Le génie pour l'industrie

[fuseelab.github.io](fuseelab.github.io)

# Deconstructing Blockchains: Concepts, Systems, and Insights

Blockchain @ SACMAT: blockchain-conf.github.io

Link to our companion papers:
http://msrg.org/papers/bcbi-tr

BY KAIWEN ZHANG
ÉTS MONTRÉAL
UNIVERSITY OF QUEBEC

# *Acknowledgments*

Le génie pour l'industrie

*fuseelab.github.io*

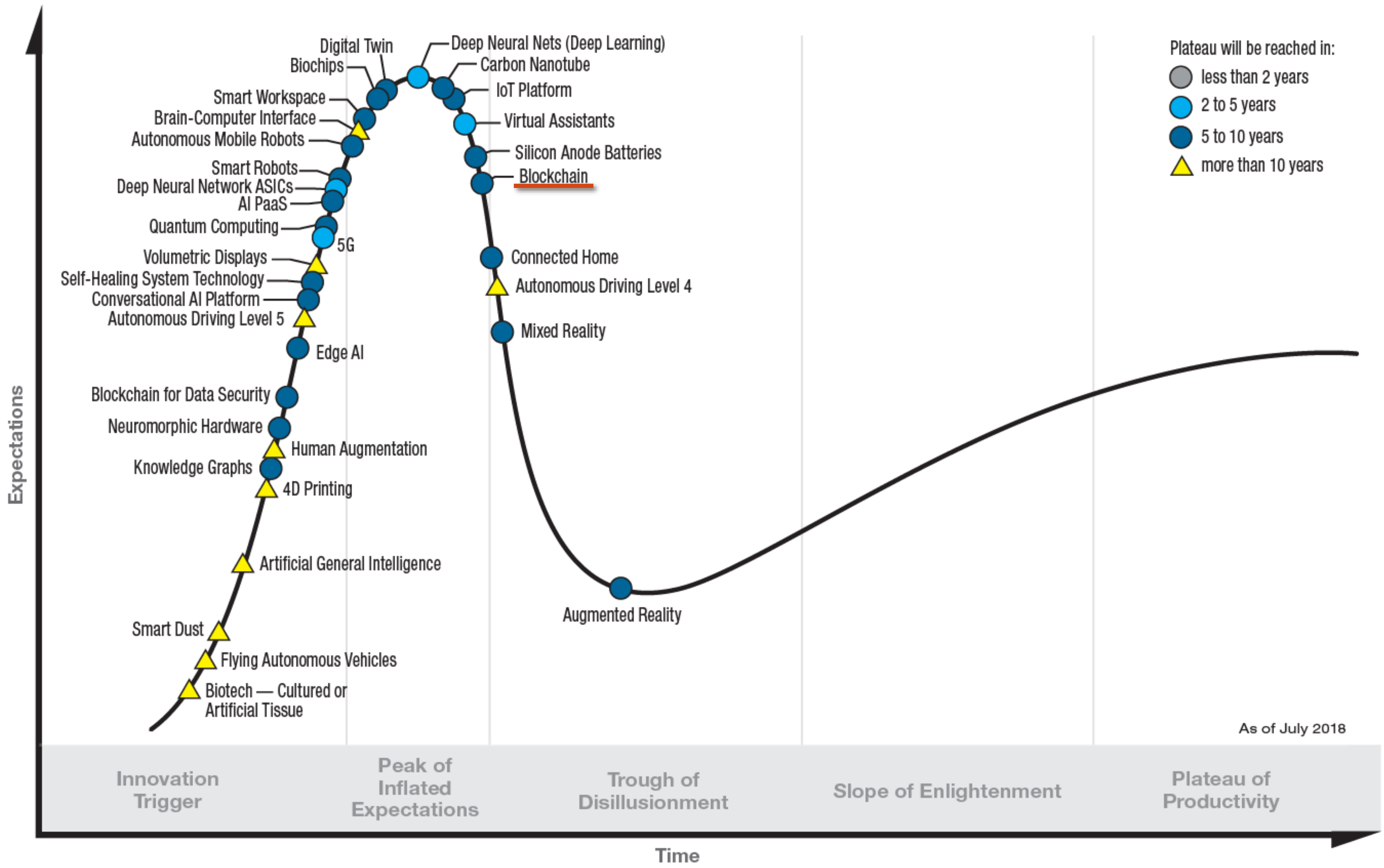
UNIVERSITY OF TORONTO

Collaborators:

- Kaiwen Zhang
- Hans-Arno Jacobsen
- Roman Vitenberg
- Mo Sadoghi

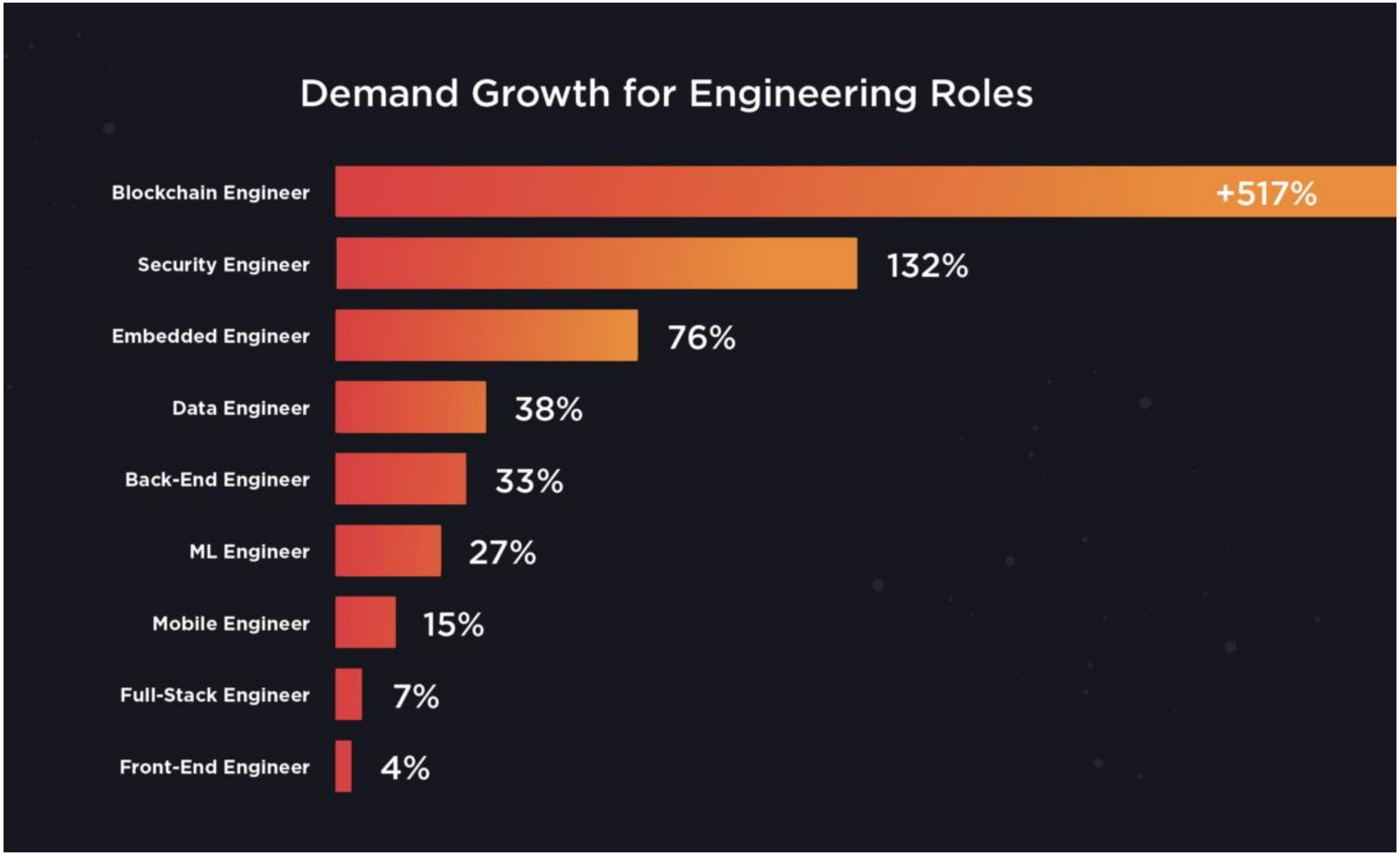
# Understanding Blockchains

# Hype Cycle for Emerging Technologies, 2018



# Comparison with BTC price

# Demand for blockchain jobs




*computerworld.com/article/3345998/demand-for-blockchain-engineers-is-through-the-roof.html*

## Blockchain and Fortune 100 Companies

You may say that I'm just a freelance blockchain writer and my opinion doesn't matter. Yes, I totally agree with that and that's the reason why I attach the list of Fortune 100 companies already working on the implementation of the blockchain solutions in all spheres of human society. According to Cryptotapas, 82% of Fortune 100 companies work with blockchain. The list below is quoted from the same article:

***1. Walmart***

*Walmart is implementing blockchain for its food businesses.*

***2. State Grid***

*The State Grid Corporation of India is using blockchain technolgy to impove data sharing.*

***5. Royal Dutch Shell***

*Royal Dutch says Blockchain will revolutionize and disrupt oil industry to trillion Dollar Industry.*

***6. Toyota Motor***

*Toyota seeks blockchain technology in developing Self Driving Cars.*

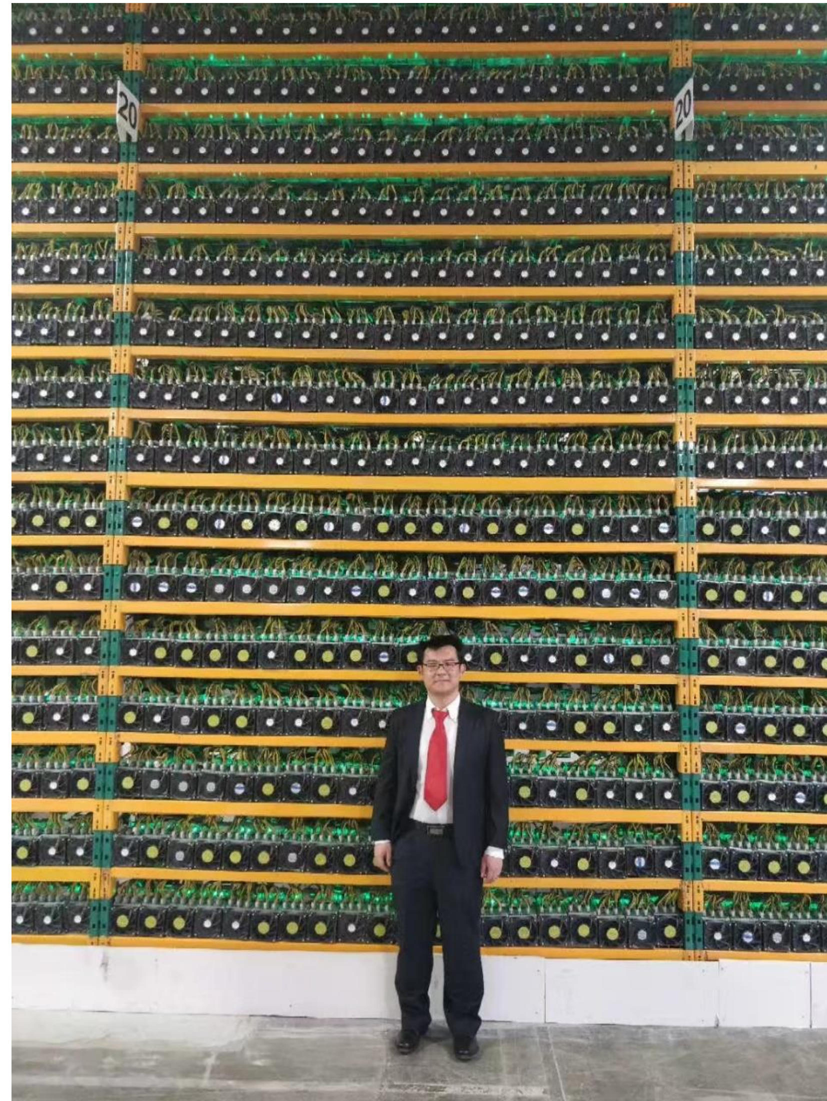
***7. Volkswagen***

*Volkswagen implements and backs Blockchain technology to drive the automobile industry to a new level.*

https://medium.com/altcoin-magazine/blockchain-to-become-a-commonplace-for-fortune-100-companies-3a302526d8eb

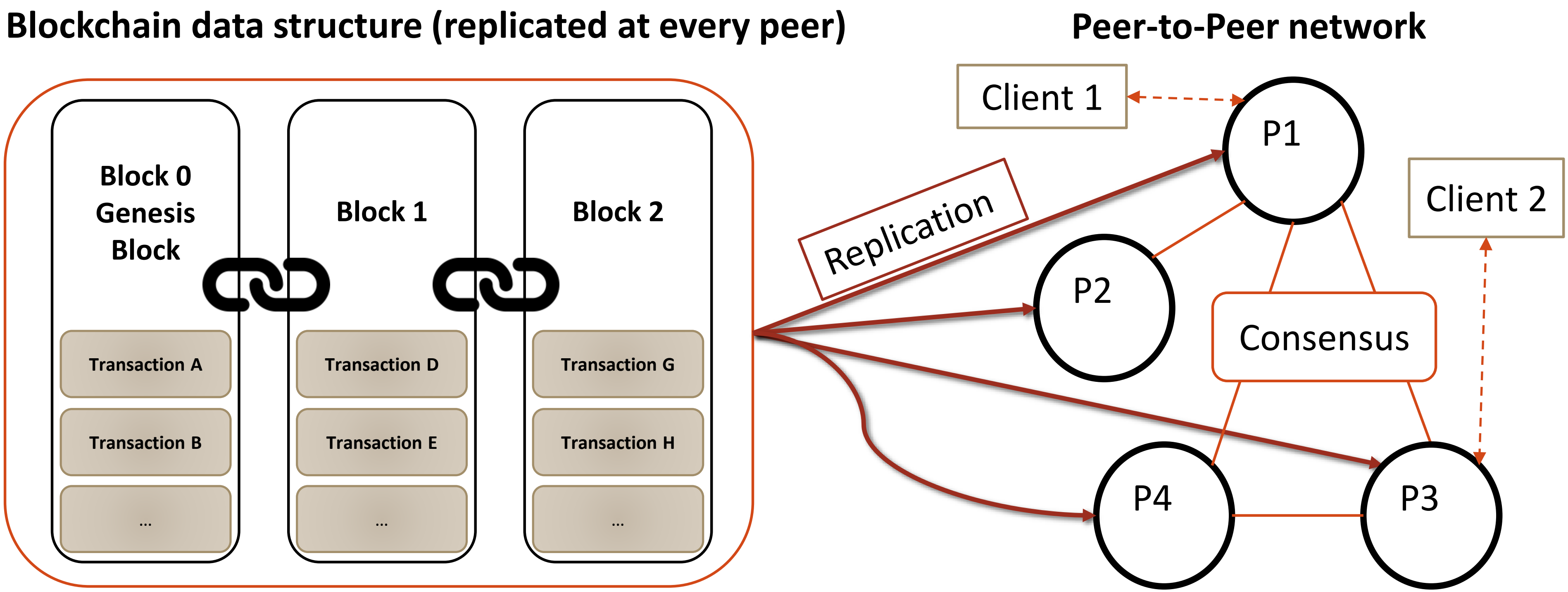
# Mining industry in Quebec

Bitfarms™



# Blockchain 101

Distributed Ledger Technology (DLT)



