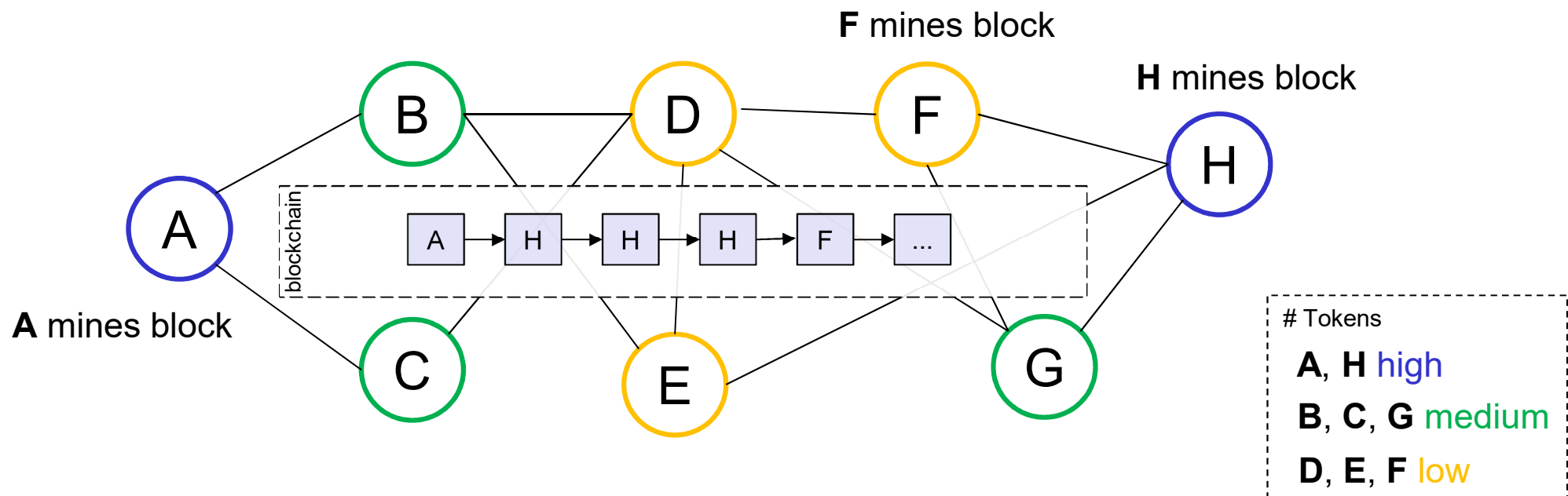
Mix Digest is compared against the predefined 32 byte **Target Threshold**. If **Mix Digest** is less than or equal to **Target Threshold**, then the **Current Nonce** is considered successful, and will be broadcast to the ethereum network. Otherwise, **Current Nonce** is considered invalid, and the algorithm is rerun with a different nonce



- **DAG** (Directed Acyclic Graph) - randomly generated dataset
- Designed to be **ASIC-resistant** as it only require large amounts of memory, which favors smaller miners

Proof-of-Stake – PoS (1)

- ❑ Blocks are “mined” according to the amount of “tokens” he or she holds (**stake**)
 - The higher is the number of tokens (coins) at stake, the higher is the “mining power” (linear relation)
 - Nodes receive block reward as inherent incentive



Proof-of-Stake – PoS (2)

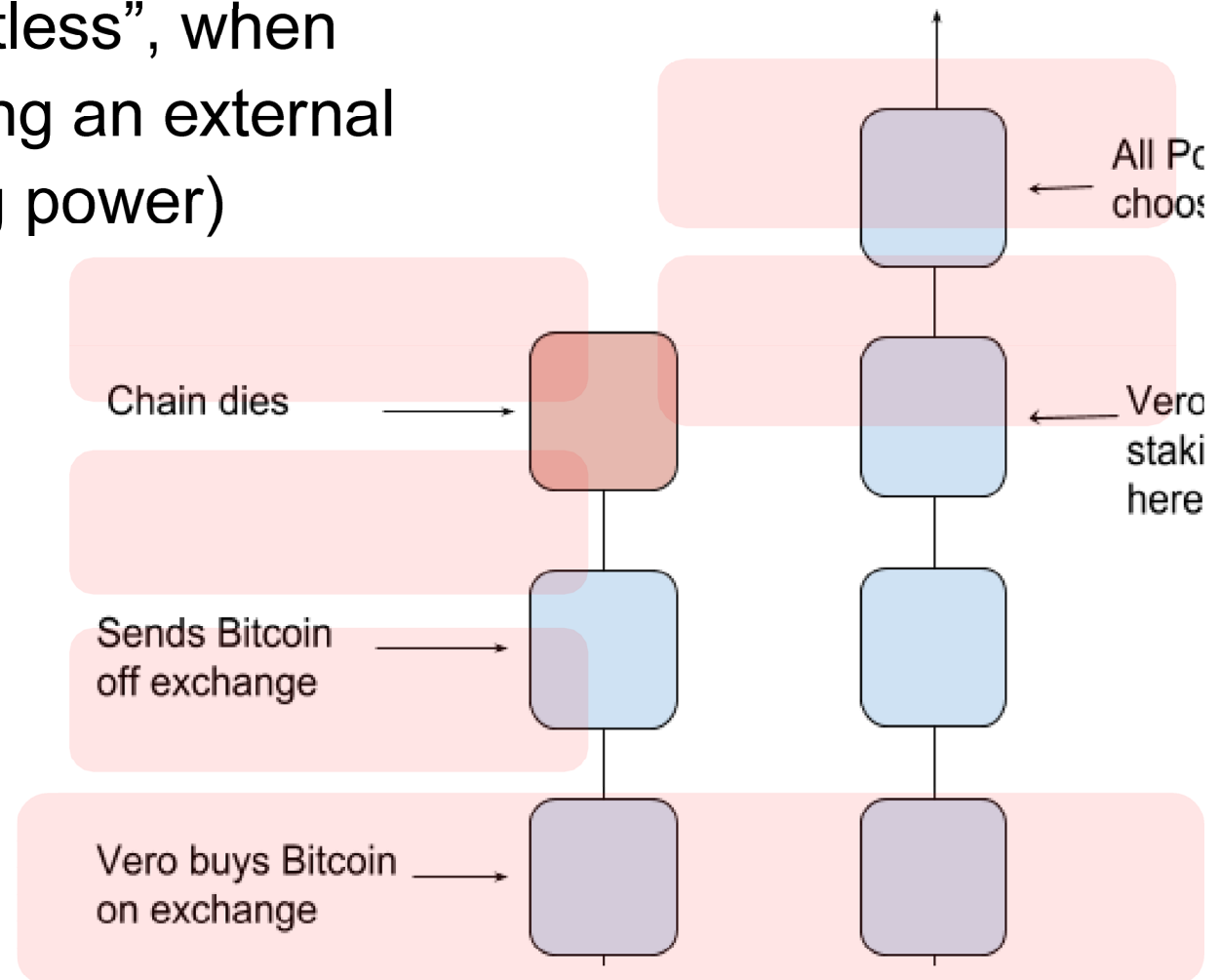
□ Nothing-at-stake problem

- Creating forks is “costless”, when someone is not burning an external resource (e.g., mining power)

- PoS alone is “unworkable” and leads to misuse

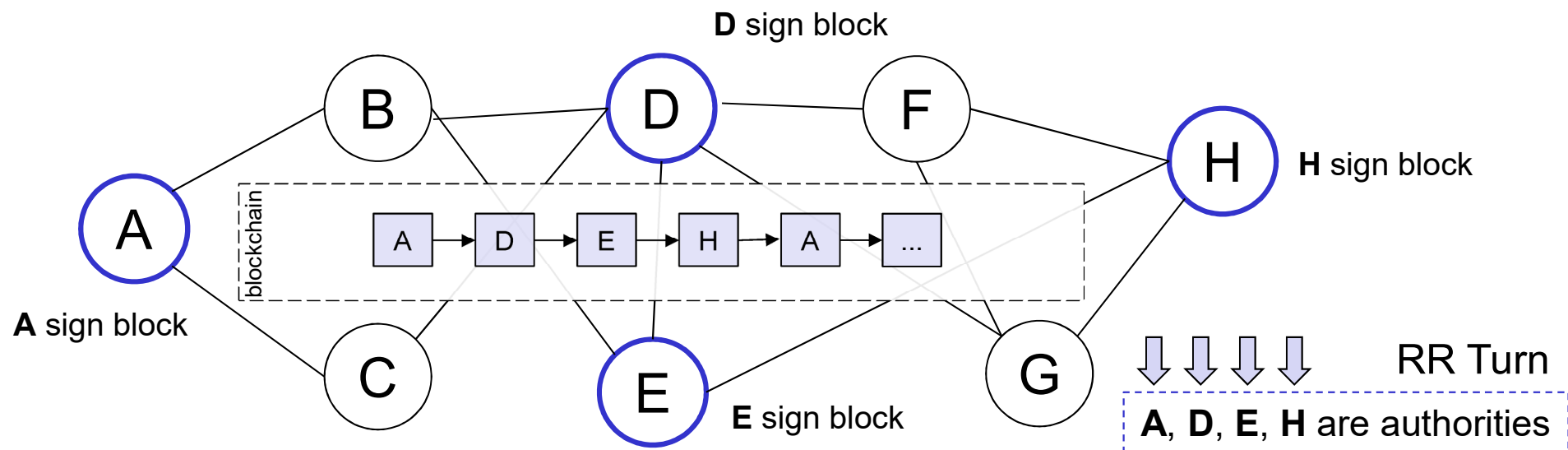
□ PoS variant:

- **DPoS**: nodes can delegate their stakes to a list of nodes



Proof-of-Authority (PoA)

- ❑ PoA is a modified form of PoS, where instead of stake a validator's **identity** performs the role of **stake**
- ❑ Authorities (nodes) are allowed to create new blocks
 - Clique (practical implementation) of PoA
 - Requires $N/2+1$ (more than 50%) of signers to be honest
 - Authorities sign new blocks in a Round-robin (RR) fashion



Alternatives

□ Proof-of-Burn (PoB)

- Mechanisms should proof that participants burned “coins”
 - Sending them to a verifiably unspendable address
 - This is expensive from each individual’s point of view
 - But it consumes no resources other than burned asset
 - Ultimate source of scarcity remains PoW-based cryptocurrency



