Advanced Topics of Software Engineering (ASE)

Chapter 3. Software architectures and their trade-offs

Prof. Dr. Florian Matthes, Prof. Dr. Alexander Pretschner

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Software architectures and their trade-offs

A technical paper published online in the year of the financial crisis

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto

October 31, 2008

Abstract

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions,

http://nakamotoinstitute.org/bitcoin/

What is a „Blockchain“?

Technical definition

“A blockchain […] is a **distributed database** that maintains a continuously-growing list of ordered records called blocks. Each block contains a timestamp and a link to a previous block. **By design** blockchains are **inherently resistant to modification** of the data: once recorded, the data in a block cannot be altered retroactively.”

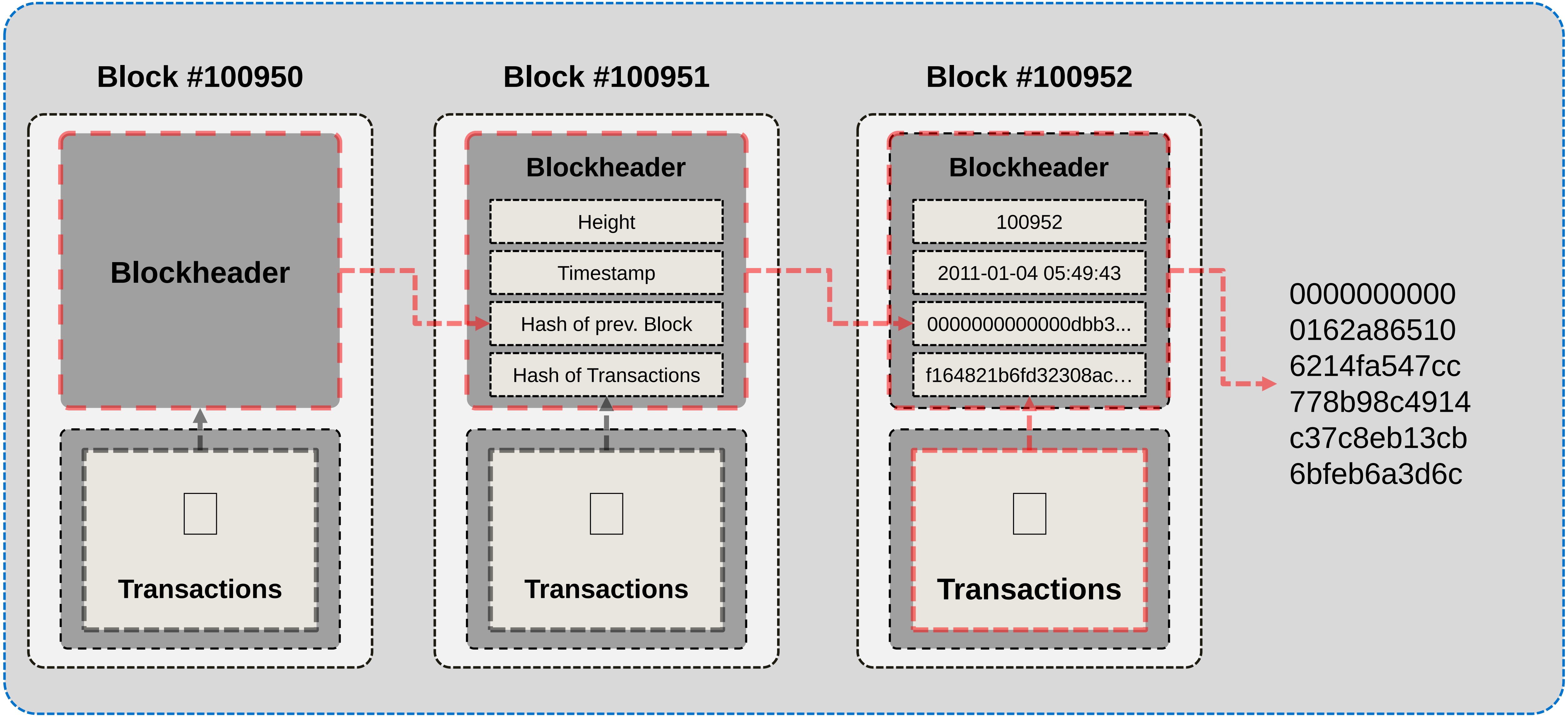
https://en.wikipedia.org/wiki/Blockchain_(database)

Functional description

[...] are **systems** that enable parties who don’t fully trust each other to **form and maintain consensus** about the **existence, status and evolution** of a set of shared facts.