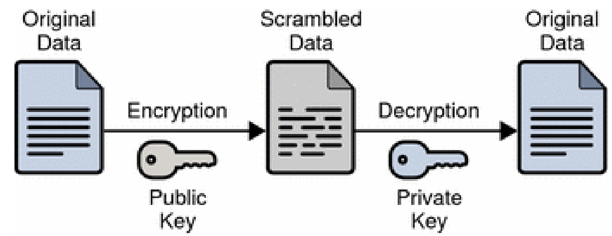

*Cryptography is used to…*

***…encrypt data, prevent modification, insert new blocks, execute transactions, and query…***

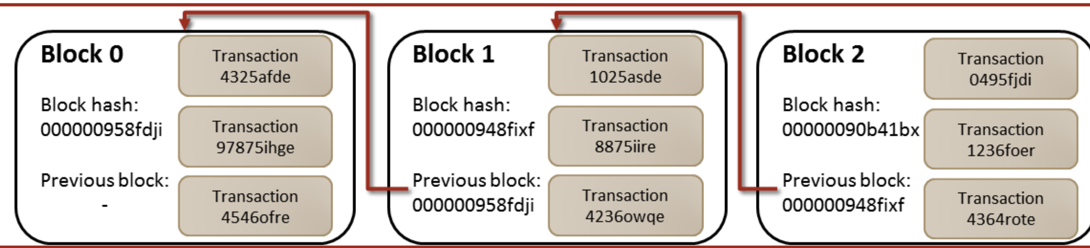
*the distributed ledger*

# Cryptography: the Magic Ingredient!

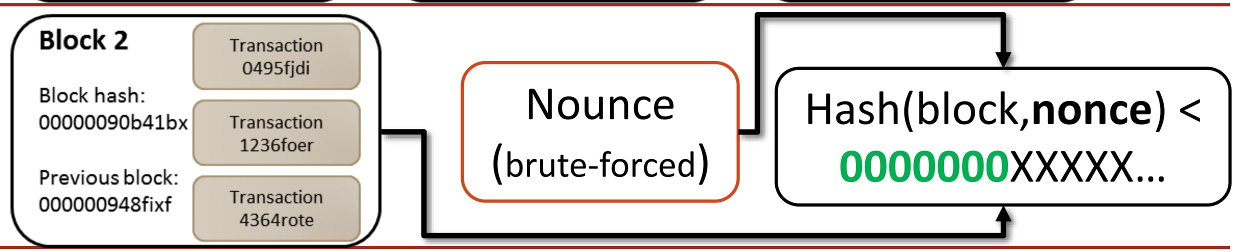
Encrypt data:
**Public Key Encryption**



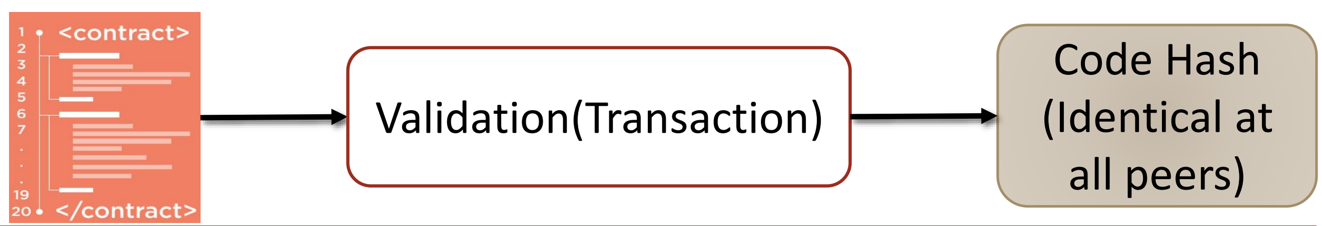

Prevent modification:
**Hashed Linked List**



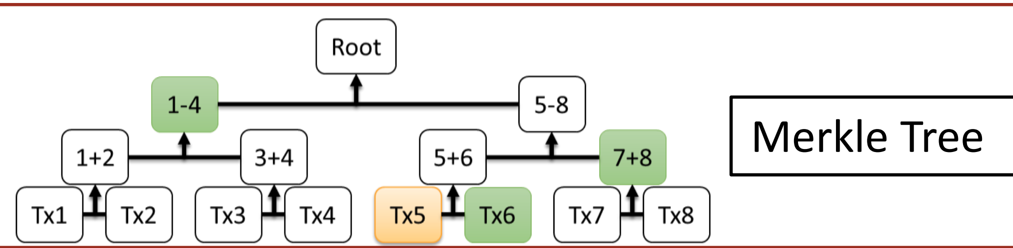

Insert new blocks:
**Proof-of-Work**




Execute transactions:
**Smart Contracts**




Query the blockchain:
**Simple Payment Verification**



# What is a blockchain-based distributed ledger?

- ✓ *An append-only log* storing transactions
- ✓ Comprised of *immutable* blocks of data
- ✓ *Deterministically* verifiable (using the *blockchain* data structure)
- ✓ Able to execute transactions *programmatically* (e.g., Bitcoin transactions and smart contracts)
- ✓ *Fully replicated* across a large number of peers (called miners in Bitcoin)
- ✓ *A priori decentralized*, does not rely on a third party for trust

# Comparison with Databases

| | Single Machine DBMSs | Distributed Databases | | |
|---|---|---|---|---|
| | | OLTP | OLAP | |
| Logically centralized (Single entity) | MySQL, Oracle, DB2, … | NewSQL: Spanner, VoltDB | Distributed SQL data warehouses | Relational |
| | Berkele LevelDE | | , duce | Non-relational |
| Decentralized (Public/Private) | | Distributed Ledgers (DLT) | | Blockchain |

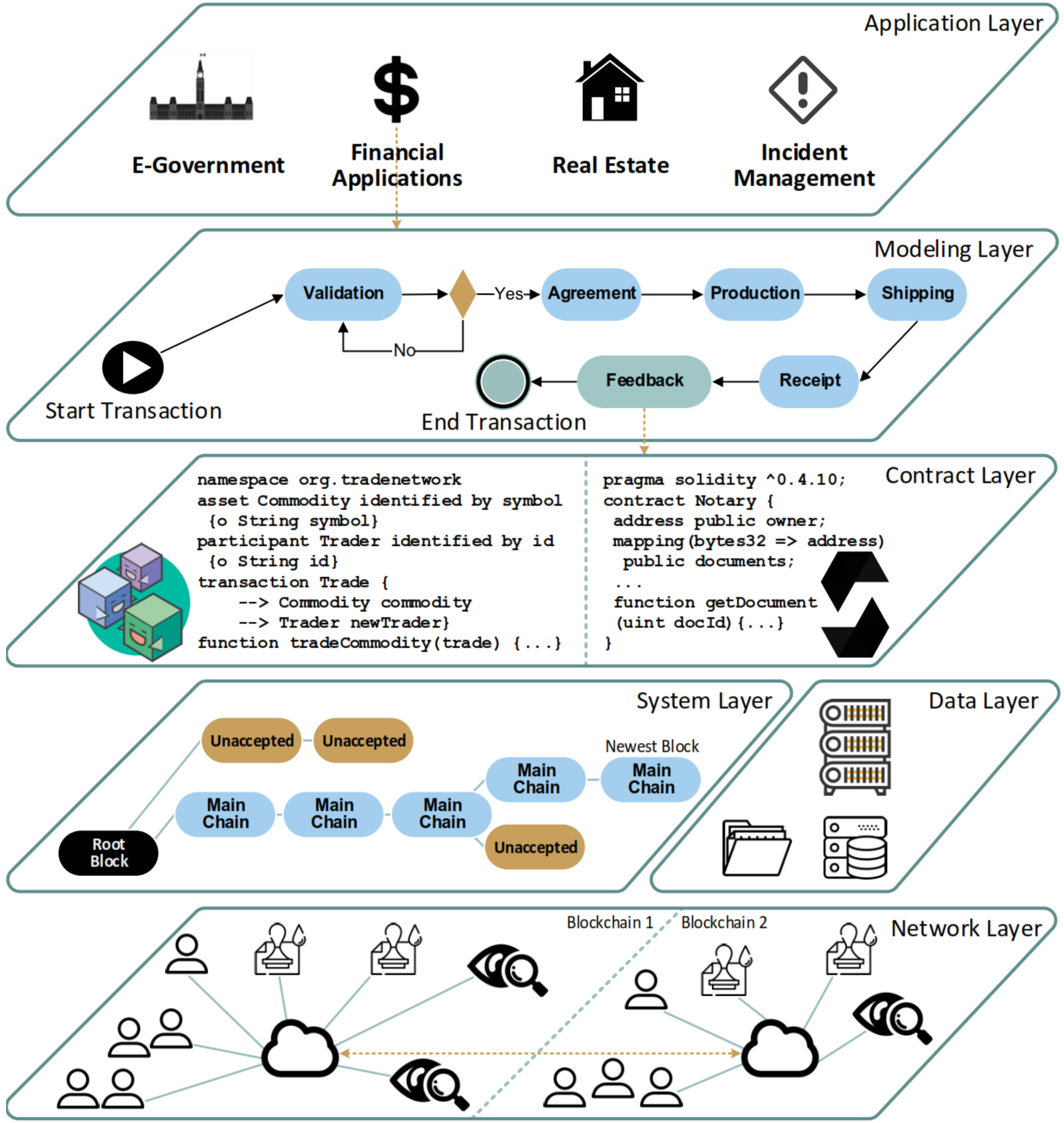
The key distinction is the use of *cryptography* to enable operation in a decentralized trustless environment.

# Blockchain Reference Architecture

This vision diagram encompasses all aspects related to blockchain technologies.

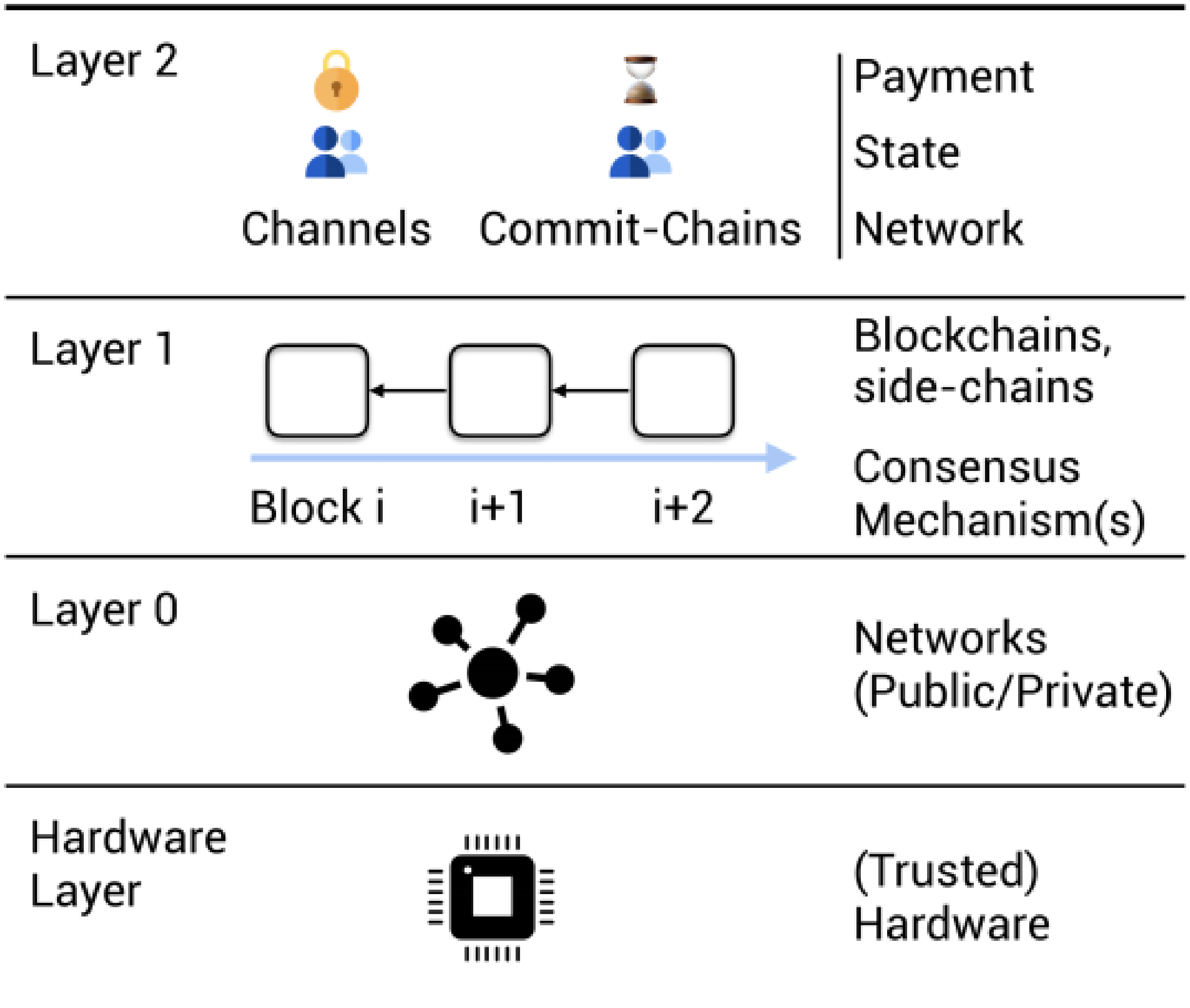
**Upper layers** capture application semantics and their implementation.