□ Proof-of-Capacity (PoC)

- Also known as Proof-of-Space



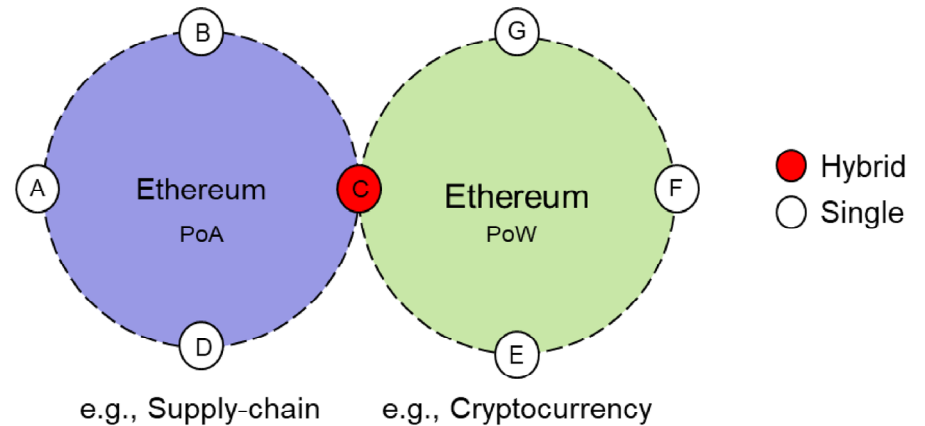
- Miner has to proof a (free) storage capacity to be entitled to create blocks
- Miners has to stake their disk (hard-drive or SSD) capacity to create blocks
 - Similar to PoS in this sense

Hybrid Consensus

❑ Hybrid consensus models

- Using a single consensus results in limitations

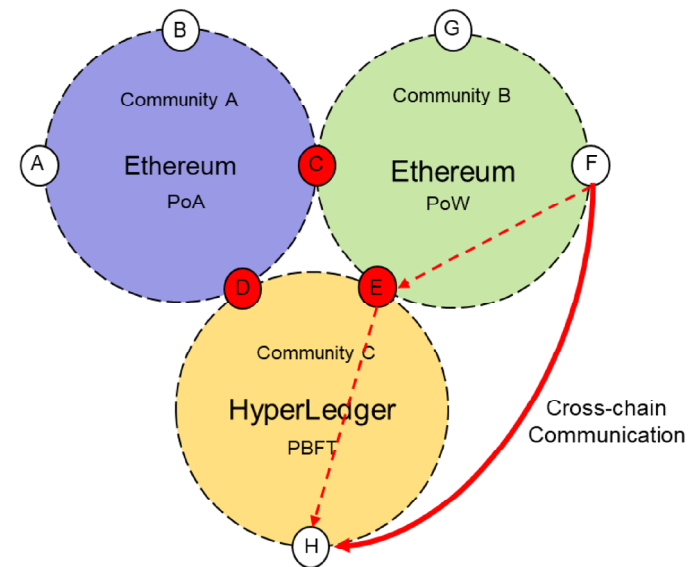
- Combination of different consensus mechanisms



❑ Hybrid sharding

- System can be organized into shards (communities)

- Cross-chain communications
- Applied by CSG's Bazo BC (*cf.* later)



Comparison of Consensus Mechanisms

*Incomplete
view!*

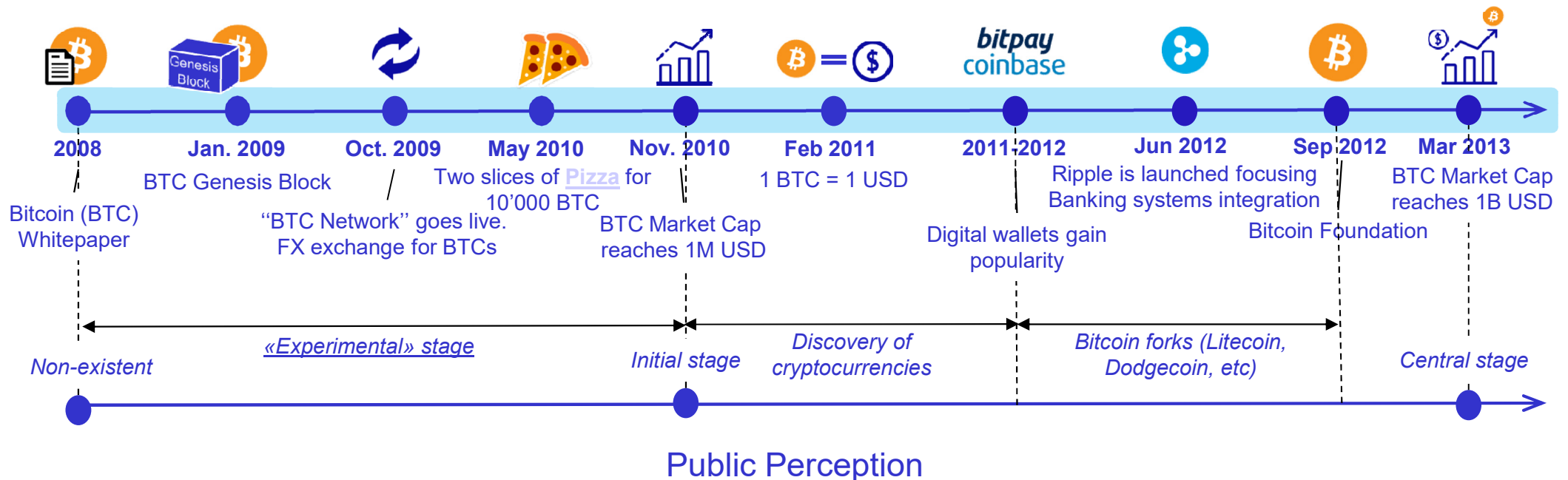
Mechanism	Security Level	Depending on	Scalability	Remarks
BFT	“Reasonable”: Leader pre-elected 51% failure	-	Medium	Trust in pre-election
dBFT	“Reasonable”: set of leaders pre-elected	-	Medium	Trust in set of leaders
PoW	High: 51% attack	Hashes	Controversial	Energy consumption high, needed to ensure high security level (by design)
PoS	Unknown: “Nothing-at-Stake”	PoW-based “stake”	Under discussion	“Costless” forking, thus, measurable assets needed
PoA	Identity-based	PoS, PoW	Under discussion	Authorities required
Shards	Unknown	Any Po“X”	Unknown	Communities, interoperability

5. Blockchain Eras

Blockchain 1.0

❑ Digital Currency or Cryptocurrencies

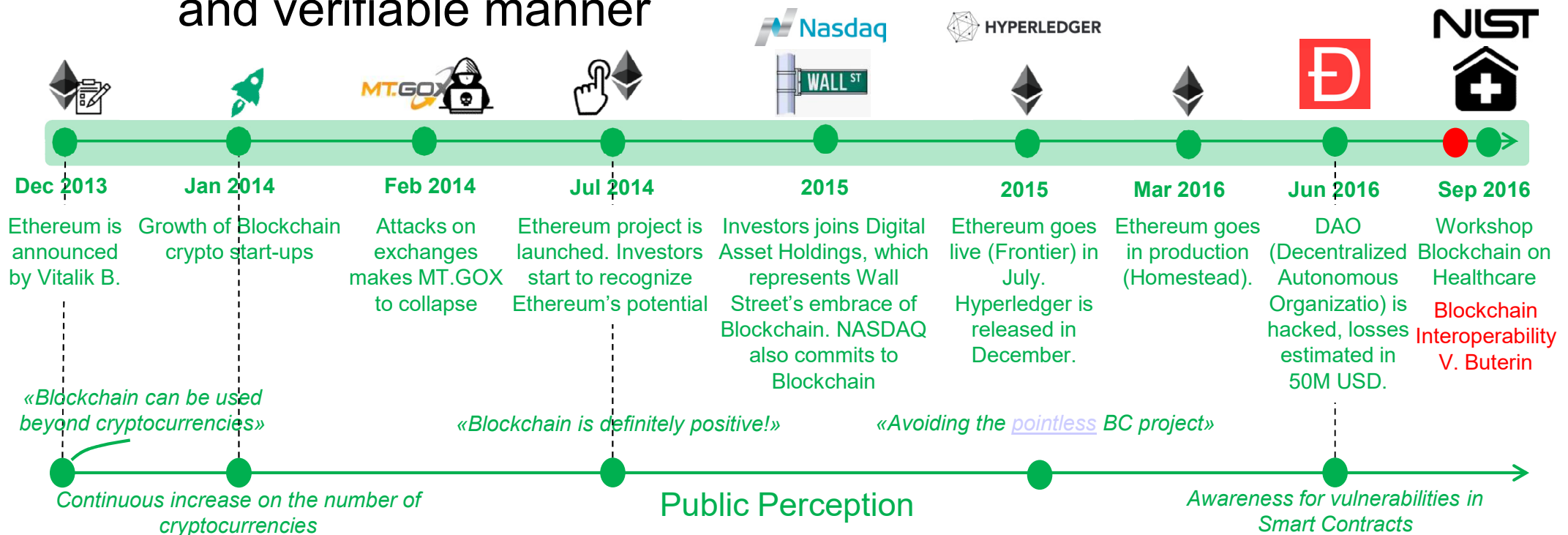
- Decentralized payment system
- Bitcoin as the father of digital currencies
 - Still, not much awareness of (other) blockchain capabilities
- Proof-of-Work (PoW)



Blockchain 2.0

□ Smart Contracts

- Ethereum unlocks the blockchain potential beyond cryptocurrencies
- Blockchain is able to run computer programs in a transparent and verifiable manner