**Lower layers** are concerned with technical system details.




ZHANG ET AL. © 2019

# System-Oriented Perspective



# Outline

## Session 1: Foundations

- Bitcoin: Consensus, transactions, networking, rewards

## Session 2.1: Beyond Bitcoin

- Smart contracts
- Platforms: Ethereum, Hyperledger


## Session 2.2: Research

- System insights
- Research directions

## *Session 4: Hands-on tutorial on Ethereum*

- *Smart contract development and deployment*
- *Tools for deploying and managing Ethereum*

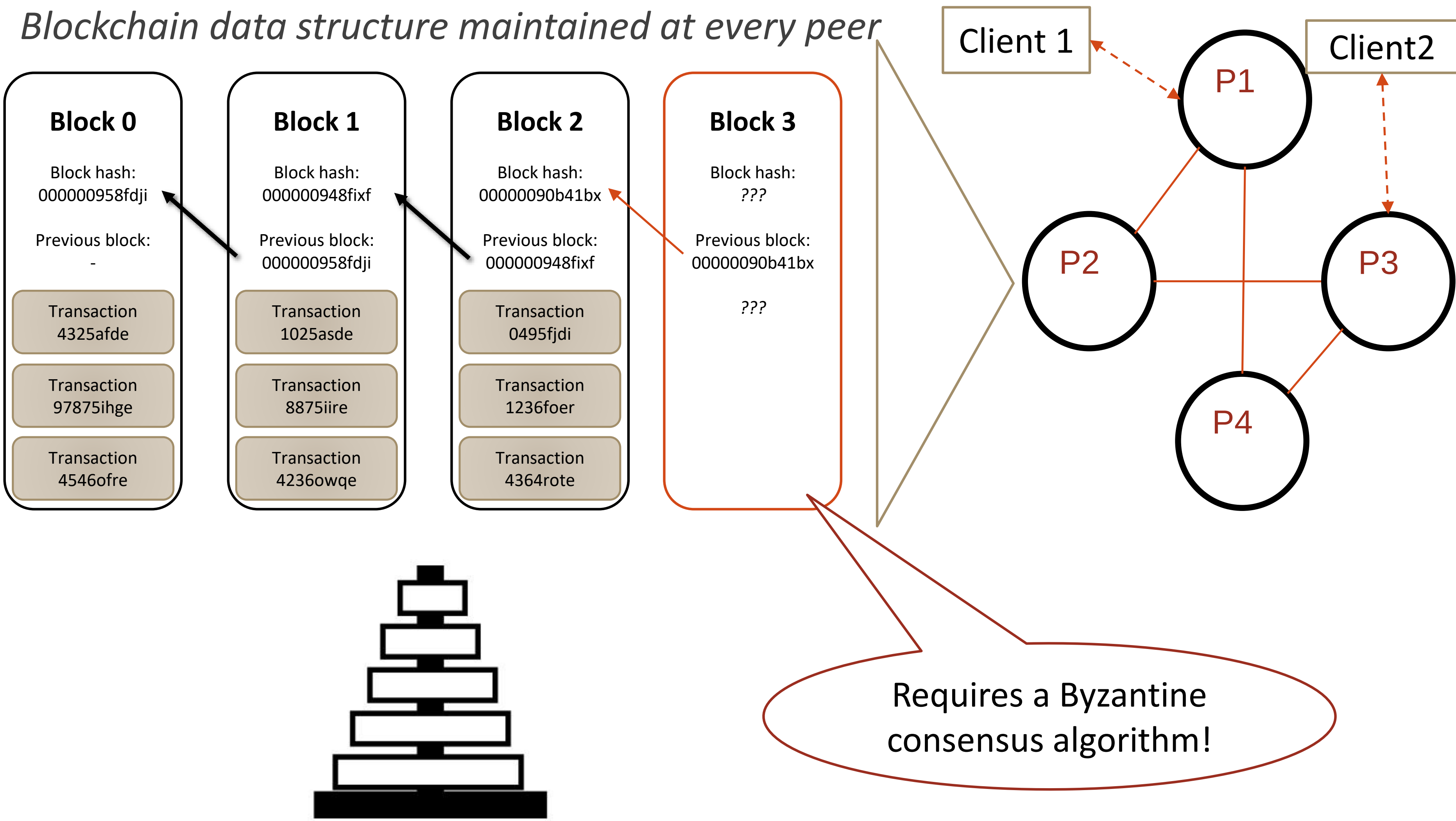
# Blockchain Concepts

DEFINITIONS

BITCOIN OVERVIEW

# Immutability using Hashing

*Blockchain data structure maintained at every peer*



# Consensus

# Consensus in Bitcoin

Byzantine consensus in history

- Dozens of impossibility results since 1983
- Does not scale beyond 30 participants
- Takes a long time to converge

Bitcoin requirements

- Decentralized and public network
- Supports 10,000 participants

Key insight: Probabilistic consensus

Make a proposal => Proof-of-Work

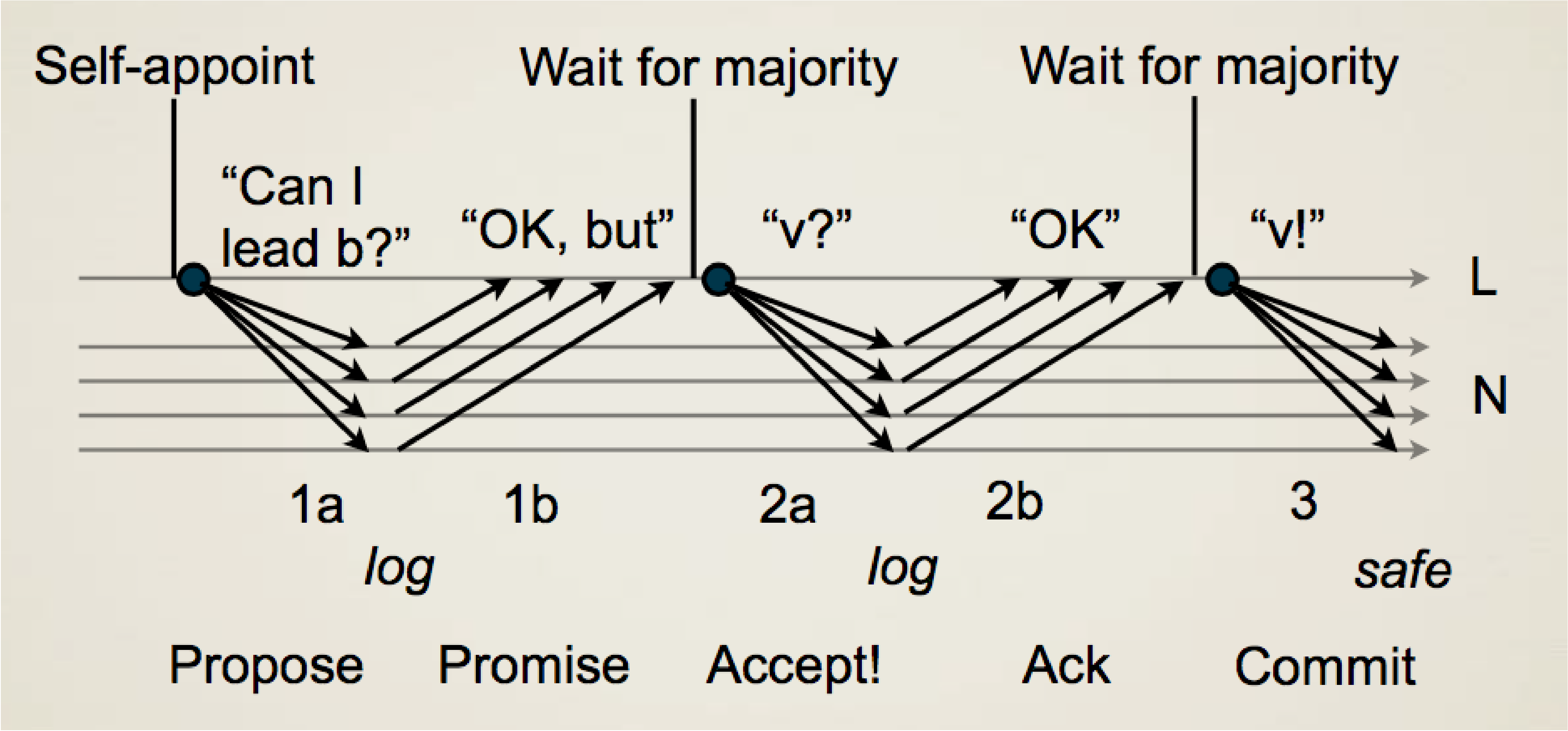
↓

Decide a value => Longest branch selection

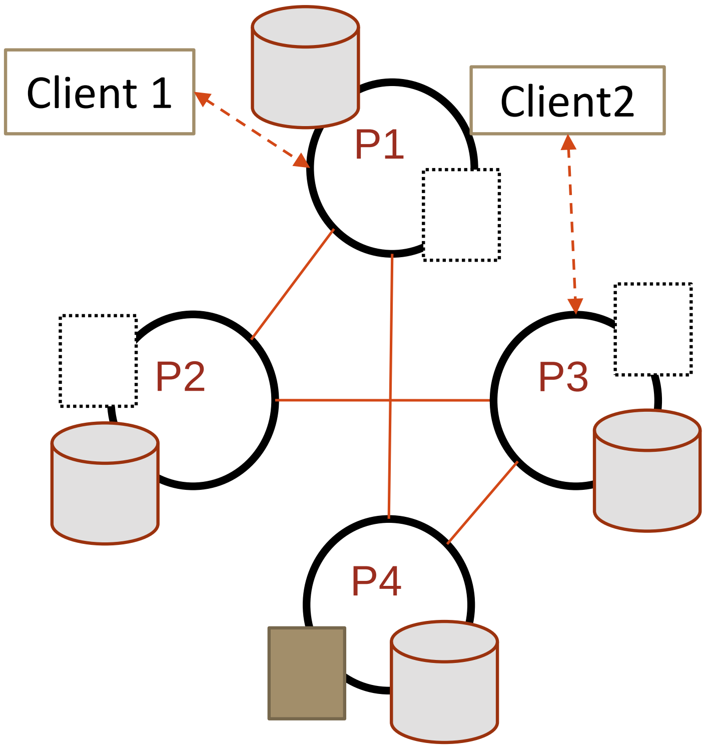
↓

Announce the decision (finality) => Confirmations wait

# Comparison with Basic Paxos



# Block Proposal: Proof-of-Work




Each client maintains a *mempool* of unconfirmed transactions

Each peer constructs its own block it wants to propose

- Free to pick and choose transactions from its own *mempool*

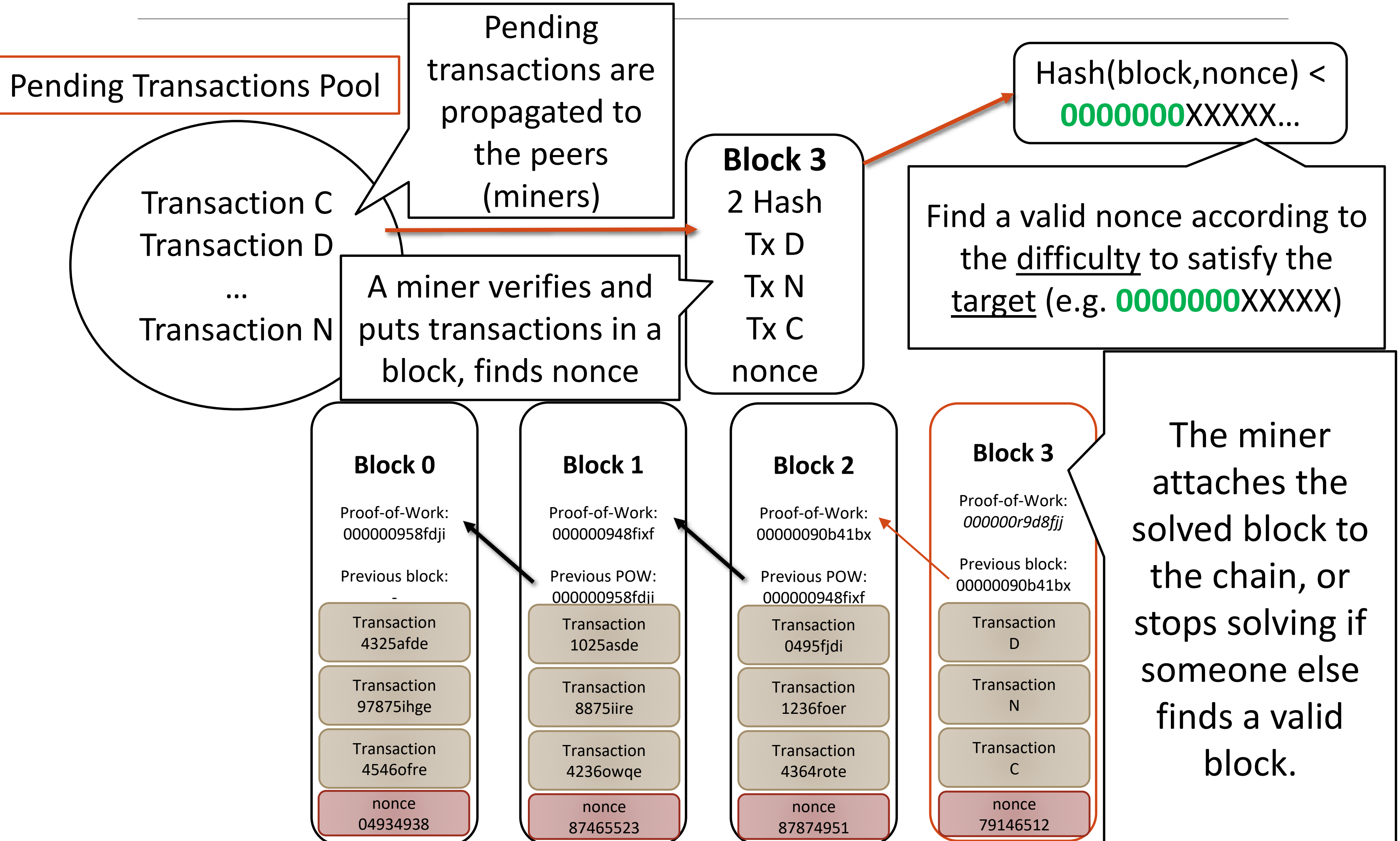
The fastest peer to solve the *cryptopuzzle* of its own block can propose the block to others