Blockchain 3.0

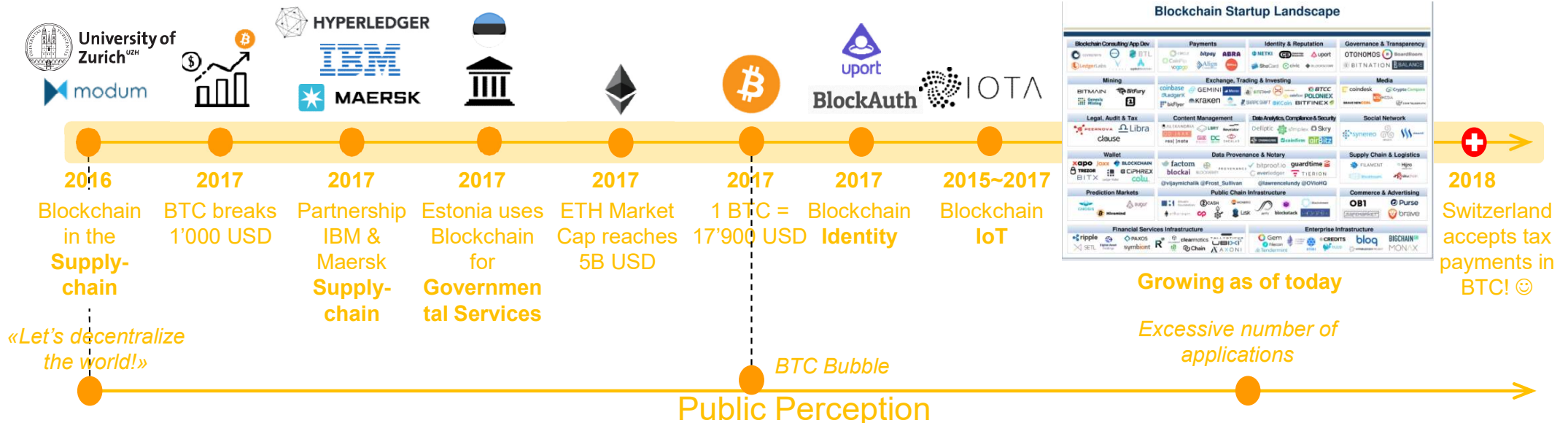
❑ Decentralized Applications (DApps)

– Production stage:

- Large number of applications

– Scalability/Performance issues:

- Need for performance → new consensus protocols
- Need for storage → off-chain storage tools



Blockchain 4.0

❑ Ecosystem and Industry Integration

- Making blockchain effective in industry
- Decentralized and disconnected blockchain networks
 - Vendor-specific blockchain technology, interoperable chains
- Need for standardization

As of today



Applications of Eras and Consensus Mechanisms within CSG Projects

Project	BC Technology	Application Domain	Era	Consensus
<i>IoT-based Pollution Monitoring</i>	Ethereum	Environment	3.0	PoW
<i>Blockchain-based Trading Contracts</i>	Ethereum	Marketplaces	3.0	PoW
<i>Blockchains for Cold-chains (BC4CC)</i>	Multiple	Interoperability	4.0	Multiple
<i>BLW: Foodchain</i>	Ethereum	Food Supply-chain	3.0	PoW
<i>Cryptocurrency from Scratch: BAZO</i>	BAZO	Cryptocurrency	1.0	PoW $\Rightarrow$ PoS
<i>Blockchain Signaling System (BloSS)</i>	Ethereum	Network Security	3.0	PoA
<i>Blockchain eVoting</i>	Dedicated BC	eVote	3.0	PoA
<i>Automated SLA Compensation</i>	Ethereum	Business Support	3.0	PoW

Pollution Monitoring



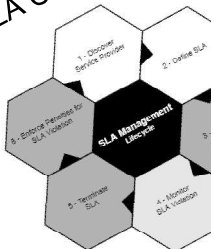
Trading Contracts



BloSS



SLA Compensation



6. Electronic Payment Systems

Electronic Payments (or Systems)

- ❑ Traditional payment approaches
 - Cash, cheque
- ❑ Gap in the past: no transfer of money via the Internet
- ❑ Electronic payment approaches
 - Means of running payments electronically over a network, such as the Internet or dedicated networks (IP-based)
 - Example (Systems): Electronic Fund Transfer (EFT), debit/credit card, ecash, software wallets, smart cards, cryptocurrencies, Financial Electronic Data Interchange), interbank clearing (SWIFT)
- ❑ Micropayment systems
 - Pre-pay, post-pay, pay-as-you-go (once per transaction)

All with “measurable”
centralized elements

Cryptocurrencies (1)

- ❑ A **form of digital cash** with general characteristics
 - Faster, cheaper, reliable than “paper/coin” issued currency
- ❑ Main **two alternatives** for creation of **coins**
 - Issued **by government(s)** – centralized approach
 - Involving Central Bank, banks, brokers, end-users
 - Issued **in a fully distributed manner** – modern cryptocurrency
 - Users transact directly with each other (no involvement of banks)
 - Users store money themselves
- ❑ General **problems of cryptocurrencies**
 - Countermeasures against distributed fraud and manipulation
 - Lack of trust
 - Malicious transactions
 - Double spending

Cryptocurrencies (2)

- ❑ Automated prevention mechanisms against fraud
 - Issuing of currency to be traceable & deterministic (algorithm)
 - All transactions can be recorded in a decentralized manner
 - All transactions can be verified, own and everyone's else transactions
 - Cryptographic mechanisms essential
- ❑ Blockchains, distributed ledgers or public records, provide the key basis for modern cryptocurrencies
 - They do not require a Trusted Third Party (TTP) nor trust as such, neither banks or users → “trustless”
 - Everyone can trace the money being sent, received, verified, and recorded

Blockchains and Cryptocurrencies

- ❑ Cryptocurrencies can either be
 - Mined (CRY-M)
 - Result of mining is the reward, a “coin” (native)
 - *E.g.*, Bitcoin rewards BTC, Ethereum rewards ETH
 - Others define other crypto currencies or assets (non-native)
 - Pre-created/burned (CRY-P)
 - Result of pre-creation distributed by an Initial Coin Offering (ICO)
 - Incentives for mining is collection of transaction fee
 - No cryptocurrency (NRCY)
 - Instead overlay assets are offered
 - Many private blockchains do not use any native cryptocurrency

Cryptocurrency Overview (Never Complete)

~ 2900 cryptocurrencies today:
<https://coinmarketcap.com/>

Approach	Accessibility	Consensus	Crypto Currency
Bitcoin	Public	PoW/ASIC	CRY-M/Bitcoins
Ethereum	Public	PoW/MEM-HARD	CRY-M/Ethers
Ethereum Casper/Serenity	Public	PoS/PoA	CRY-M/Ethers
Litecoin	Public	PoW/MEM-HARD	CRY-M/Litecoin
Monera	Public	PoW/MEM-HARD	CRY-M/XMR
Lisk	Public	PoS/Del	CRY-M/Lisk
R3 Corda	Private	TE	NRCY
Openchain	Private/sidechain	TE	NRCY/various
IOTA	Public	PoW	CRY-P/IOTA tokens
Eris:DB	Private	PoS	CRY-M/Ethers
Chain Core	Private	TE	NRCY/various
Hyper Ledger	Private	TE/PoW/PoS	NRCY
Nxt	Public	PoS	CRY-P/various
Stratis	Private/sidechain	PoW (PoS in future)	CRY-M/STRAT token
Multichain	Private	PoW	NRCY/various
BigchainDB	Private	TE	NRCY/various
Rootstock	Public/sidechain	PoW (Bitcoin)	CRY-M/Bitcoin-Rootcoin
Counterparty	Public	PoW (embedded Bitcoin consensus)/PoB	CRY-P/various
Ripple	Public	TE	CRY-P/Ripple/various
Stellar	Public	TE/PoP	CRY-P/Lumen/various

Cryptocurrency ≠ Blockchain

7. Bitcoins

Bitcoin Characteristics (1)

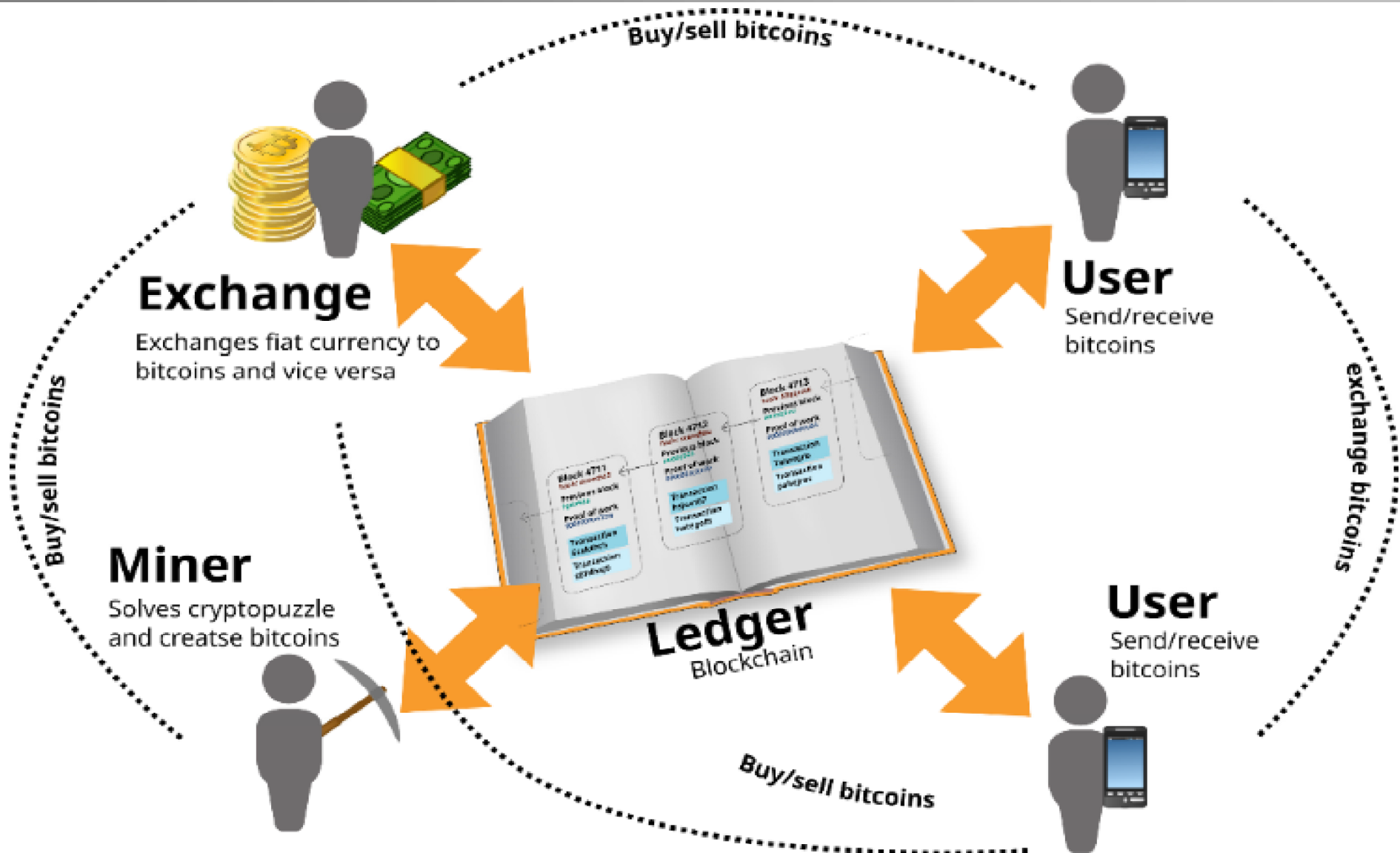
- ❑ Bitcoins are an *experimental* cryptocurrency (digital)
 - Bitcoin is fully peer-2-peer (no central entity)
 - 1st Bitcoin issued on January 3, 2009
 - Smallest unit: 0.00000001 BTC = 1 satoshi
- ❑ Key characteristics
 - Maximum of 21 million BTC
 - Every transaction broadcast to all peers
 - Every peers knows all transactions (~200 GByte as of today)
 - Validation by proof-of-work (partial hash collision SHA-256)
 - Difficult to fake Proof-of-Work (PoW)
 - No double-spending
- ❑ The Bitcoin initiator/inventor is unknown so far



Bitcoin Characteristics (2)

- ❑ Not relying on trust, but on **strong cryptography**
- ❑ Weak anonymity (pseudonymity)
 - All peers know all transactions
 - Clustering: e.g., if a transaction has multiple input addresses, assume those addresses belong to the same wallet
- ❑ Bitcoin **not controlled by a single entity**
 - Only one development community, but not a central bank – Bitcoin Core's SegWit vs. BTU (Bitcoin Unlimited)
- ❑ Creation of Bitcoins performed via the **reward** for validation of payments (partial hash collisions)
- ❑ Bitcoins can be **exchanged for real currencies**
 - Several companies allow to exchange BTC for Dollar, Euro

The Big Picture – Bitcoin (BTC)



Bitcoin Addresses

- ❑ Bitcoin address is an **identifier of 26-35 alphanumeric characters**, beginning with the number 1 , 3, or bc1 that represents a possible destination for a Bitcoin payment
 - Addresses can be generated at no cost by any user of Bitcoin
- ❑ Currently 3 formats
 1. **P2PKH** which begin with the number `1` , eg: `1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2` .
 2. **P2SH** type starting with the number `3` , eg: `3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy` .
 3. **Bech32** type starting with `bc1` , eg: `bc1qar0srrr7xfkvy5l643ltdnw9re59gtzzwf5mdq` .
- ❑ Bitcoin **transfer**: Bitcoin sent to a person by sending bitcoins to one of their addresses
 - Created on demand (offline, too), may be used multiple times
 - Multi-signature addresses: combination of multiple private keys

Bitcoin Mechanisms



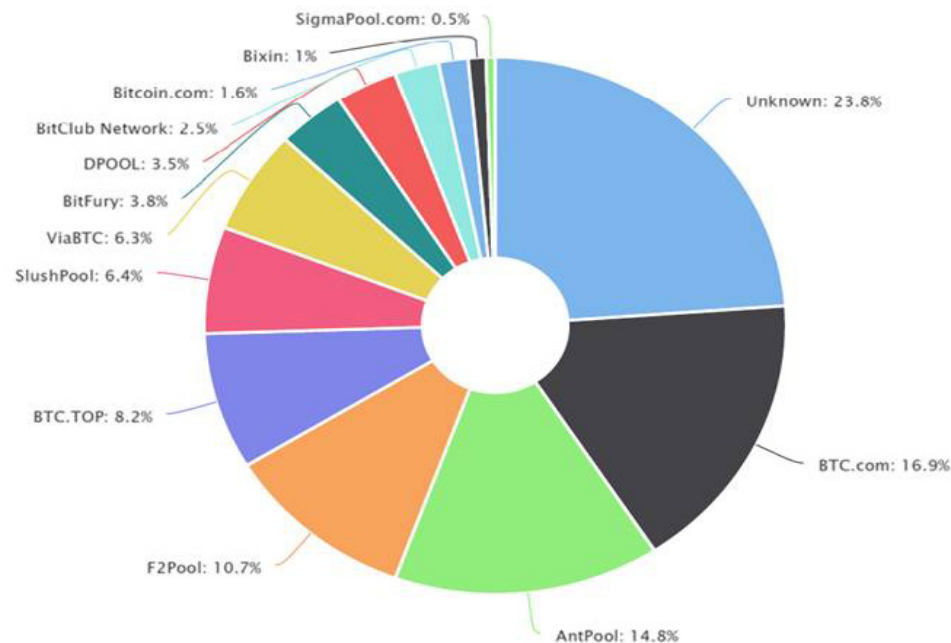
- ❑ UTXO (Unspent Transaction Output)
 - Represents a chain of ownership as a chain of digital signatures where the owner signs a message transferring ownership of his UTXO to the receiver's public key
- ❑ A **wallet** holds public-private keys (wallet.dat)
 - Public key, ECDSA 256 bit → Bitcoin address (receives BTC)
 - Private key used for signing transactions
- ❑ Transaction
 - Peer A to send BTC to peer B → Creates transaction message
 - Transaction contains input/output (BTC from/to)
 - Peer A broadcasts transaction to all peers in the network
 - Transaction stored in blocks → Block is created, verified ~1 min

ECDSA: Elliptic Curve Digital Signature Algorithm

Bitcoin Mining (1)

❑ Couple of big miners

- Miners specialized, AMD GPUs, FPGA, ASIC (Application-specific Integrated Circuit) [1][2][3]



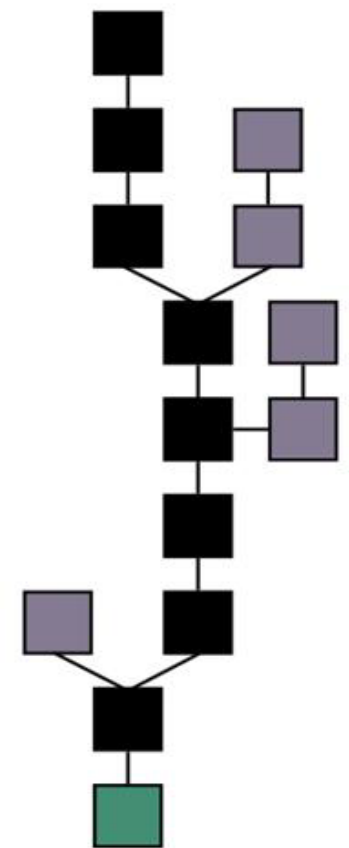
<http://blockchain.info/pools>

- ❑ Mining = creating valid blocks
- ❑ Blocks are linked to previous blocks

- Longest block survive (most difficult)

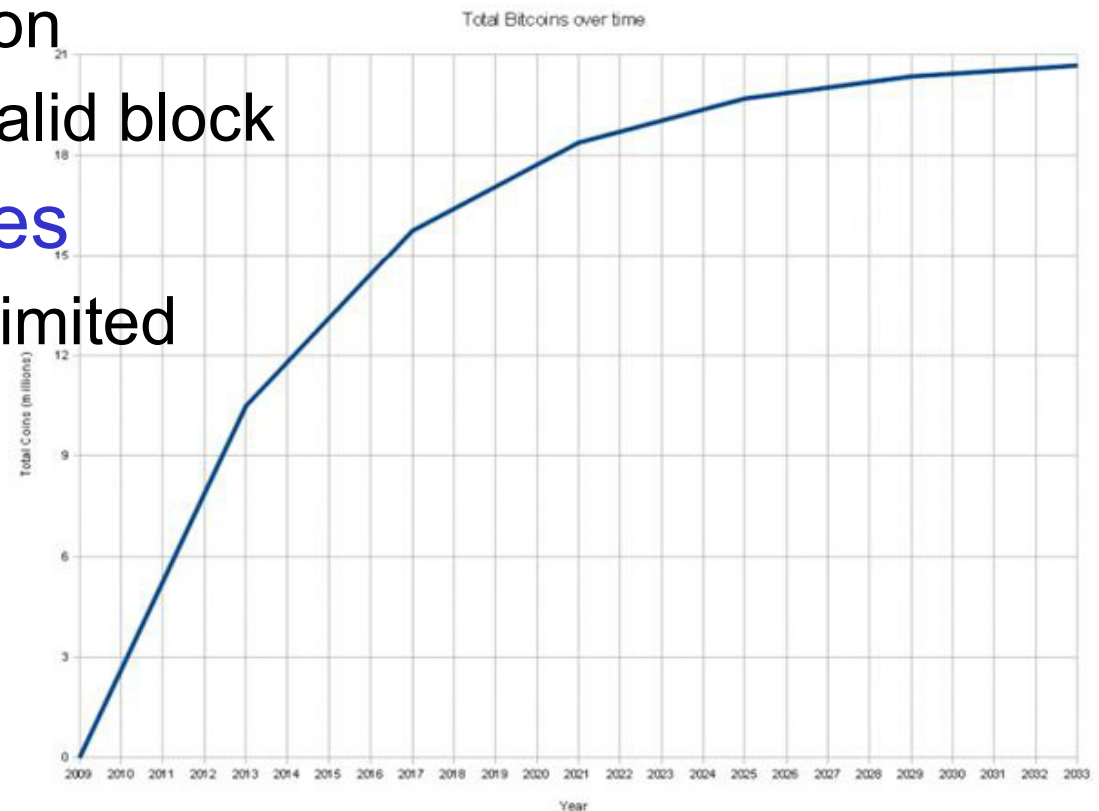
- ❑ Different level of confirmations

- 6 block conf. is considered secure



Bitcoin Mining (2)