- The block is sent through the P2P network

Other peers can verify the validity of the cryptopuzzle solution

Repeat the process for the next block

# Point of view of a miner



# Cryptopuzzles in Bitcoin

The proposer has to find ***nonce***, such that

- *hash(**block_header**) < **target***

***target*** is a fraction of the hash space

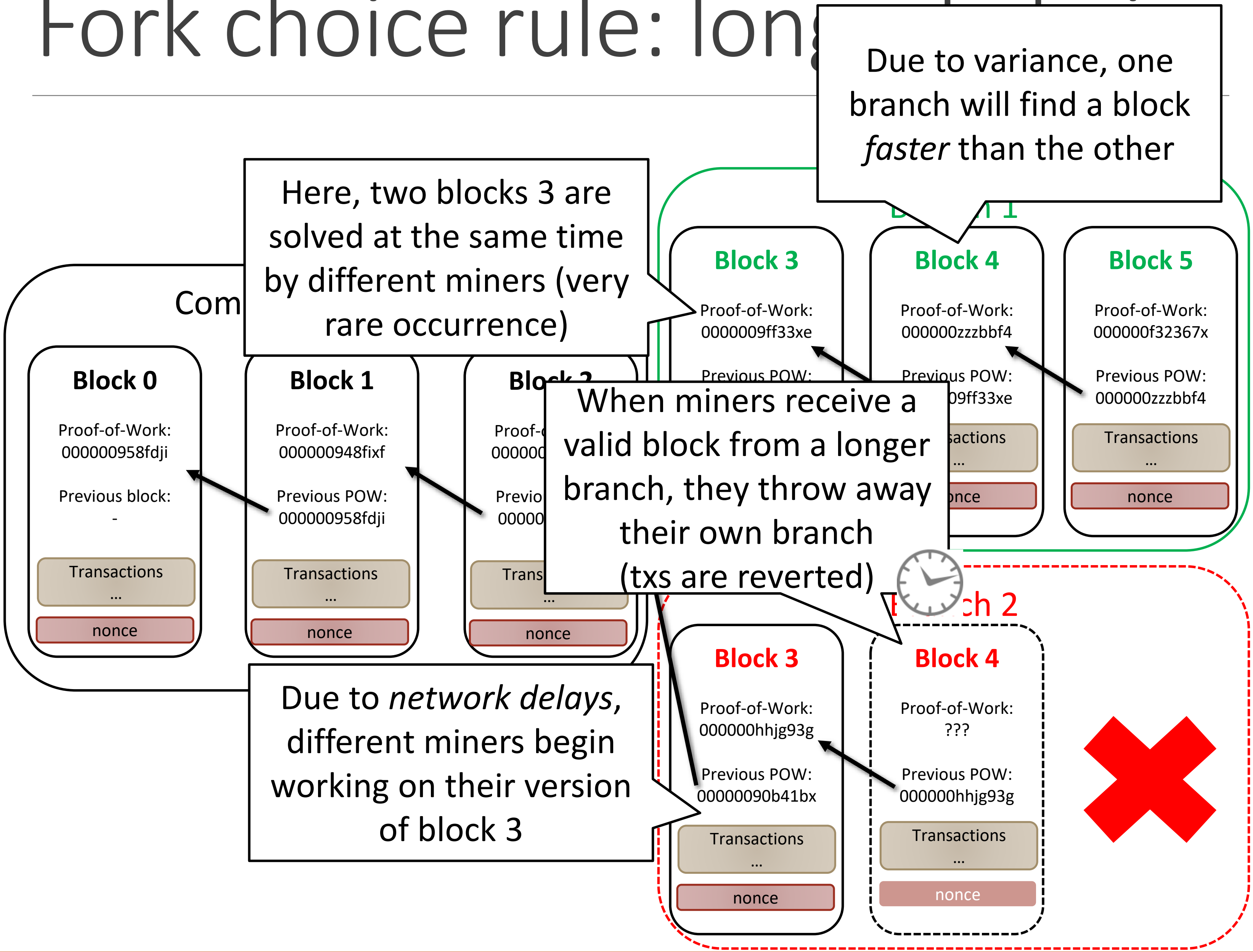
- Every node recomputes ***target*** every 2016 blocks
- Such that the average time for the whole network to solve a cryptopuzzle is 10 min
- A block time of 10 minutes ensures a significant amount of work is required to propose block
- Normally, only one block is proposed at a time, which simplifies consensus

For proposer *p*,

$$mean\ time\ to\ next\ block = \frac{10\ minutes}{fraction\ of\ p's\ computing\ power}$$

The solution is fast to verify

# Fork choice rule: lon



# Announcing results: Confirmation wait

When a transaction is included in a **newly mined block**, it is said to have “one confirmation”.

Each subsequence block mined afterwards **adds one confirmation** to the transaction.

The more confirmations a transaction have, **the more likely** it is to stay in the blockchain.

Each client is free to choose **how many confirmations** to wait for in order to consider a transaction as committed to the blockchain.

With high probability, a client is recommended to wait for **6 confirmations** before considering a transaction completed.

Note that **Bitcoin lacks finality**: a transaction can never be 100% guaranteed to stay in the blockchain!

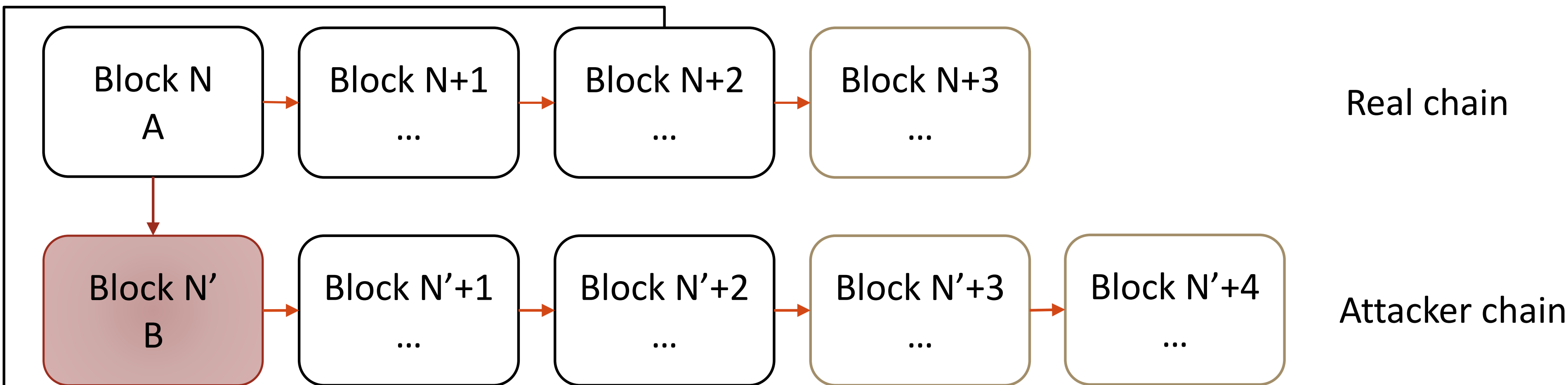
# Preventing double spending

**(51% Attack)**

Transaction A
฿1 ->
Merchant 1

Transaction B
฿1 ->
Merchant 2

A malicious attacker creates two transactions using the same money (*double-spending*)




- The *continuous generation* of blocks in the main chain *limits the amount of time* an attacker has to create its own chain.
- If the attacker owns *>51% of the power* in the network, it will eventually surpass the main chain and be able to *tamper existing data*!

It must replace A with B in N, and solve the modified puzzles for the blocks faster than the real chain grows so that it can become longer

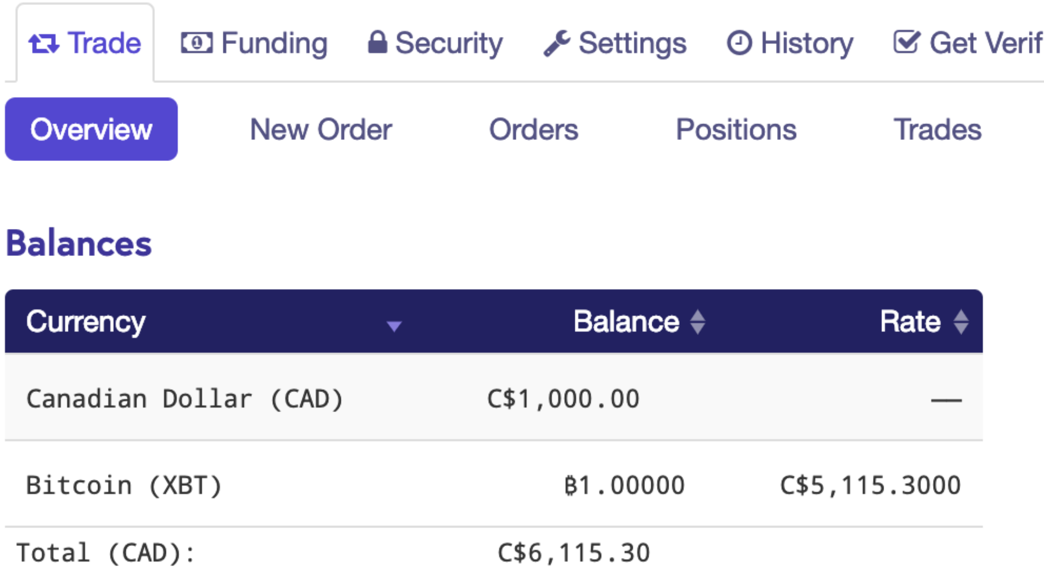
# Why maintain Bitcoin?

## Two incentive mechanisms in Bitcoin

- Block creation reward: a block proposal creates a number of new bitcoins and transfers them to the proposer
  - The only way to create new bitcoins
  - The amount is predefined and gets halved every 210,000 blocks
  - Predicted to go down to zero before year 2140
  - The geometric progression totals to 21 million bitcoins
- Transaction inclusion fee: Alice can decide to pay a small fee to the block creator as part of her transaction
  - Voluntarily, there is no predefined amount
  - Miners will naturally prefer to mine transactions with higher fees
- These fees are collected in the **coinbase transaction**
  - Sends the bitcoins to the address of the miner

# Transactions

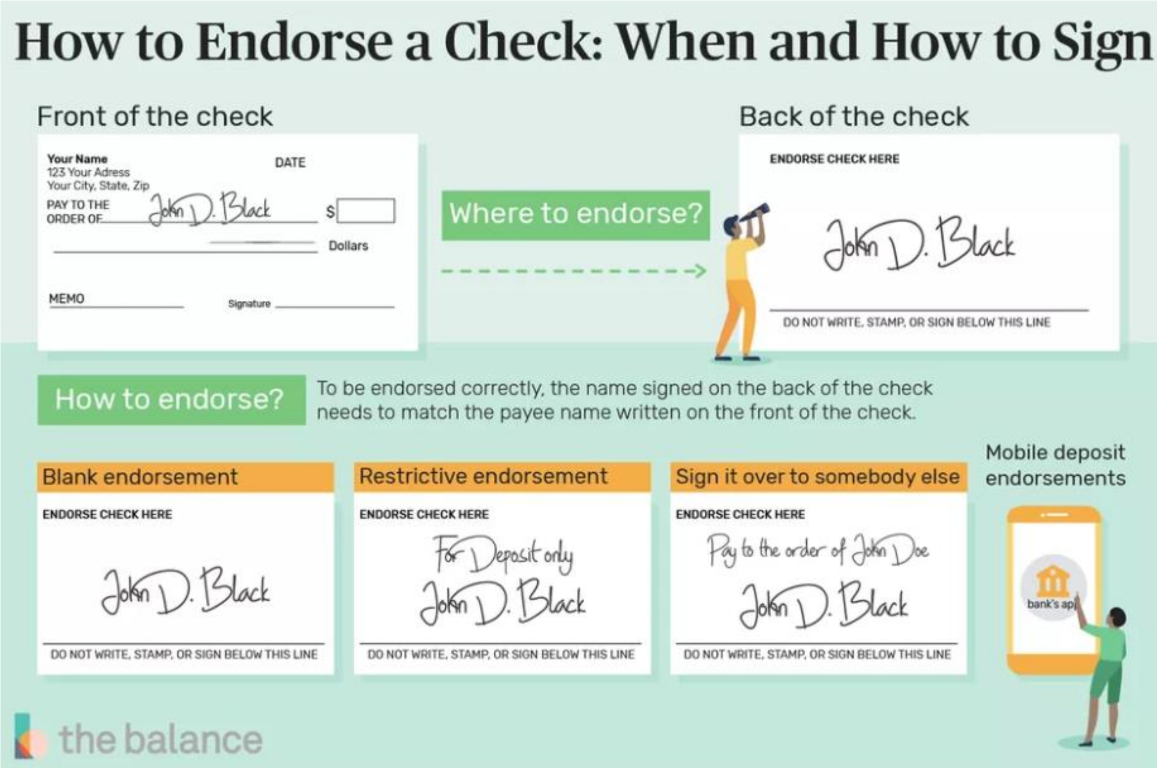
# UTXO vs. Balance




In the balance model, the system maintains the sum of currencies held by an account

It is the most popular and intuitive model

# UTXO Model

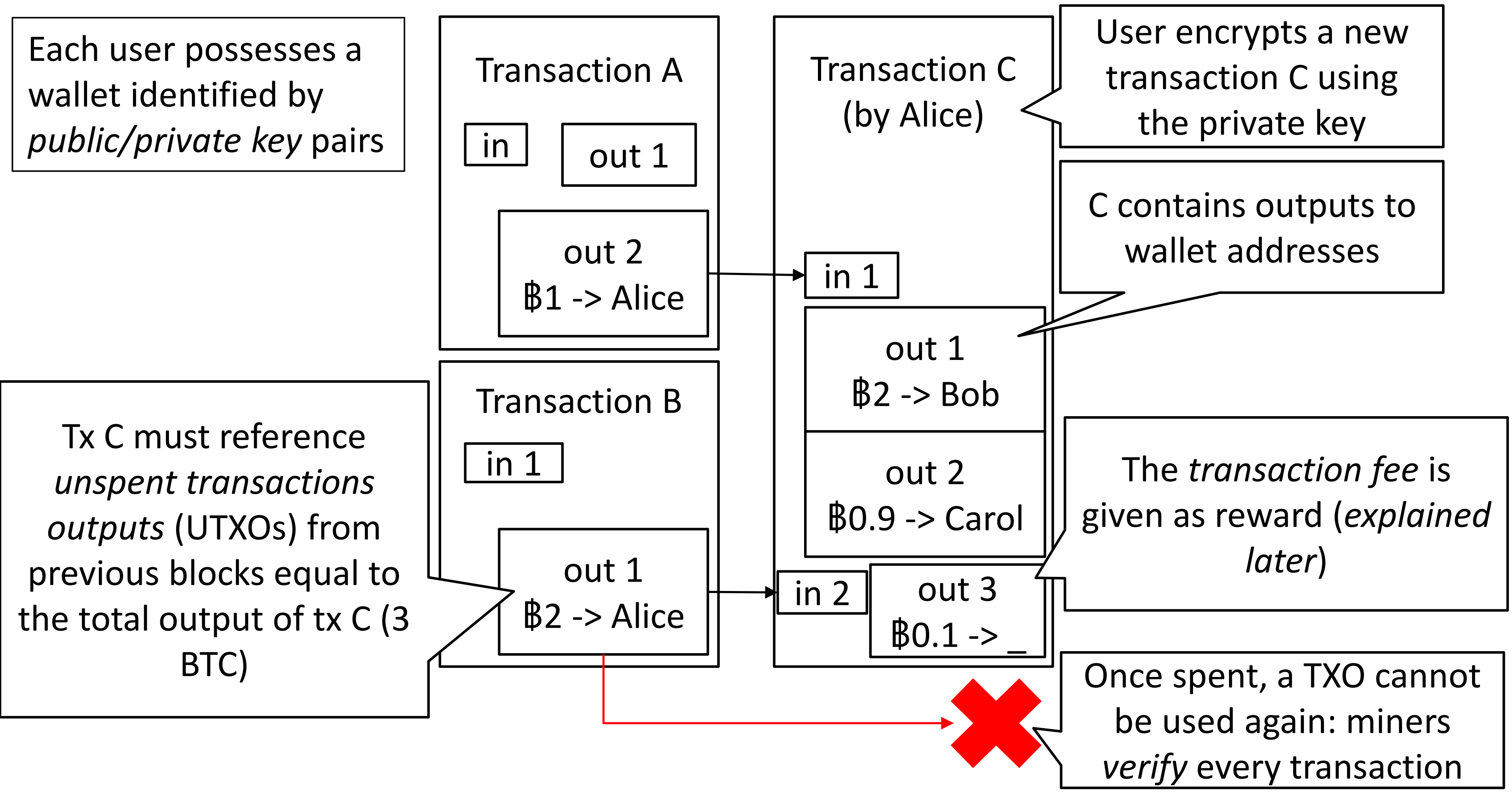



In the "Unspent Transaction Output" model, there is no balance or concept of account.

To spend money, we simply transfer a "check" from one person to another.

**Bitcoin uses this model!**

# Bitcoin Transactions



# Wallets and addresses

Users generates its own key pairs

- This includes any user, **including but not limited to** miners
- Uses ECDSA with 256 bits (Elliptic curve cryptography)

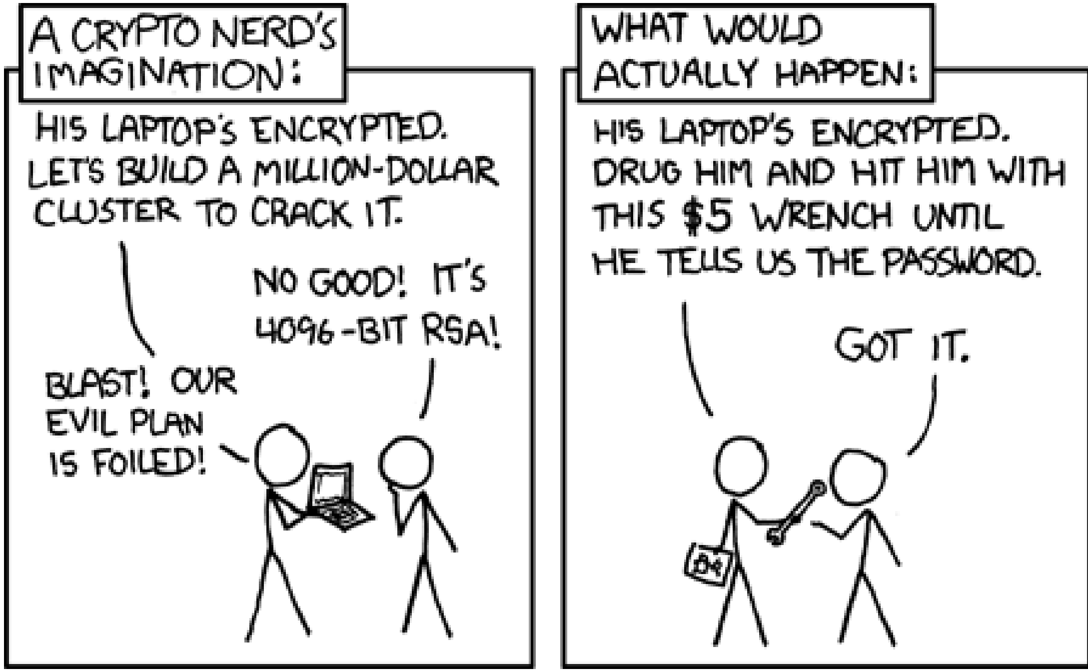
To receive bitcoins, a user will normally share an address

- This address is generated from its public key
- The user can claim a transaction output to an address by signing with the associated private key

Key pairs management

- Each user is encouraged to generate a new key pair per transaction
- A wallet is used to manage multiple key pairs
- Certain wallets can also generate key pairs (see **HD Wallet)**

# Wallet security





Losing your private key:

- Loss of private key means any UTXO to the associated address cannot be redeemed
- This money is essentially lost, thereby reducing the total amount of currency in Bitcoin
- Trusting an online service to store your private key is also risky, since there is no way to prove that you are the rightful owner if the key is stolen or misused
- The most reliable solution is to store your private keys on tamper-proof hardware wallets or to memorize them (e.g. using a *seed phrase*)

# Transaction Flow

Alice
(Sender)

Bob
(Receiver)

1. Bob generates and send a public key address.
2. Alice creates a transaction using this address.
3. Alice sends the new transaction to the network.
4. The transaction is broadcast using gossiping.
5. The transaction is included in a block.
6. Bob can verify the transaction is in the blockchain.
7. Bob can now sign new transactions which redeem this address.

**Transaction A**

in 1

out 1
฿1 ->
Bob.Address1

**Transaction B**

in 1
Bob.Address1

out 1

# “Smart contracts” in Bitcoin

A transaction output includes a verification script

- representing the conditions under which the output can be redeemed, i.e., included as an input in a later transaction
- A typical script: “can be redeemed by a public key that hashes to X, along with a signature from the key owner”

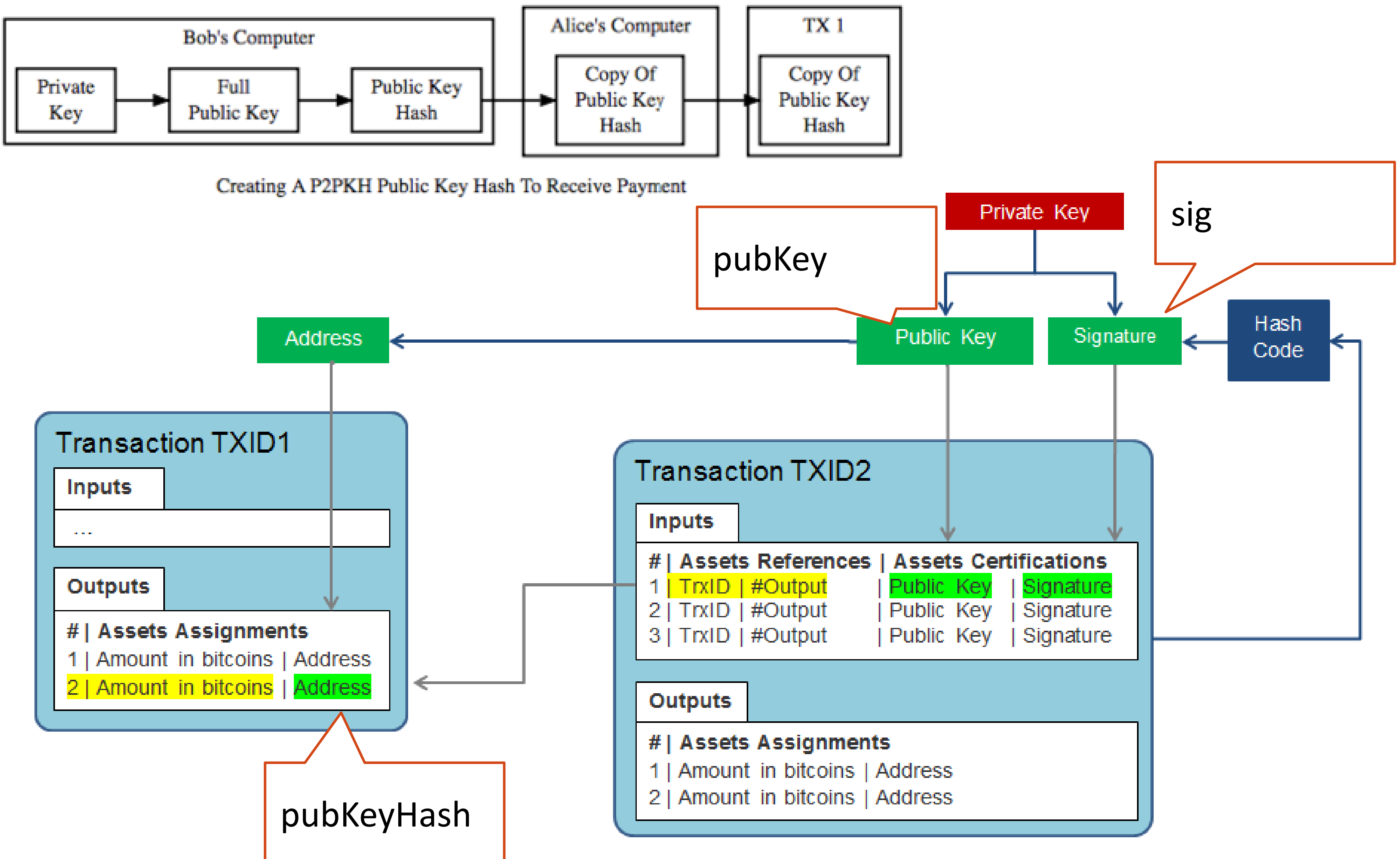
There is also a redeeming script attached to the input

Both scripts are executed by whoever verifies the redeeming transaction, such as a proposer

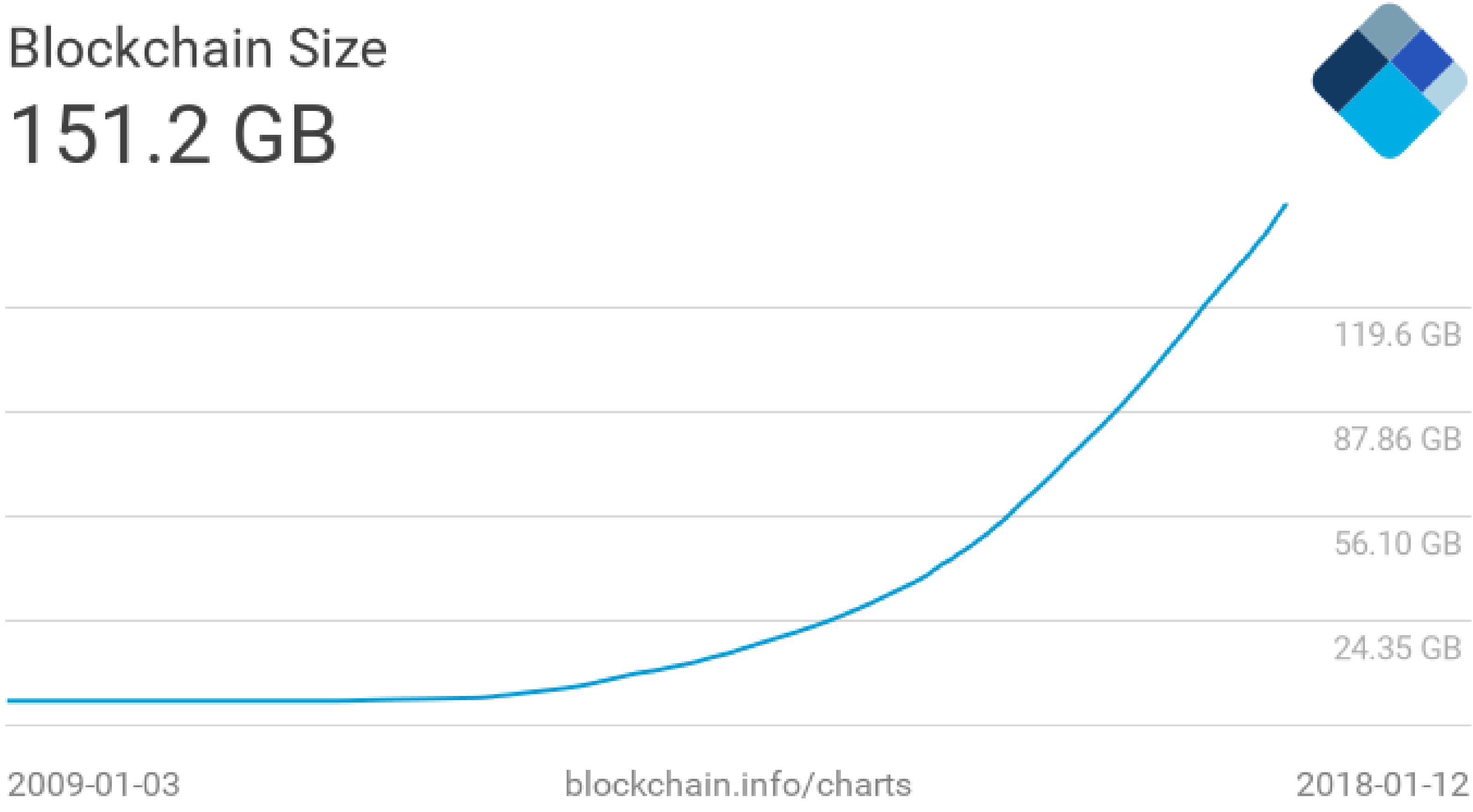
A script language with an order of 200 commands