- ❑ Mining reward
 - Halves every 210.000 blocks → max. 21 million BTC
 - 12.5 BTC per block creation
 - Competition for creating valid block
- ❑ Miner receives also tx fees
 - Required, since BTC are limited
- ❑ Dangerous, if someone has more than 51% computing power
 - Can exclude and modify ordering of transactions



8. Ethereum

Ethereum Characteristics

- ❑ Generalized blockchain, contracts Turing-complete
 - Solidity with loops, arithmetical operations
 - Every instruction needs to be paid for in “gas”
 - Running out of gas, leads to state reversion, ETH gone
 - Miner decides which transaction at which gas price to include
 - Market for TX
- ❑ White paper released in December 2013
- ❑ Protocols designed from scratch
 - Ethereum foundation located in Zug, non-profit
 - Initiator known: Vitalik Buterin
- ❑ Mining reward at
 - Every block creation, at ~14 s

Ethereum Smart Contracts (1)

- ❑ Computation on EVM is “very expensive”: every contract is run on every full Ethereum node
 - Result on every node is the same
 - “Global computer”, always running, always correct
- ❑ For now, Proof-of-Work (PoW)
 - But difficult with ASIC (memory-hard), starts at 1 GB RAM for full node and miner (2GB RAM should be enough)
- ❑ Account-based
 - External accounts: controlled by private keys
 - Contract accounts never executed on their own
 - Controlled by code
 - All action fired from externally controlled accounts

Ethereum Smart Contracts (3)

- ❑ ERC20 technical standard for Smart Contracts on Ethereum blockchain for implementing tokens
 - Proposed: November 19, 2015
 - January 2018: more than 21,000 ERC20 token contracts
 - EOS, Tronix, VeChain, OmiseGO, MOD Token

- ❑ ERC20 Token (Simplified)
 - No allowance/approval/transferFrom
 - Also no Approval event
 - Using SafeMath (always use it)
 - Creator gets 1,000 coins

Ethereum Releases

- ❑ Olympic (past) – released 9.5.2015
 - Last Ethereum Proof-of-Concept series
- ❑ Frontier (past) - released 30.7.2015
 - Main public network, “Beta”/use at your own risk
- ❑ Homestead (past) - released 3.14.2016
 - Public network considered “stable”, critical protocol changes
- ❑ Metropolis (Byzantium, past) - released 16.10.2017
 - Initial support for ZKP (private smart contracts), reward 3 ETH
- ❑ Metropolis (Constantinople, now) – released 27.2.2019
 - Bit-shifting, reward 2 ETH, other optimizations
- ❑ Serenity (future)

9. Ethereum Wallet – Practical Part I

Exercise Goals

1. Differentiate between blockchain nodes
2. Create an address and send a transaction
3. Send funds and verify account balance
4. Verify a transaction sent using a blockchain explorer



Set-up Overview

- ❑ Focus on **Ethereum blockchain**
 - Ethereum Basics (cf. above) and Geth Commands here
 - Connecting to your Node
 - Connecting to CAS Blockchain

- ❑ **Connect to the CAS Wi-Fi**
 - Wi-Fi: **cas2019**
 - Password: **nopassword**



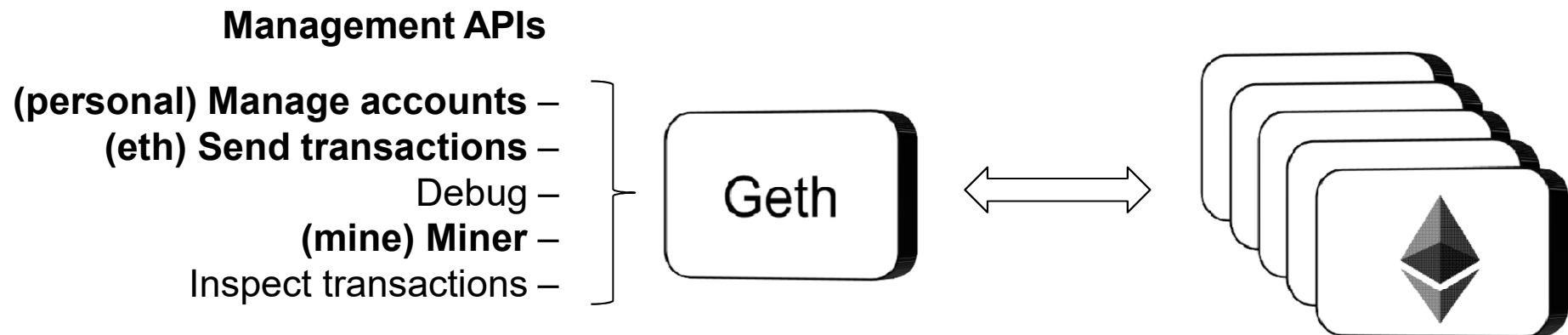
Ethereum Basics – Recalled

- ❑ Ethereum is a **general purpose** blockchain
 - Support the execution of complex Smart Contracts (SC)
 - Empowered the blockchain beyond financial applications

- ❑ An Ethereum Virtual Machine (EVM) processes and executes smart contracts.
 - Abstraction layer: the EVM connects requests (transactions) to and from the network
 - SC operations requires Gas
 - Mathematical operations, write data
 - [Table](#) of EVM OP costs

Geth Client

- ❑ Go implementation of the Ethereum Protocol
 - CLI (Command Line Interface) client
 - Entry point to the main, test and private networks



Management APIs

❑ Personal API:

- `personal.newAccount()` // create a new acc.

❑ Eth API:

- `eth.accounts[0]` //returns the first created acc.
- `eth.sendTransaction({from:acc, to:acc, value:ethers})` //send transaction to acc.

❑ Miner API:

- `miner.setEtherBase(account)` // define which account to receive ethers

Geth Commands

```
$ geth account <command> [options...] [arguments...]
```

Description	Command
Print summary of existing accounts	<i>\$ list</i>
Create a new account	<i>\$ new</i>
Update an existing account	<i>\$ update</i>
Import a private key into a new account	<i>\$ import</i>
Info about subcommands	<i>\$ [command] --help</i>

Geth Options (1)

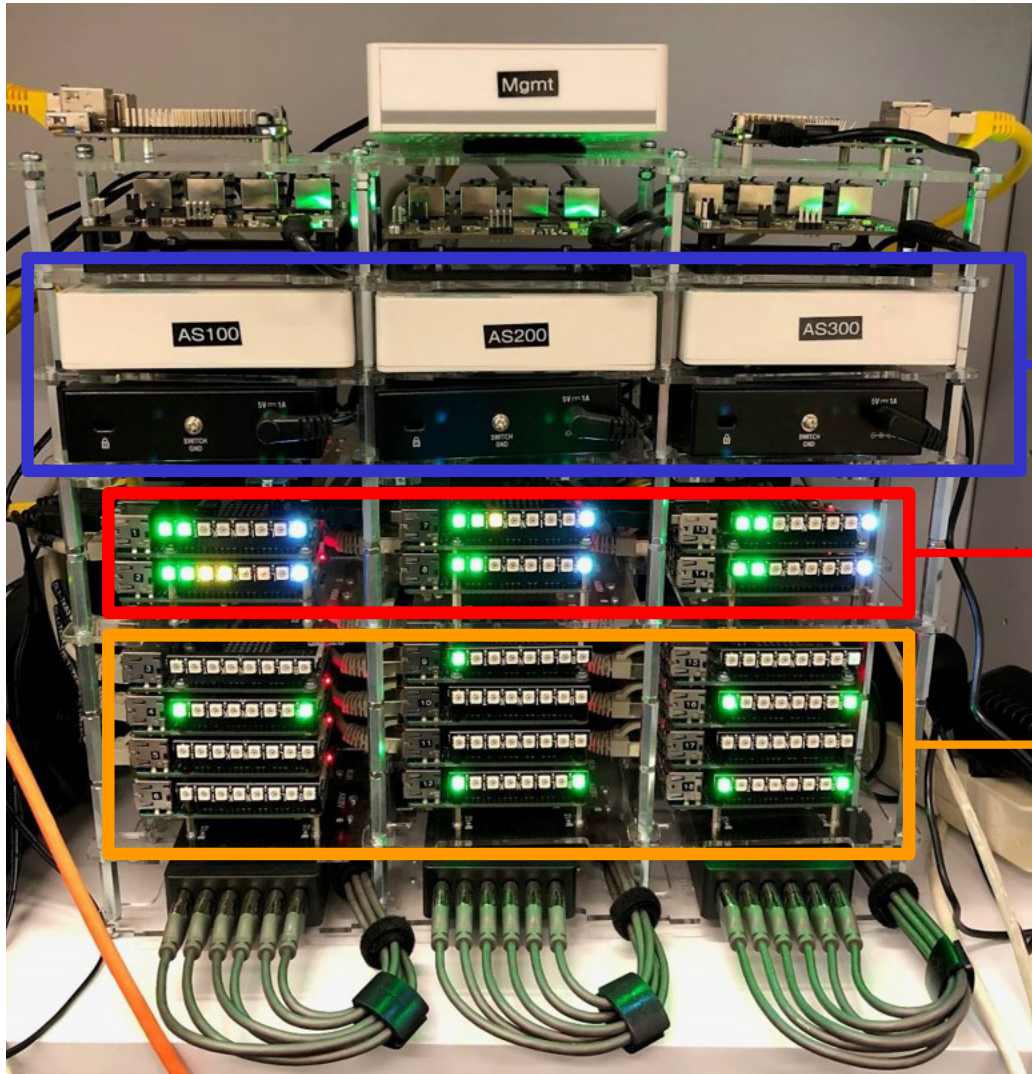
```
$ geth [options] command [command options] [arguments...]
```

Description	Command
Data directory for the databases and key-store	<i>\$ --datadir "path"</i>
Directory for the keystore (default = inside the datadir)	<i>\$ --keystore</i>
Network identifier (integer, 1=Frontier, 2=Morden (disused), 3=Ropsten, 4=Rinkeby) (default: 1)	<i>\$ --networkid value</i>
Ropsten network: pre-configured proof-of-work test network	<i>\$ --testnet</i>
Rinkeby network: pre-configured proof-of-authority test network	<i>\$ --rinkeby</i>