- Support for cryptographic primitives

# Redeem a UTXO (P2PKH)

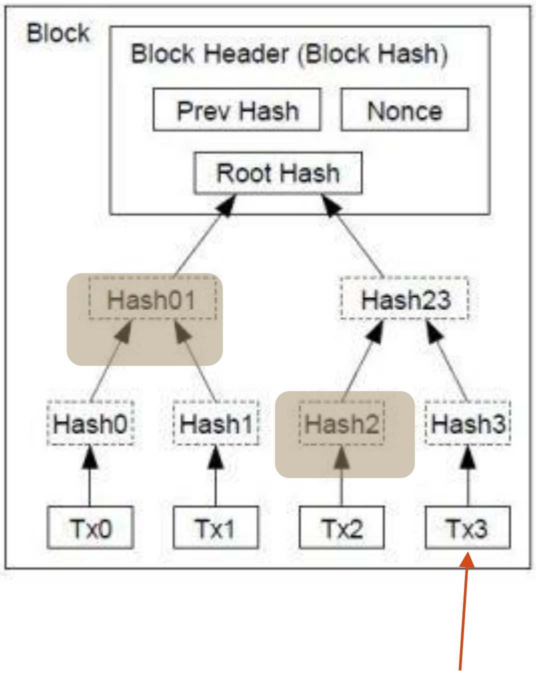


# Size of ledger: 219 GB (2019/06)



# Data Structure within a Block

**Merkle Tree**




- To avoid hashing the entire block data when computing PoW, only the *root hash* of the Merkle tree is included.
- For users without a full copy of the blockchain, *simple payment verification (SPV)* is used to verify if a specific transaction exists.
  - SPV users have a full copy of the block headers
  - A *Merkle proof* contains the transaction itself, all hashes to go up from the transaction to the root, e.g., Hash01, Hash2 (for Tx3).

*Presentation by Yahya Shahsavari*, PhD Student at ÉTS Montréal

# Networking

## GOSSIPING PROTOCOLS

# Analysis of Bitcoin

LIMITATIONS AND SOLUTIONS

# Low transaction throughput

Bitcoin has a max throughput of 7 transactions/second

- VISA Network: 2000 tps (average)

Two factors: block size (1 MB) and block time (10 minutes)

SegWit addresses the block size issue:

- Separates scripts and signatures from the block proper
- Increases the number of transactions per block

Slow block time:

- Ethereum uses a much faster time of 10-20 seconds
- But this increases the number of forks (concurrent proposed blocks)
- Ethereum uses a different consensus protocol

Other solution: Lightning network

- Layer 2 microtransactions
- Periodic settlement on the blockchain

# Hard/soft forks

Updates to the code cause forks

To preserve backward compatibility, soft forks cannot make drastic changes to the code

- C.f. the complexity of SegWit and its limited impact

If not possible, a hard fork is created

- This duplicates the money prior to the fork

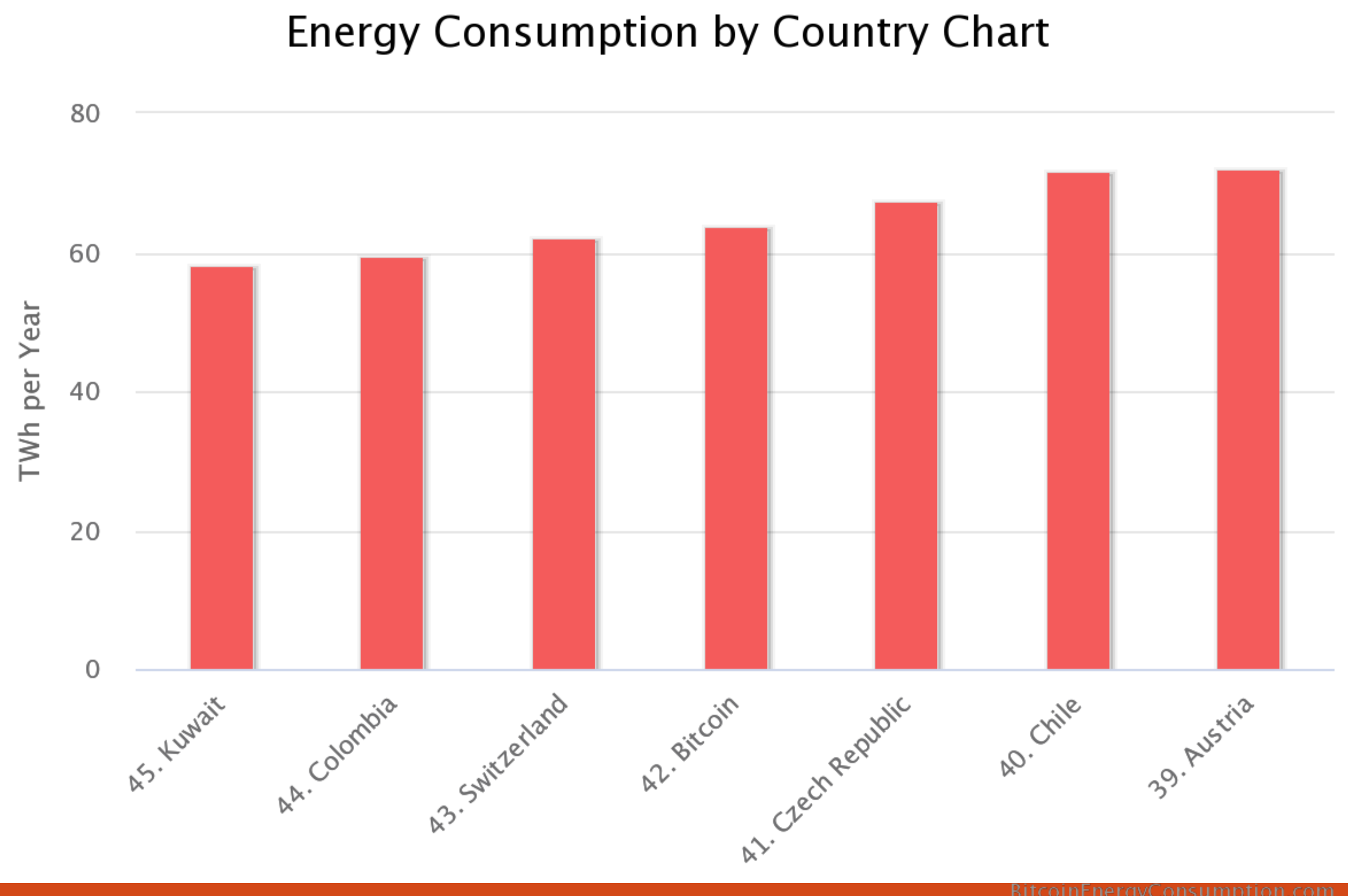
There exists over 13700 cryptocurrencies

- Many are forks of the original Bitcoin

# Energy consumption of PoW

Environmental impact: ~1000x more energy than credit card

Currently 43th in energy consumption (comparable to Switzerland)



# Alternative: Proof-of-Stake

Simple PoS solution:

- *sha256(PREVHASH + ADDRESS + TS) <= 2^256 * BALANCE / DIFFICULTY*
- ADDRESS of wallet of the miner, BALANCE is the recorded stake for the wallet
- TS is the timestamp in UNIX time (seconds)
- Thus, only one hash needed per second (per wallet)

Branches can still exist in PoS:

- Due to propagation delays, multiple timestamps are valid for a block
- The puzzle function does not return an unique winner

Nothing-at-Stake problem:

- PoW: cannot mine parallel branches since splitting resources is not effective
- PoS: mining parallel branches is easy since it only requires 1 hash/s
- Slasher algorithm: detection of parallel mining confiscates the stake

# “Meaningful” PoW

Primecoin

FoldingCoin FLDC
www.foldingcoin.net
Mine Medicine, Not Hashes

FoldingCoin

Mine Medicine, Not Hashes

# Variance in mining rewards

Current global hash rate: 48,000,000 TH/s

- Expected time to block for a single GPU: 7 million years!

Solution: pools allow miners to combine their hashing power

- Reduces variance
- Miners must trust the pool operator to divide the rewards **fairly**

Solution: Share-based mining

- Miners submit shares with low difficulty to prove their hash rate
- Divide the rewards based on shares: PPS, Score-based, etc.
- Attacks possible: lie-in-wait, block withholding…

Centralisation of mining power

- Threat of 51% attacks
- Other attacks possible with less power (e.g. selfish mining)

# Blockchain Systems

ETHEREUM

HYPERLEDGER

ETHEREUM

Managing entity: Ethereum Foundation

- Major players: Deloitte, Toyota, Microsoft, …

Focus: Open-source, flexible, platform

- Cryptocurrency: 1 Ether = 1e18 Wei (502 USD, 2018/04)
- Smart contracts: Solidity, Remix (Web IDE), Truffle (Dev./Test), *Vyper*
- Ethereum Virtual Machine (EVM), Ethereum Web Assembly (eWASM)
- Permisionless (public) ledger: Proof-of-Work, *Proof-of-Stake (Casper)*

Notes

- DOA Event: $150 million lost, hard forked into Eth. Classic
- GHOST Protocol: Merging of branches (uncle blocks)
- Ethash: Memory-hard hashing protocol which is ASIC-resistant
- *Scalability: L1 Sharding and L2 Plasma*

# Smart Contracts

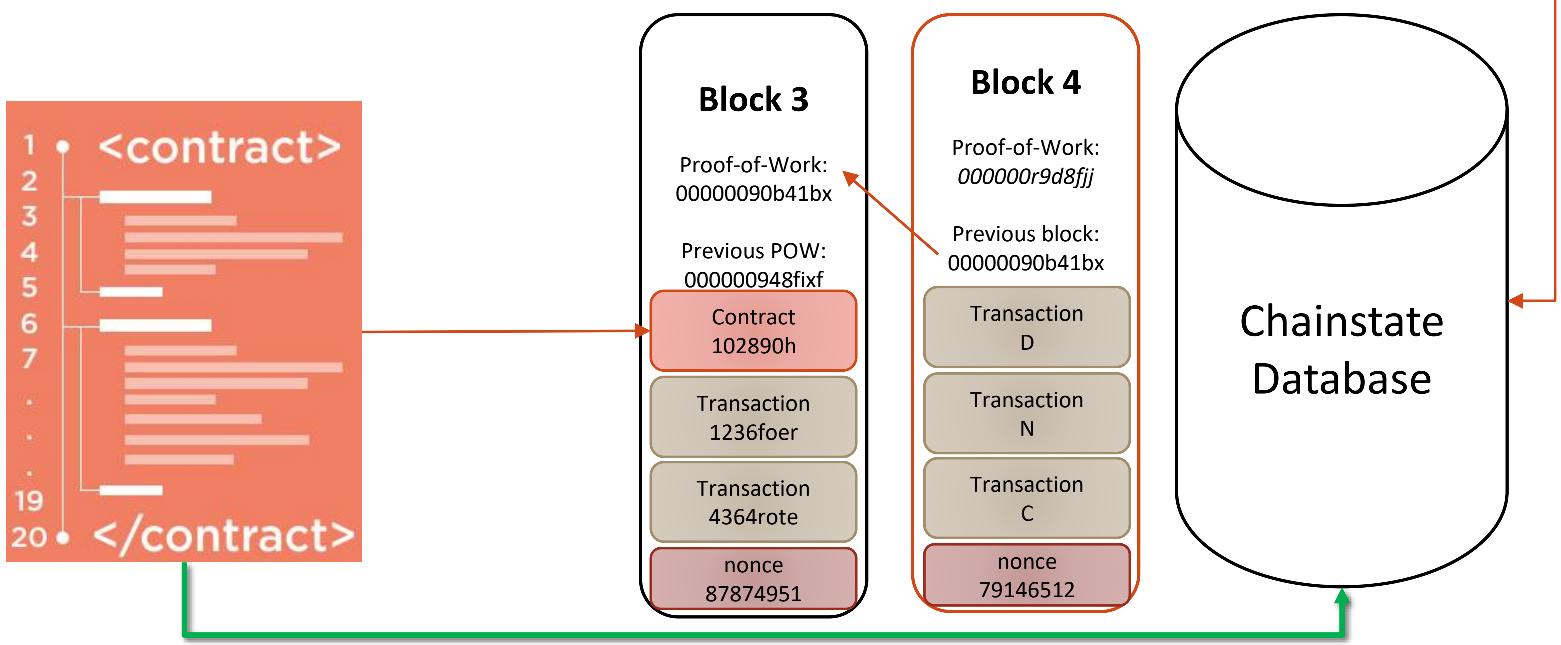
- Contracts contain *executable bytecode*
- Created with a blockchain tx
- Contracts have internal storage

Contracts execute when triggered by a transaction (or by another contract)

Execution time is limited by *gas*

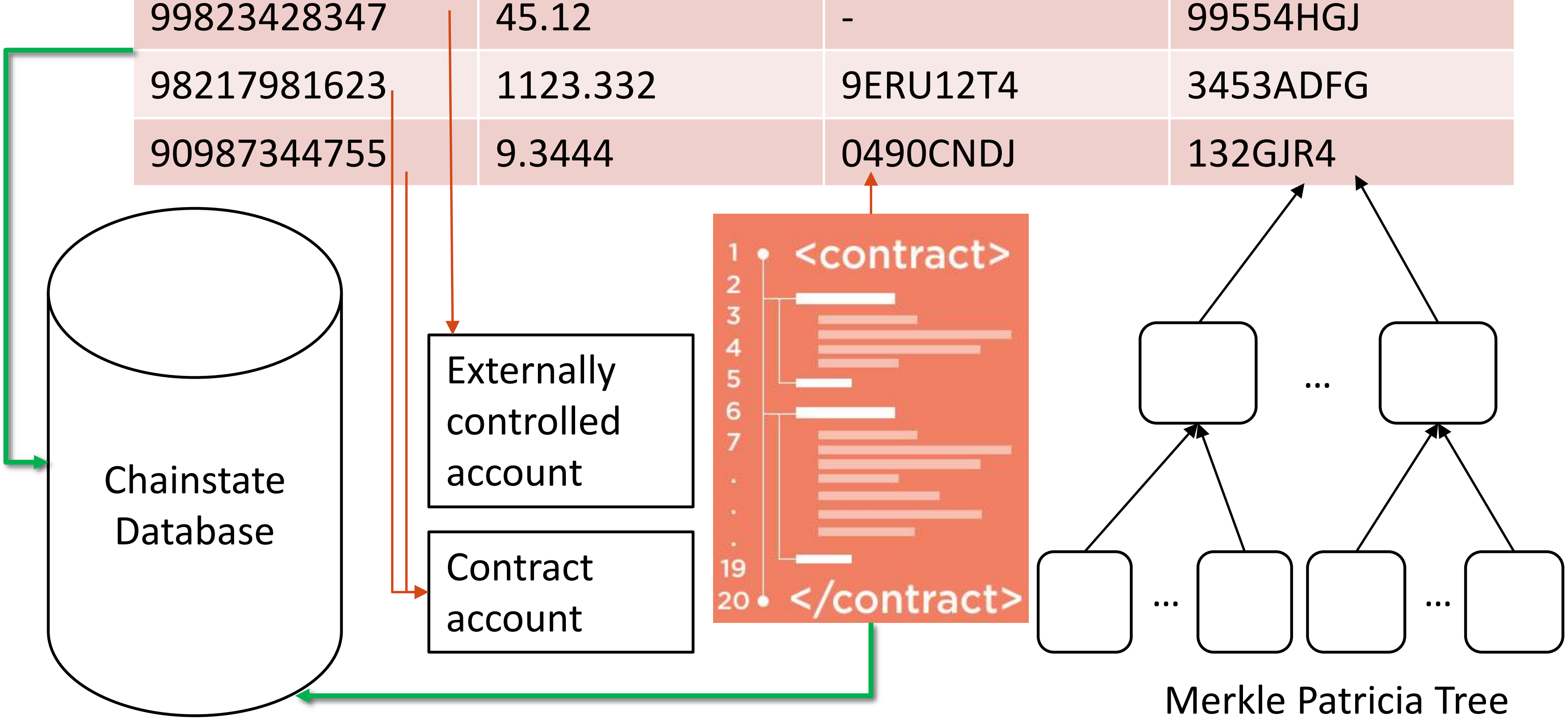
*Example: Land registry*

| Wallet ID | Held Titles |
|---|---|
| 99823428347 | 34356,324324 |
| 98217981623 | 677343,4444 |
| 90987344755 | 994,38842,439 |

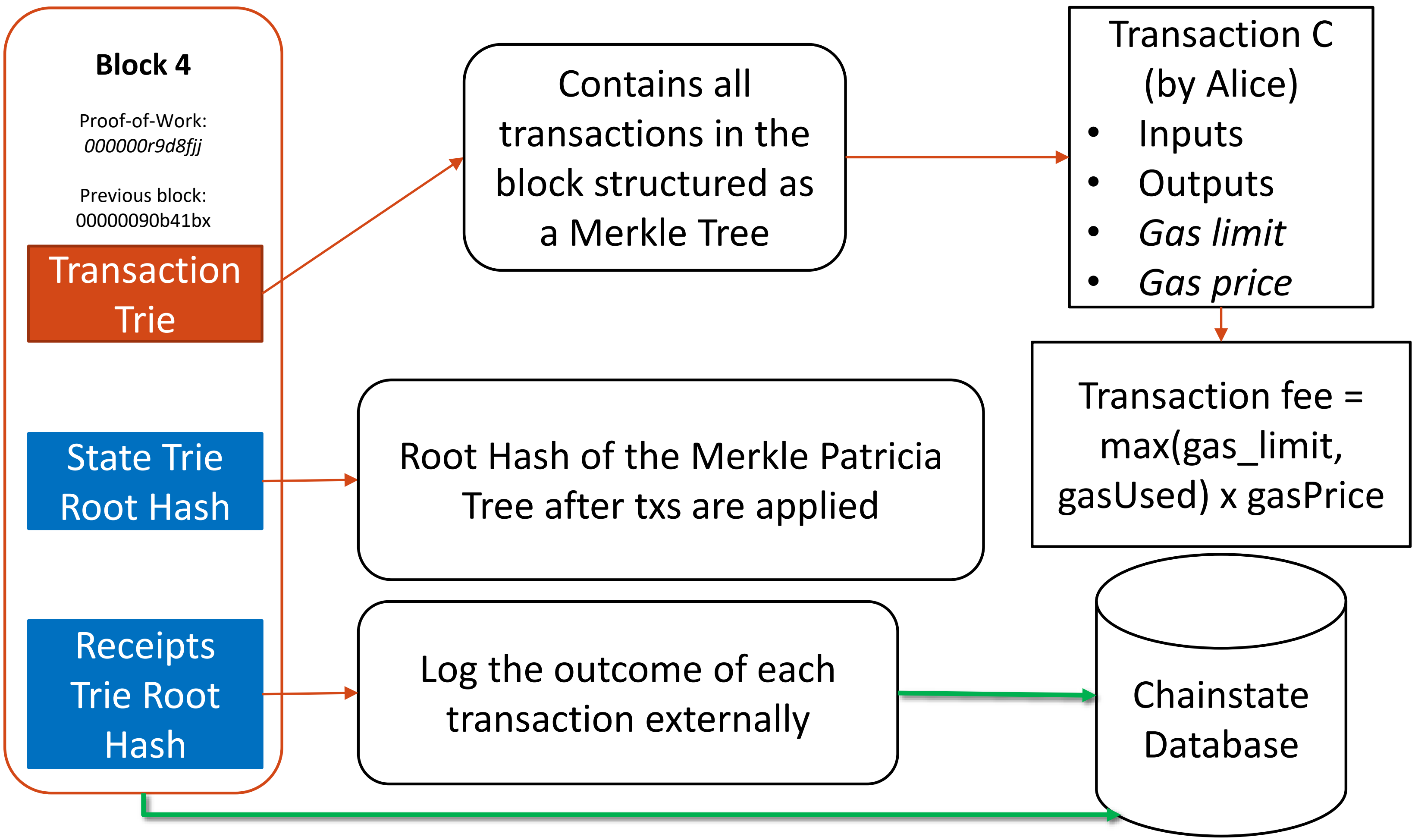


# Account State ("World State")

| Wallet ID | Balance | Code Hash | Internal State |
| --- | --- | --- | --- |
| 99823428347 | 45.12 | - | 99554HGJ |
| 98217981623 | 1123.332 | 9ERU12T4 | 3453ADFG |
| 90987344755 | 9.3444 | 0490CNDJ | 132GJR4 |

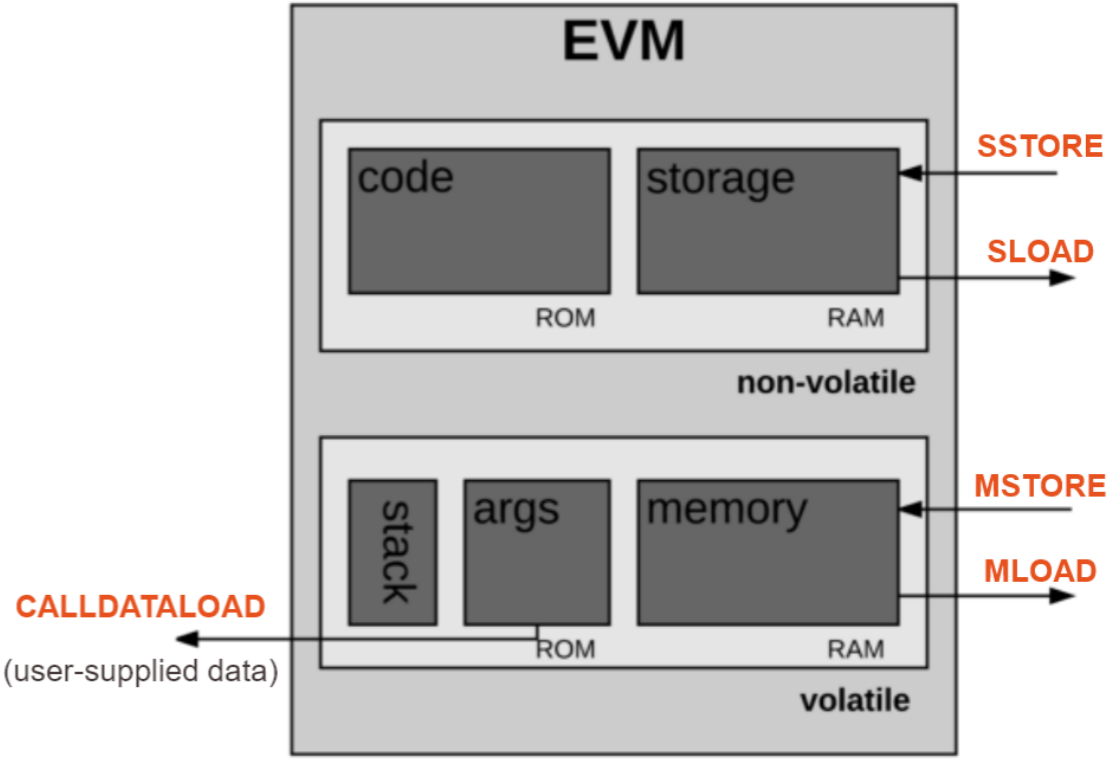


# Execution and Mining



# Ethereum Virtual Machine

| Architecture | | |
|---|---|---|
| Stack machine | | |
| Turing complete | | |
| Instruction set | ~180 Opcodes | |
| Memory type | | |
| Stack | volatile | byte-array (list []) |
| Memory | volatile | byte-array (list []) |
| Storage | persistent | key-value database (dictionary {}) |

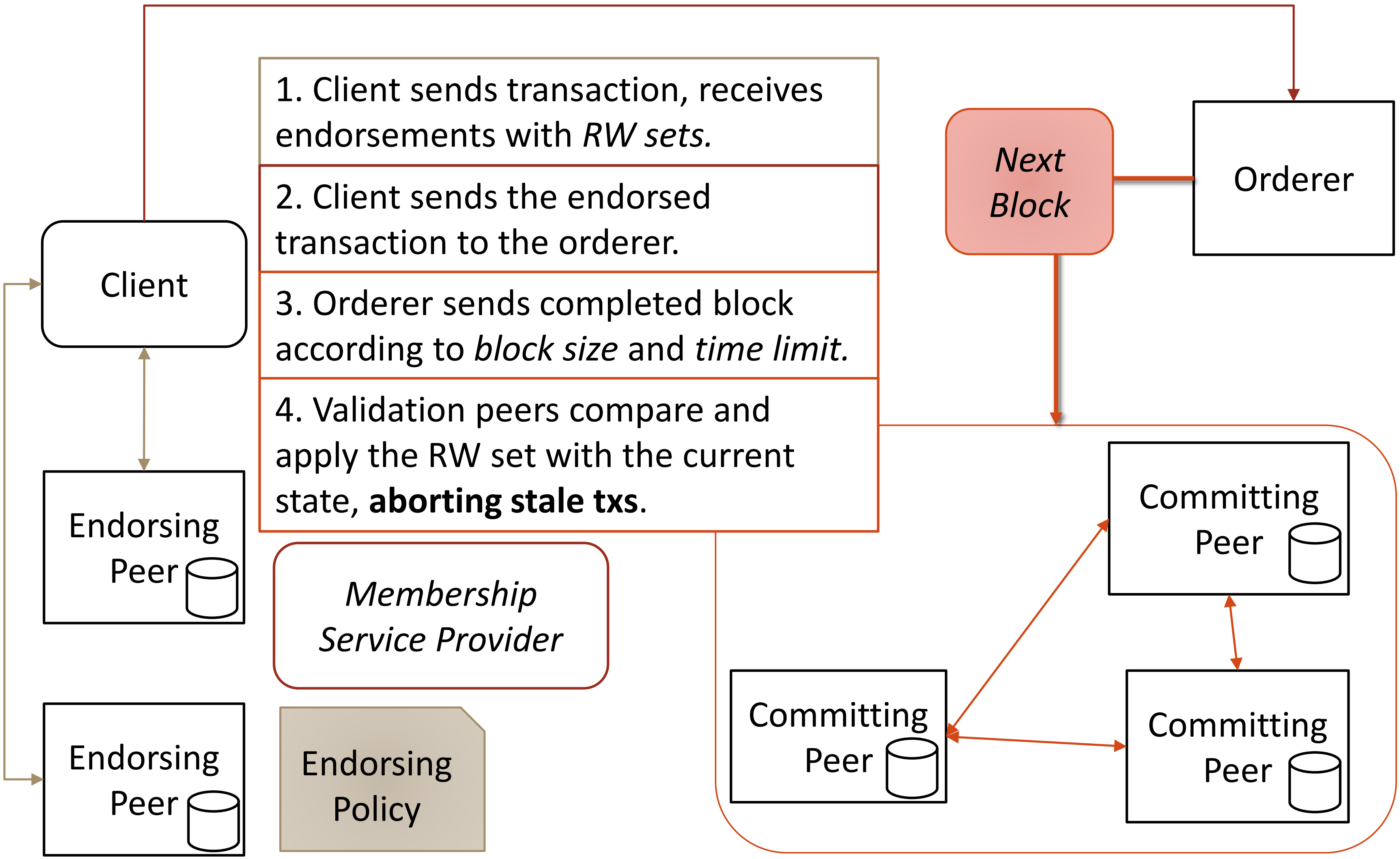


# Comparison with Bitcoin

| | **Bitcoin** | **Ethereum** |
|---|---|---|
| Transactions | Transfer of bitcoins | *Contract creation*, transfer of ether, *contract calls, internal transactions* |
| Accounts | User wallets | Externally owned accounts, *contract accounts* |
| Transaction fees | Amount specified by sender | Gas calculated using sender's values |
| Block content | Transactions trie | Transactions, *State Root Hash, Receipts Root Hash* |
| Chainstate Database | UTXO Model | World state, balance, *receipts, bytecodes for contracts* |
| Querying | Simple Payment Verification | Merkle proofs for *events*, transactions, *balance*, etc. |