Geth Options (2)

blockchain sync mode ("fast", "full", or "light")	<i>\$ --syncmode "fast"</i>
Enable the HTTP-RPC server	<i>\$ --rpc</i>
Change, respectively, the address and port of the HTTP server endpoint	<i>\$ --rpcaddr value</i> <i>\$ --rpcport value</i>
Ropsten network: pre-configured proof-of-work test network	<i>\$ --testnet</i>
Rinkeby network: pre-configured proof-of-authority test network	<i>\$ --rinkeby</i>
blockchain sync mode ("fast", "full", or "light")	<i>\$ --syncmode "fast"</i>

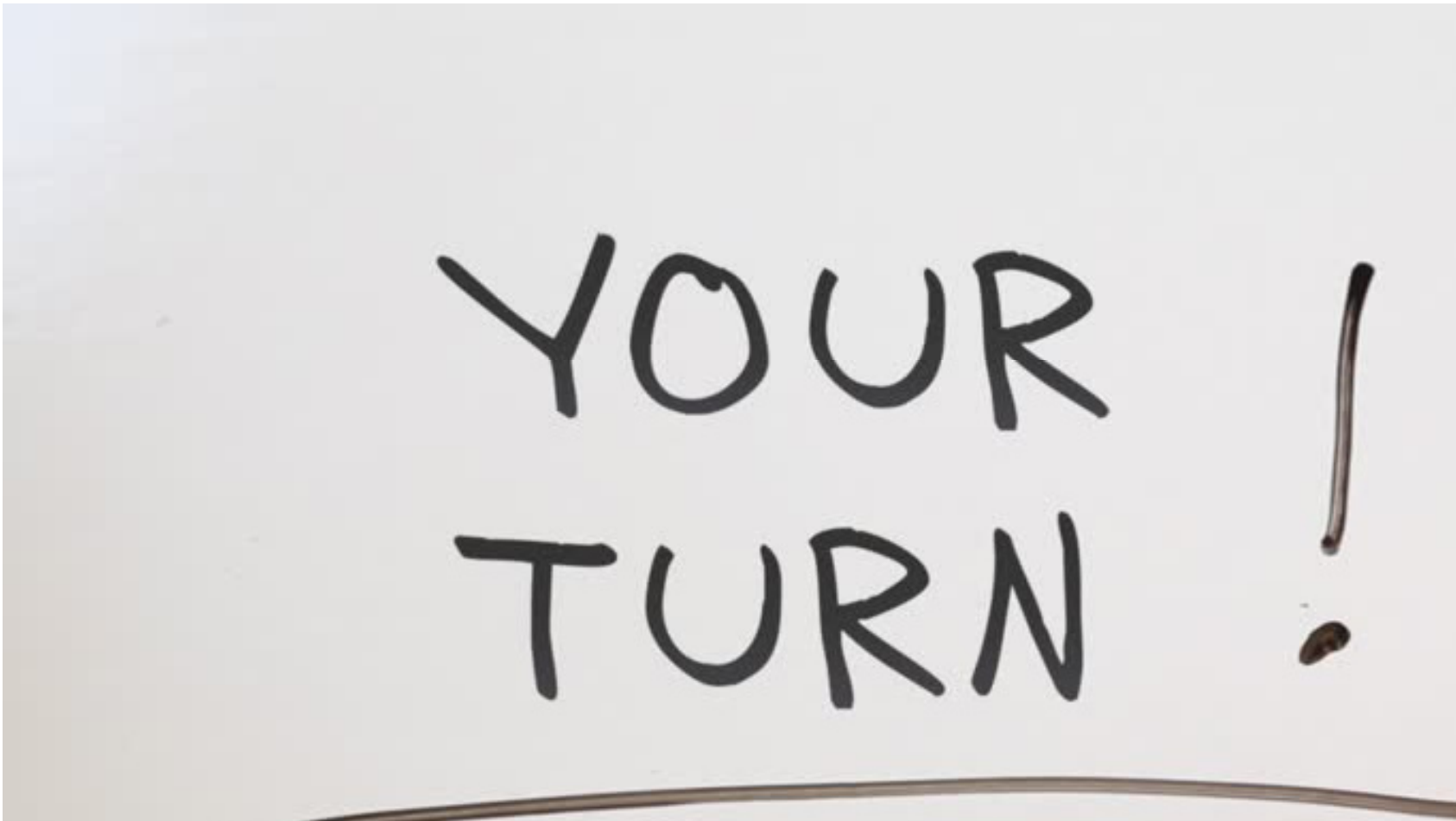
Blockchain Testbed Overview



Network Components

Blockchain Miners Nodes

Blockchain Client Nodes
(You will be accessing these)



YOUR
TURN !

Connecting to Your Node

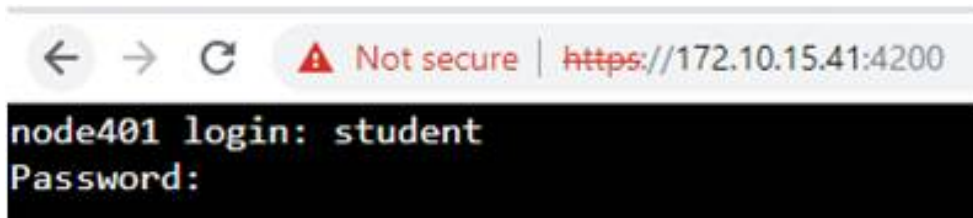
- ❑ Nodes host a **web-based** shell:

XX = Written in provided card

- <https://172.10.15.XX:4200>

- ❑ User: **student**

- ❑ Password: **123456**



Your connection is not private

Attackers might be trying to steal your information from **172.10.15.41** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)

Hide advanced

Back to safety

This server could not prove that it is **172.10.15.41**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to 172.10.15.41 \(unsafe\)](#)

Blockchain Initialization

Step 1: Type: **sh init-blockchain.sh**

```
student@node401:~$ sh init-blockchain.sh
rm: cannot remove 'history': No such file or directory
INFO [05-04|15:29:12] Maximum peer count           ETH=25 LES=0 total=25
INFO [05-04|15:29:12] Allocated cache and file handles database=/home/student/geth/chaindata
INFO [05-04|15:29:12] Writing custom genesis block
INFO [05-04|15:29:12] Persisted trie from memory database nodes=356 size=65.35kB time=10.523177ms
INFO [05-04|15:29:12] Successfully wrote genesis state database=chaindata
INFO [05-04|15:29:12] Allocated cache and file handles database=/home/student/geth/lightchaindata
INFO [05-04|15:29:12] Writing custom genesis block
INFO [05-04|15:29:12] Persisted trie from memory database nodes=356 size=65.35kB time=6.780313ms
INFO [05-04|15:29:12] Successfully wrote genesis state database=lightchaindata
```

Step 2: Type: **sh start-blockchain.sh**

```
student@node401:~$ sh start-blockchain.sh
INFO [05-04|15:30:02] Maximum peer count           ETH=25 LES=0 total=25
INFO [05-04|15:30:03] Starting peer-to-peer node   instance=Geth/v1.8.2-stable-b8b9f7f4/linux-arm/go1.9.4
INFO [05-04|15:30:03] Allocated cache and file handles database=/home/student/geth/chaindata cache=768 handles=512
WARN [05-04|15:30:04] Upgrading database to use lookup entries
INFO [05-04|15:30:04] Initialised chain configuration config="{ChainID: 2019 Homestead: 1 DAO: <nil> DAOSupport: false EIP150: 2 EIP155: 3 EIP158: 1}"
INFO [05-04|15:30:04] Initialising Ethereum protocol versions="[63 62]" network=2019
INFO [05-04|15:30:04] Loaded most recent local header number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Loaded most recent local full block number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Loaded most recent local fast block number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Database deduplication successful deduped=0
INFO [05-04|15:30:04] Regenerated local transaction journal transactions=0 accounts=0
INFO [05-04|15:30:04] Starting P2P networking
INFO [05-04|15:30:06] UDP listener up              self=enode://26cae4bd6a1dd523861b0f3cf821f40c212744eb1eca54e3963824689f2cd14a84f2f9ce0
0:30303
INFO [05-04|15:30:06] RLPx listener up             self=enode://26cae4bd6a1dd523861b0f3cf821f40c212744eb1eca54e3963824689f2cd14a84f2f9ce0
0:30303
INFO [05-04|15:30:06] IPC endpoint opened           url=/home/student/geth/ipc
```

Blockchain Syncing

- ❑ If the blockchain is syncing you will see this type of information:

```
> INFO [05-04|15:30:15] Block synchronisation started
INFO [05-04|15:30:16] Imported new chain segment      blocks=137 txs=0  mgas=0.000 elapsed=383.255ms  mgasps=0.000 number=137 hash=7a970e
INFO [05-04|15:30:18] Imported new chain segment      blocks=823 txs=8  mgas=2.107 elapsed=2.307s    mgasps=0.913 number=960 hash=9ef2d3
WARN [05-04|15:30:20] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:30:24] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=8.955ms   mgasps=0.000 number=1379 hash=7d733
INFO [05-04|15:30:29] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.700ms   mgasps=0.000 number=1380 hash=7ae35
INFO [05-04|15:30:34] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.680ms   mgasps=0.000 number=1381 hash=3698c
INFO [05-04|15:30:39] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=15.529ms  mgasps=0.000 number=1382 hash=ca556
INFO [05-04|15:30:44] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=10.861ms  mgasps=0.000 number=1383 hash=43356
WARN [05-04|15:30:45] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:30:49] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=12.175ms  mgasps=0.000 number=1384 hash=262bb
INFO [05-04|15:30:54] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.500ms  mgasps=0.000 number=1385 hash=c859c
INFO [05-04|15:30:59] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=13.292ms  mgasps=0.000 number=1386 hash=b96c7
INFO [05-04|15:31:04] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.004ms  mgasps=0.000 number=1387 hash=03eaa
INFO [05-04|15:31:09] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.047ms  mgasps=0.000 number=1388 hash=19cf8
INFO [05-04|15:31:14] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.896ms  mgasps=0.000 number=1389 hash=fdeee
WARN [05-04|15:31:16] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:31:24] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=8.481ms   mgasps=0.000 number=1391 hash=fa510
INFO [05-04|15:31:29] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.796ms   mgasps=0.000 number=1392 hash=ef345
```

- ❑ You can also see the blocks being appended
- ❑ Blockchain Explorer: <http://172.10.15.22:8000>