- Managing entity: Hyperledger Consortium
  - Major players: IBM, NEC, Intel, R3, …
- Focus: Enterprise blockchains
  - Permissioned ledger (private/consortium network)
  - Open-source
  - World state on CouchDB/LevelDB, event listener
  - Membership service provider, access control, channels
- Projects
  - Fabric: Execute-Order-Validate transaction processing
  - Sawtooth: Proof-of-Elapsed-Time (using Intel SGX)
  - Composer: Smart contract language and development tool
  - Cello: Blockchain-as-a-Service framework
  - R3 Corda: Financial applications

# Fabric: Transaction processing flow

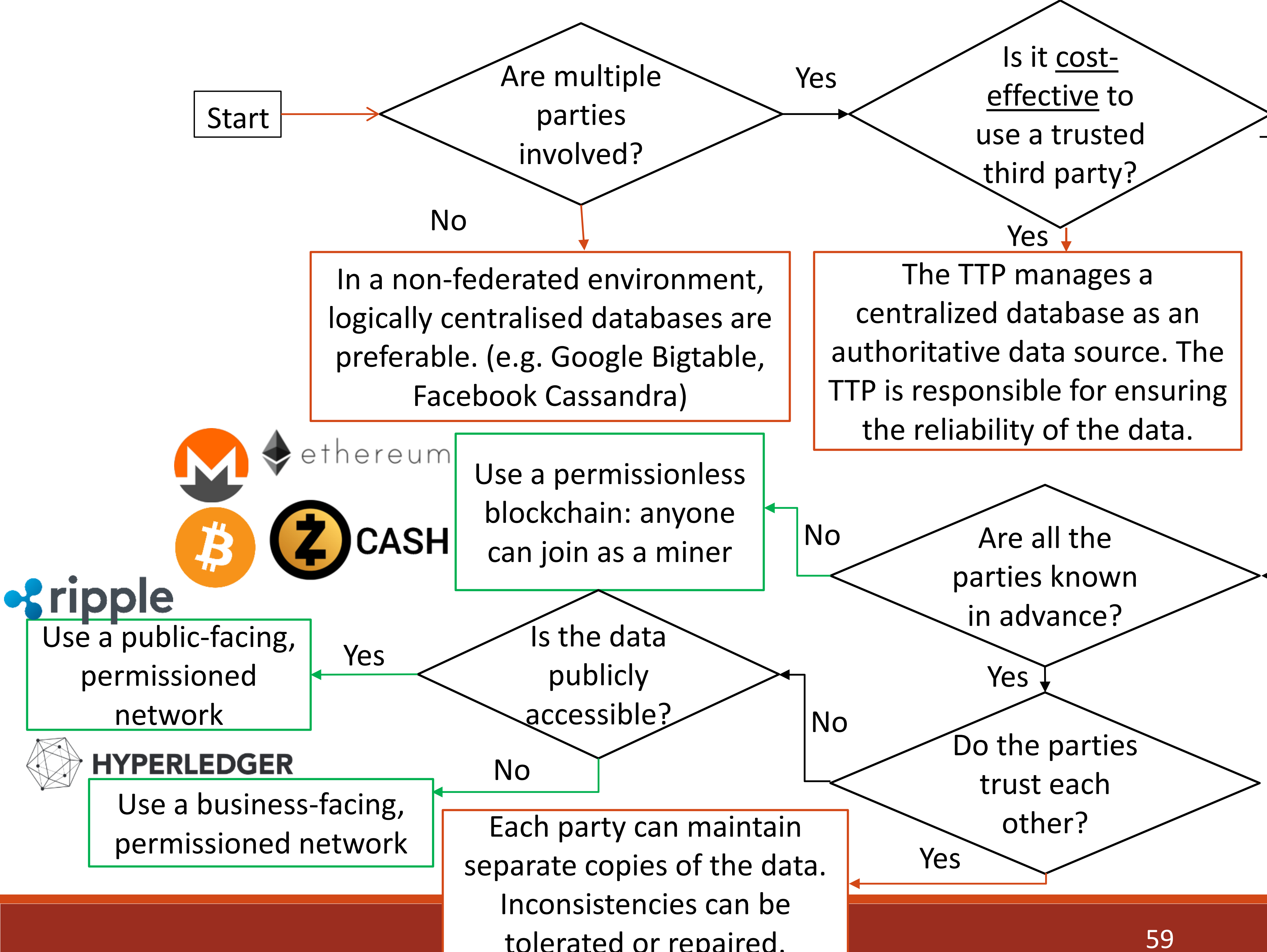


# Blockchain Insights

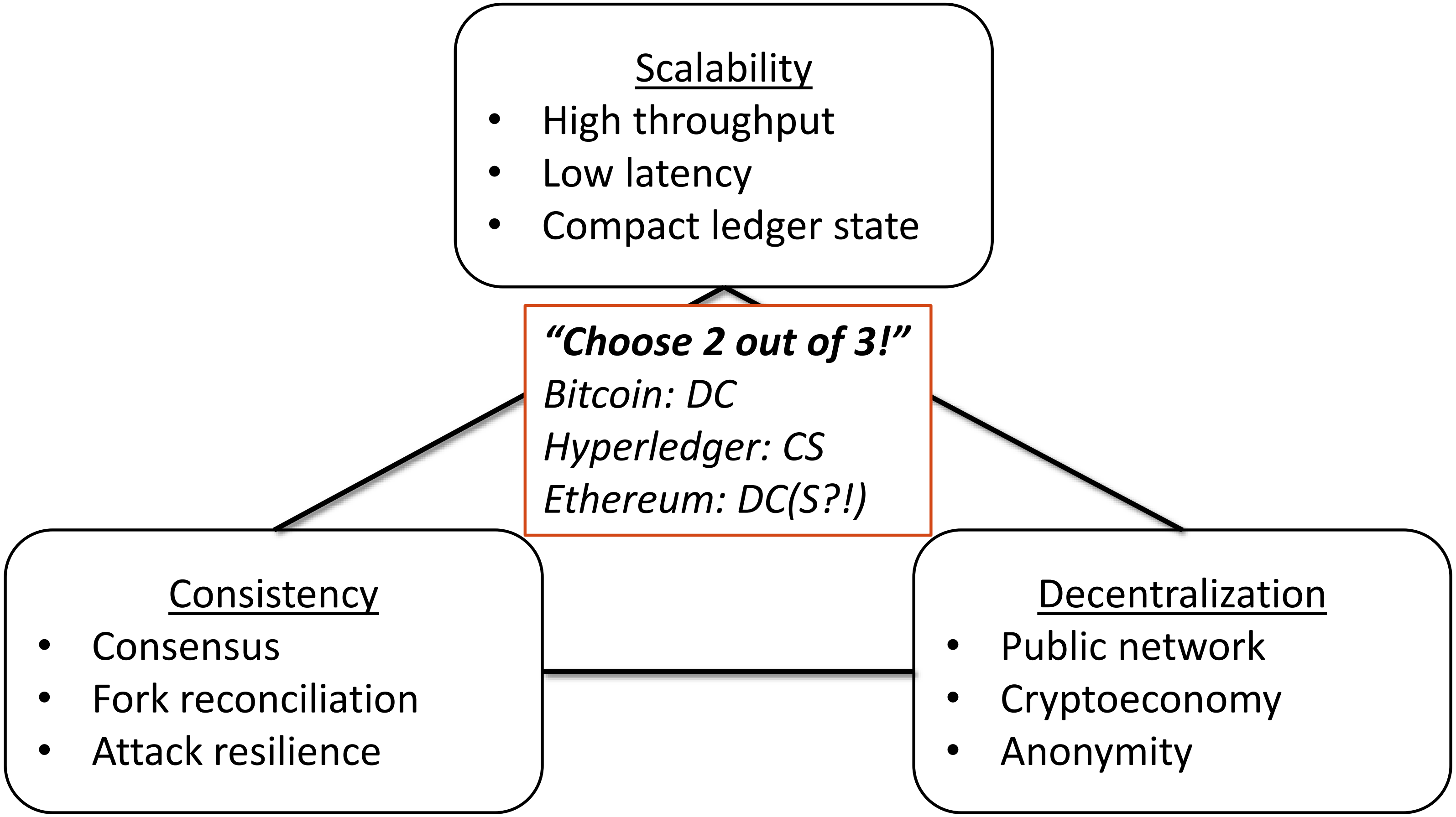
BENEFITS AND CHALLENGES

TAXONOMY OF BLOCKCHAINS

RESEARCH OPPORTUNITIES


Start
Are multiple parties involved?
No
In a non-federated environment, logically centralised databases are preferable. (e.g. Google Bigtable, Facebook Cassandra)
Yes
Is it cost-effective to use a trusted third party?
Yes
The TTP manages a centralized database as an authoritative data source. The TTP is responsible for ensuring the reliability of the data.
Are all the parties known in advance?
No
Use a permissionless blockchain: anyone can join as a miner
ethereum
CASH
ripple
Yes
Do the parties trust each other?
Yes
Each party can maintain separate copies of the data. Inconsistencies can be tolerated or repaired.
No
Is the data publicly accessible?
Yes
Use a public-facing, permissioned network
No
Use a business-facing, permissioned network
HYPERLEDGER

# “CAP Theorem” for DLTs



# DCS Conjecture

Safe and verifiable smart contracts
Attacker models: <51% attacks
Security of off-chain services (e.g. exchanges)
"Garbage in, garbage out": IoT barrier

Incentives, mining rewards
Privacy: Anonymity, fungibility
Endorsement policies, governance
Selective replication: State channels

Consistency

Sharding, sidechains, tree-chains, …
Large-scale chainstate storage
Big Data analytics
Layer 2 Network: Lightning, Raiden
Proof-of-Stake, POET, PBFT, …

Decentralization

Scalability

"Choose" 2 out of 3!

Bitcoin: DC
Hyperledger: CS
Ethereum: DC*(S?!)*

Investigate **potential use cases**
Choose and **tune** the right platform
Develop **reusable middleware**

## Applicability of blockchains

- DCS: May lead to fundamental research
- Applications: mostly 3.0, and some 2.0
- Layers: application, modeling, contract

## Blockchain middleware

- Applications: 1.0 – off-chain exchanges and payment networks, 2.0 – reusable online services, 3.0 – data integration, analytics
- Layers: contract

## Security and privacy

- DCS: +DC, -S
- Applications: 1.0 –transactions, 2.0 – smart contracts, 3.0 – data privacy
- Layers: contract, system, data, (network)

## Scalable system innovations

- DCS: +S, -DC
- Applications: 1.0 – incremental, 2.0 – public smart contracts, 3.0 – clean slate designs
- Layers: system (consensus), data

# Blockchain 1.0: Currency


Over 13700 public cryptocurrencies available!

# Research for 1.0 Apps

Formally analyze the *security* model of Bitcoin

- 51% attack
- DoS attacks on: mining pools, currency exchanges, …

Conduct *performance modelling*

- Simulate various Bitcoin scenarios
- Understand impact of network topologies (e.g. partitions)

Develop *scalable* mechanisms with *legacy support* to maintain the *sustainability* of Bitcoin

- SegWit2x
- Bitcoin-NG (NSDI '16)
- Off-chain (Lightning network)
- Algorand (SOSP '17)

# Blockchain 2.0: Decentralized Apps

ĐApps are applications built on blockchain platforms using smart contracts (e.g. Ethereum)

Đapps

ETHEREUM

GNOSIS

Forecast market (e.g. betting, insurance)

EtherTweet

Decentralized Microblogging

Token Distribution

Crowdfunding

alice

Charity donation payment

# Research for 2.0 Apps

Formal *verify* smart contracts, detect and repair security flaws

- Ethereum Viper

Develop *scalable consensus* mechanisms which support *smart contracts* in an *public* network (w/ *incentives*)

- Proof-of-Stake (Casper)
- Side-chain (Plasma)
- Sharding (ShardSpace)

Develop *efficient data storage* techniques to store *smart contracts* and the *chainstate*

- AVL+ (Tendermint)
- Merkle Patricia Trees (Ethereum)
- Zero-Knowledge Proofs: zk-SNARK

# Blockchain 3.0: Pervasive Apps

everledger

Diamonds Provenance

Applications involve entire industries, **public sector**, and IoT.

FACTOM

Land Registry in Honduras

BlockchainHealth

Electronic Health Records

VOTEWATCHER

Transparent Voting System

# Killer app: Supply chain management?

MAERSK TRADELENS

Containers shipping

IBM Food Trust™


Food crates

# Research for 3.0 Apps

Develop *"clean-slate"* scalable distributed ledgers:

- Permissioned ledgers (Hyperledger Fabric)
- Blockless DLTs (IOTA Tangles, R3 Corda Notaries, Hashgraph)

Develop *blockchain modelling tools and middleware*

- BPMN, Business Artifacts with Lifecycles, FSM
- Authentication, reputation, auction, voting, etc.

Support strict *governance, security, and privacy* requirements

- State channels
- Endorsement policies

Overcome the *cyber-physical barrier for data entry*:

- Object fingerprinting
- Secure hardware sensors

# IBM Verifier