Practical Steps

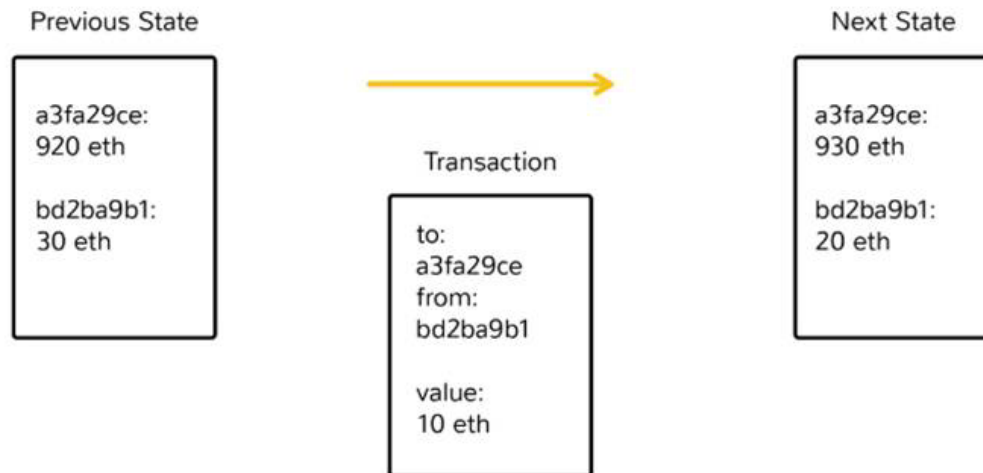
1. Create account, check balance, receive Ethers
2. Send/transfer Ethers
3. Interaction with a Smart Contract (SC)



1. First Step

- ❑ Create an account, check your balance, and receive Ethers

Ethereum



<https://www.coindesk.com/information/how-ethereum-works>

Wei	1000000000000000000
Kwei, Ada, Femtoether	1000000000000000
Mwei, Babbage, Picoether	1000000000000
Gwei, Shannon, Nanoether, Nano	1000000000
Szabo, Microether, Micro	1000000
Finney, Milliether, Milli	1000
Ether	1
Kether, Grand, Einstein	0.001
Mether	0.000001
Gether	0.000000001
Tether	0.0000000000001

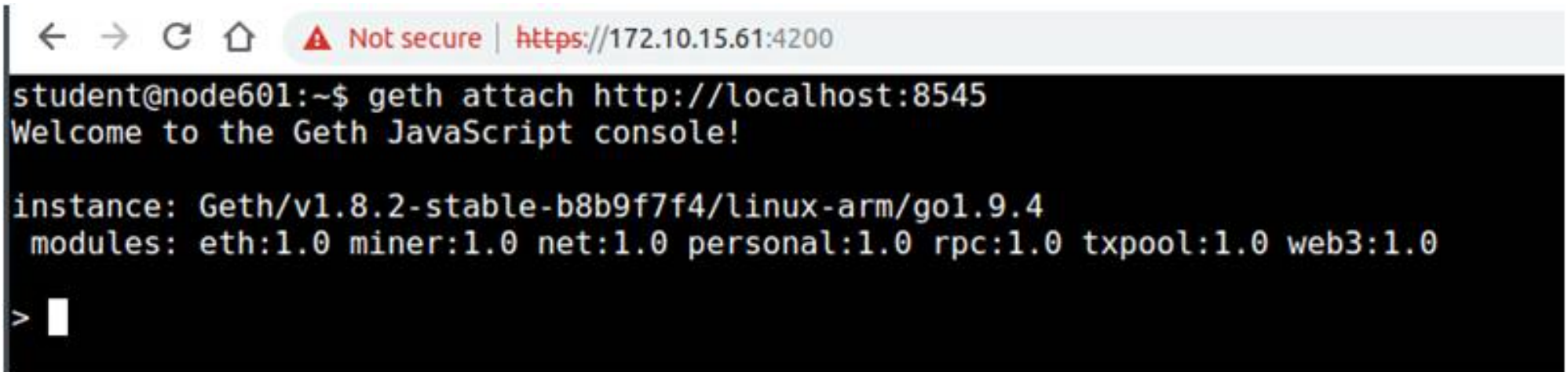
<https://etherconverter.online/>

Interacting with the Blockchain (1)

Step 3: Open a new console to your node in a new tab

□ Type: **geth attach http://localhost:8545**

□ Now you can interact with the blockchain node

A screenshot of a web browser's developer console. The address bar shows a 'Not secure' warning and the URL 'https://172.10.15.61:4200'. The console text shows a terminal session where the command 'geth attach http://localhost:8545' has been executed. The output includes a welcome message and system information for Geth v1.8.2 on a Linux ARM system with Go 1.9.4. The available modules listed are eth, miner, net, personal, rpc, txpool, and web3, all at version 1.0. The prompt is currently '>' with a cursor.

```
< > ↻ 🏠 ⚠ Not secure | https://172.10.15.61:4200
student@node601:~$ geth attach http://localhost:8545
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.2-stable-b8b9f7f4/linux-arm/go1.9.4
modules: eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
> █
```

Interacting with the Blockchain (2)

- ❑ Create a new account (do not forget your password)
- ❑ Type: `personal.newAccount()`

```
> personal.newAccount()  
Passphrase:  
Repeat passphrase:  
"0x01b93652d702a967f9eb88de3cafa770858ece34"  
>
```

- ❑ Set the new account as your main geth account
- ❑ Type: `miner.setEtherbase(eth.accounts[0])`

```
> eth.accounts[0]  
"0x01b93652d702a967f9eb88de3cafa770858ece34"  
> miner.setEtherbase(eth.accounts[0])  
true  
>
```

Interacting with the Blockchain (3)

- ❑ Unlock your account to send Ethers
- ❑ Type: `personal.unlockAccount(eth.accounts[0], "password", 9999999)`

```
> personal.unlockAccount(eth.accounts[0], "typeYourPassword", 9999999)
```

- ❑ If you want to see your account address
- ❑ Type: `eth.accounts[0]`

```
> eth.accounts[0]  
"0x688fcf36ce33356a37f129e1c1c486d7dcbba28e"
```

Check your Balance

- ❑ Set the new account as your geth main account:
- ❑ Type:
 - `eth.getBalance(eth.accounts[0])`
 - Should return 0
 - Value returned in “wei”, which is the smallest unit in Ethereum
- ❑ If you want to see your balance in Ether:
- ❑ Type:
 - `balance = eth.getBalance(eth.accounts[0])`
 - `web3.fromWei(balance, "ether")`

Receive Ethers

- ❑ Access the CAS Etherpad in your browser
 - <http://172.10.15.13:9001/p/cas>
- ❑ Inform your account to receive Ethers

```
30 Write down your name and account below:
31
32 -----
33 NAME - ACCOUNT
34
35 Eder - 0x444076e9eeafe57aa8fb63e88f942ddd1986e171
36 Bruno - 0x9d29cf9313cea1f801f2725a0e2bc82100f71616
```

- ❑ Check your balance again

2. Second Step

- ❑ Transfer Ethers to another account
- ❑ On geth console type:
 - `sender = eth.accounts[0]`
 - `receiver = "<account from colleague>"`
 - `valueEth = web3.toWei(0.5, "ether")`
 - `eth.sendTransaction({from: sender , to: receiver, value: valueEth, gas:21000})`
- ❑ Check your balance
- ❑ Verify the transaction hash in the Blockchain Explorer
- ❑ You should have less 0.5 ethers in your account

10. Smart Contracts – Practical Part II

Exercise Goals

1. Register your account in the Smart Contract
2. Retrieve an account from the Smart Contract
3. Send funds to the account retrieved



3. Third Step

□ Interacting with a Smart Contract (SC)

```
1  pragma solidity ^0.5.1;
2
3  //CAS BC 2019 Smart Contract Example
4  contract Accountbook {
5
6      address owner;
7      mapping (uint => string) accounts;
8      uint accountCount = 0;
9
10     constructor() public {
11         owner = msg.sender;
12     }
13
14     function register(string memory end) public returns (uint) {
15         accounts[accountCount] = end;
16         accountCount++;
17         return accountCount-1;
18     }
19     function getAccountCount() public view returns (uint) {
20         return accountCount;
21     }
22
23     function getAccount(uint n) public view returns (string memory) {
24         return accounts[n];
25     }
26 }
```

Each account has an unique ID

First ID is 0

New registered account receives an ID

Returns the account associated with the ID

SC Storage

- What data does the SC stores?
 - List of IDs and Account (Example)

ID	Account
0	0xC0A565abfa50349c59a4D0647834Bb6763b8fFC9
1	0xe3c01b34bBcfeDf5D5Cf090226D50767c8DBC029
2	0xd41aA1F353CdcC3d3bd22D2EcD2e4fF490f2847
3	0x5e694ceCb56d4C24a79547468B80c5ffD186B06F
4	0xAF418BAe762962dEdb6aC027A0706972260F910d
5	0xeB23D0a558482D6D706C51234F4210aef0797F5F
6	0xAB38C7C0e9FA8f30bFF71c25781e1207E6BCfb06

- Number of Accounts: 7
- New Account will have ID **7**

3.1 Step: Interact with the SC

- ❑ Create the functions to interact with the SC
- ❑ Type: `loadScript('initContract.js')`
- ❑ Retrieve the number of accounts stored in the SC
- ❑ Type: `example.getAccountCount()`
- ❑ Get the account with ID `0`
- ❑ Type: `example.getAccount(0)`
- ❑ Send 0.1 Ethers to the returned account address
- ❑ Check your balance

3.2 Step: Register your Account

- ❑ Register your account in the SC
- ❑ Type:

```
example.register(eth.accounts[0], {from: eth.accounts[0], gas: 100000})
```

The account to register

Sending from
your account

Gas paid to miners
to execute your
transaction (tx fee)

- ❑ Retrieve the number of accounts stored in the SC
- ❑ Type: `example.getAccountCount()`
- ❑ Was there any changes in the number of stored accounts?
- ❑ Find your account ID and from colleagues 😊

11. To Blockchain or not to Blockchain?

Key BC Characteristics

- ❑ **Permissions** (write/read)
 - **Fully**: everyone in the world can participate (W/R)
 - **Partially**: selected public can participate (W/R)
- ❑ **Transparency**
 - **Fully**: all members access the history of transactions
 - **Partially**: some members access the history of transactions
 - Better than a traditional DB
- ❑ **Permanent storage, *i.e.*, immutability**
 - Transactions cannot be removed
- ❑ **Level of decentralization**
 - **Fully**: consensus with elected leader
 - **Partially**: consensus with selected leader(s)

BC Advantages and Drawbacks

Selection only!

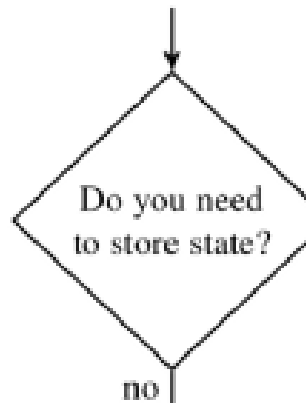
Characteristics	Advantages	Drawbacks	Remarks
Distributed	No central control, no “master” needed	No central control, no “master” exists	Censorship vs. conflict resolution!
Unknown stakeholders	Everyone can participate	Lacking control of participants’ “writes”	Application-specific needs
Open, transparent	No hiding possible	Stakeholders’ activities publicly viewable	Application-specific needs
Immutable	Once persisted, persisted forever	Wrongly deployed SC(s) not retractable	Realistic for “useful” SCs? GDPR?
Append-only	No deletions	Growing in size	GDPR compliance?
Traceable	Proof of actions	No hiding of actions	Error handling?
Technical aspect	Effective	Efficiency, energy dem.	Sustainability?
Economic aspect	Cryptocurrency (fully elect., decen.)	Impacts on economic stability, currencies, ...	Survivability of too many “tokens” or “coins”?
Legal aspect	Contracts without intermediary	“Unknown” conflict resolution instance(s)	No jurisdictional borders, enforceability?

Choosing a Blockchain

- ❑ Not a **trivial** question to answer
 - ... besides these 2900+ BCs available
- ❑ **Myriad of facets/parameters**
 - Marketcap
 - Value to buy all shares at today's market value
 - Community involvement
 - Telegram chats, discussion channels
 - Full Node/Miners Geolocation
 - Politics, possibility of centralization
 - Technical concerns
 - Transactions per second, implementation language, design choices
 - Security
 - Hashing algorithm, possible attack vectors

[1] <https://coinmarketcap.com/>

Blockchain Demand “Checker”



K. Wüst, A. Gervais, 2017+2018

12. A Critical Review

Neither Good* nor Bad*, Only “as is”!

* “Good” or “bad” qualifications can be determined from results of evaluations, which depend heavily on models and assumptions!

Reviewing Bitcoins

- ❑ Fully decentralized and distributed
 - + everyone can participate, no central bank needed
 - + everyone interested can access full transaction history
 - + almost no change of the history of transactions
 - no control of “everyone”
 - no-one to complain to
 - history of transactions grows forever
- ❑ Impacts on person's payments (daily society)
 - + person-to-person transactions without any intermediary
 - + performance at very low (varying) payment costs
 - user interactions are final
 - no overall maintenance and surveillance

Reviewing “Digital Assets”

□ Societal impacts

– Tokenization of everything

- Data generated by electronic algorithms (e.g., mining) vs.
- Data “sensed” from the real world (from objects, Internet-of-Things)

– Digitization reached the “value” (besides representations)

- Electronic data maintained immutably (no copy&paste of assets)
- IoT data “hard” to be made valuable

– Consensus mechanisms’ sustainability dimension

- Very strong, thus, highly secure (such as PoW) are energy-demanding
- Less energy-demanding ones (such as PoS) are much less secure
- Byzantine Fault Tolerance ok, centralized with trust assumptions

– Application-specific

- Cryptocurrencies, supply chains, insurances, ...

PoW: Proof-of-Work
PoS: Proof-of-Stake

Public Blockchain Challenges!

Selection only!
Solutions seem possible

- How to handle reliably **tangible (non-digital) assets** in BC?
 - A token is represented in **bits** vs. property/real estate as **physical items**
- **Sustainability**: **Energy efficiency** of consensus mechanisms?
 - Energy consumption Bitcoin BC only in 2018 $\approx$ The Netherland's production
- **Scalability**: BC throughput as a number of **transactions per second**, **volume of data** persisted in Mega (?) bytes, **costs**?
 - *E.g.*, BC sizes grow faster than the density of HDDs/SSDs
 - BC (always) better than a (distributed) data base? Exorbitant costs?
- **Identity management** (users, objects) and **anonymity**
- **Standardized APIs** for switching BCs for BC-based dApps
 - *E.g.*, in contrast, databases from different vendors offer “similar” APIs
- Many **economic effects** of BC-based cryptocurrencies unknown
 - Role of **national “e”-currency**, **interrelationships** of about 2100+ cryptocurr.
- **Legal/regulative** compliance, **societal/governmental** acceptance

Public Blockchain Risks!

Selection only!
Fundamental concerns

- ❑ BCs' "true semantics" depend on the input received!
- ❑ BCs' security, privacy, and reliability
 - Unknown attack vectors (& 51% attack), Programming errors in SCs
 - Alternative consensus mechanisms beyond PoW? Security at stake?
 - The breaking of currently used security algorithms
 - Long-term storage? Quantum Computing impacts?
 - Privacy: persisted data at stake? GDPR?
 - The right to forget vs. immutability
 - Transparency (public knowledge of BC) vs. privacy (private data)
- ❑ Networking infrastructure's reliability (critical infrastructures)
 - Lacking Internet connectivity for a "longer" period of time?
- ❑ Economic/legal risks (cryptocurrency/tokens/coins, BC)
 - Fraudulent profitability projections, volatility, dispute resolutions, no central entity or simply a central Customer Relationship Management

Private Blockchain Risks? Challenges?

Selection only!
Real needs?

- How to handle reliably **tangible (non-digital) assets** in BC?
 - A token is represented in **bits** vs. property/real estate as **physical items**
- **Sustainability**: Energy efficiency of consensus mechanisms!
 - Known stakeholders' tx persisting based on round-robin schemes
- **Scalability**: BC throughput as a number of transactions per second, volume of data persisted in “any” volume
 - *E.g.*, BC sizes grow faster than the density of HDDs/SSDs
 - BC (always) better than a (distributed) data base?
- **Service** offered: **trusted third party time stamping service**
 - Cryptographically sealing electronic data with a tamperproof digital signature and BC timing information (well known from the past)
- BCs' **security, privacy, and reliability**
 - Depending on known stakeholders' agreements (like within today's IT)
 - With a **very different trust model** as compared to public BCs

Broader Challenges? Risks?

- ❑ Discontinuation of “cash” and replacement with a cryptocurrency – loss of any anonymity?
- ❑ Cryptocurrencies only – failure of energy supply?
Sustainability? Payments made with empty battery?
 - On-the-spot payments? Input channels are “delicates”
- ❑ Decentralization of token handling – full control of the individual? No central “CRM” for sure CRM: Customer Relation Management
 - Full decentralization today impossible, thus, “central” players
- ❑ Reliability of IT systems – here “billions” of handhelds are part of the game? BYOD? BYOD: Bring Your Own Device
IF: Interfaces
 - Different OSes? Different HW? Hundreds of applications? IF?

Probabilities and Blockchains (1)

❑ Public-private key pairs:

- Two public-private keys may be the same
 - **Probability** of that “astronomically low” (not zero), technically irrelevant
 - Primality tests: Sieve of Eratosthenes, probabilistic Miller–Rabin tests
 - Composite number at 75% chance of being detected per round
 - After 64 rounds of testing, the probability that composite number is undetected is at probability of 2^{-128}

❑ Mining:

- **Probability** of winning “block race” proportional to computational power of participants
 - Time restriction on block mining; Bitcoin set to the block time: 10 min
 - Computational, hashing power increase $\Rightarrow$ chance of discovering a new block under 10 min increases $\Rightarrow$ increase target value

Probabilities and Blockchains (2)

- Centralized mining with unfairness and 51% attack
 - Incentives to “combine” miner into pools
 - Chances to take over a BC from a country with “many” miners
 - Mining power of above 51% at one country

M. Saad et al.: Exploring the Attack Surface of Blockchain: A Systematic Overview, 2019

□ Consensus mechanisms (distributed)

- **Randomness** for multiple parties trying to come to agreement
 - Constraints as upper bounds on (a) number of messages sent within a block time (throughput) and (b) time it takes for message to be sent (latency) $\Rightarrow$ BC (public) make it hard to “consent” an agreement via messages needed $\Rightarrow$ solved by PoW and messages needed!
 - PoS: selection of subcommittee $\Rightarrow$ unbiased? Not possible!

□ Stakeholders (maliciously behaving trusted nodes)

- **Probabilities** of an “internal” attacker within a given private BC

Conclusions (1)

1. Blockchains **do** show a logical evolution of linked lists, however, public BCs “exaggerate” processing demands
 - By design: Proof-of-Work (PoW), but this ensures immutability
2. Traceability of objects, especially food, **viable** via BCs, modulo “trust in sensor” and “coloring” problem, contradiction of full transparency vs. internal matters remains
 - Role of BC vs. DL only arbitrable on a case-by-case analysis
3. Blockchains **do not** solve the media break, thus, leading to a lack of trust for tangible assets, key: trust in “sensor”
 - Major trust model differences for public BC vs. “private” BC/DL

Conclusions (2)

4. The technical future of blockchains is still based on ***security ingredients*** of today's technology, however, long-term security management is key
 - “Unknown” impact of quantum computing (certainly on all IT!)
5. Blockchains and Distributed Ledgers ***have reached society***, especially FinTech, but not necessarily daily life
 - Visibility of technical basis as hidden from users as for other IT
6. Blockchains show ***no revolution***, but a typical Computer Science (Abstract Data Type) ***evolution*** of linked lists
 - But “distribution” of consensus ***does not always*** make sense
 - Any system as of the past has ***not*** been replaced fully by a BC

Disclaimer

- ❑ All material was prepared with utmost care.
- ❑ Technical details, planned and published use, and real installations may change without notice.
- ❑ Different versions of software, clients, systems, documents, papers may show very different effects, functionality, or behavior.

Thank you for your attention!