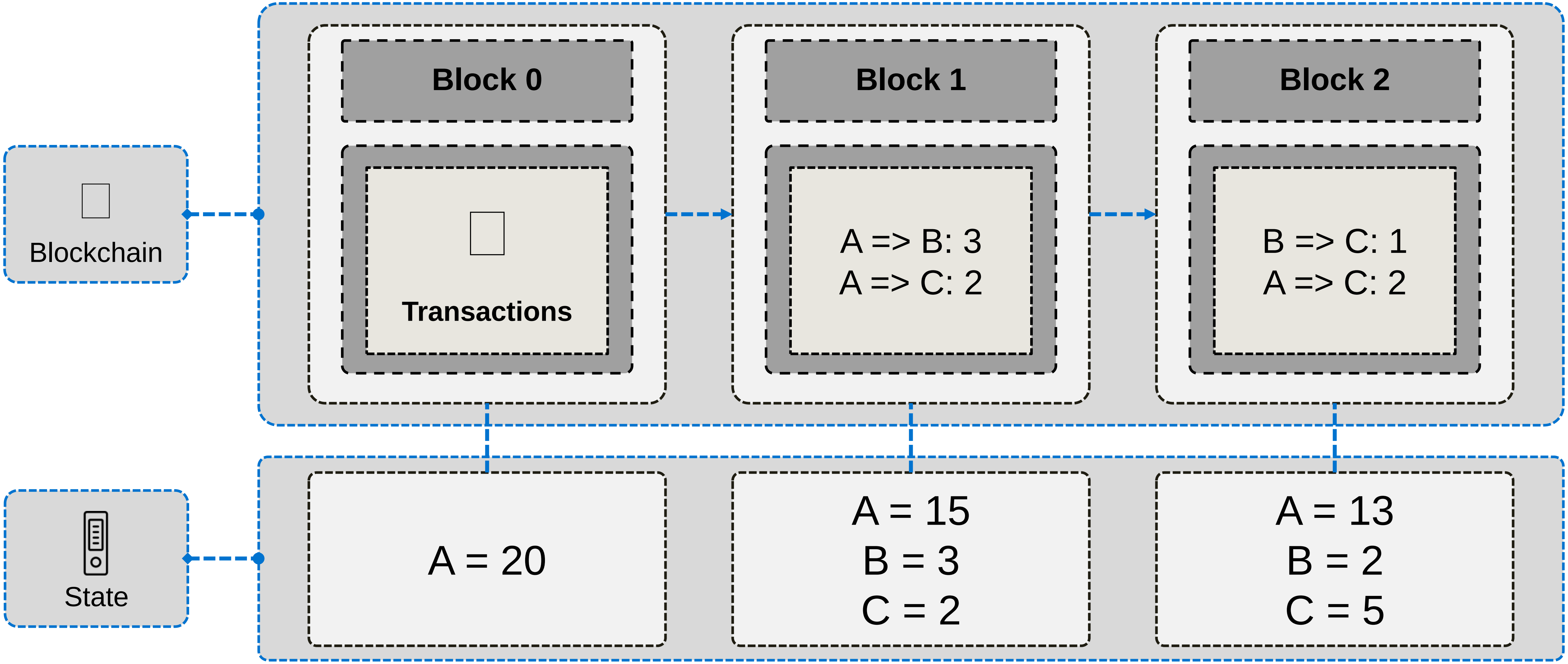
Richard Brown, R3 CTO

The structure of the Bitcoin blockchain



[1]In Bitcoin, the height is included in the Coinbase Transaction.

The blockchain records a strict sequence of transactions.
The state modified by the transactions is **not** stored on the blockchain.



Roles in the blockchain network

Wallet Owner

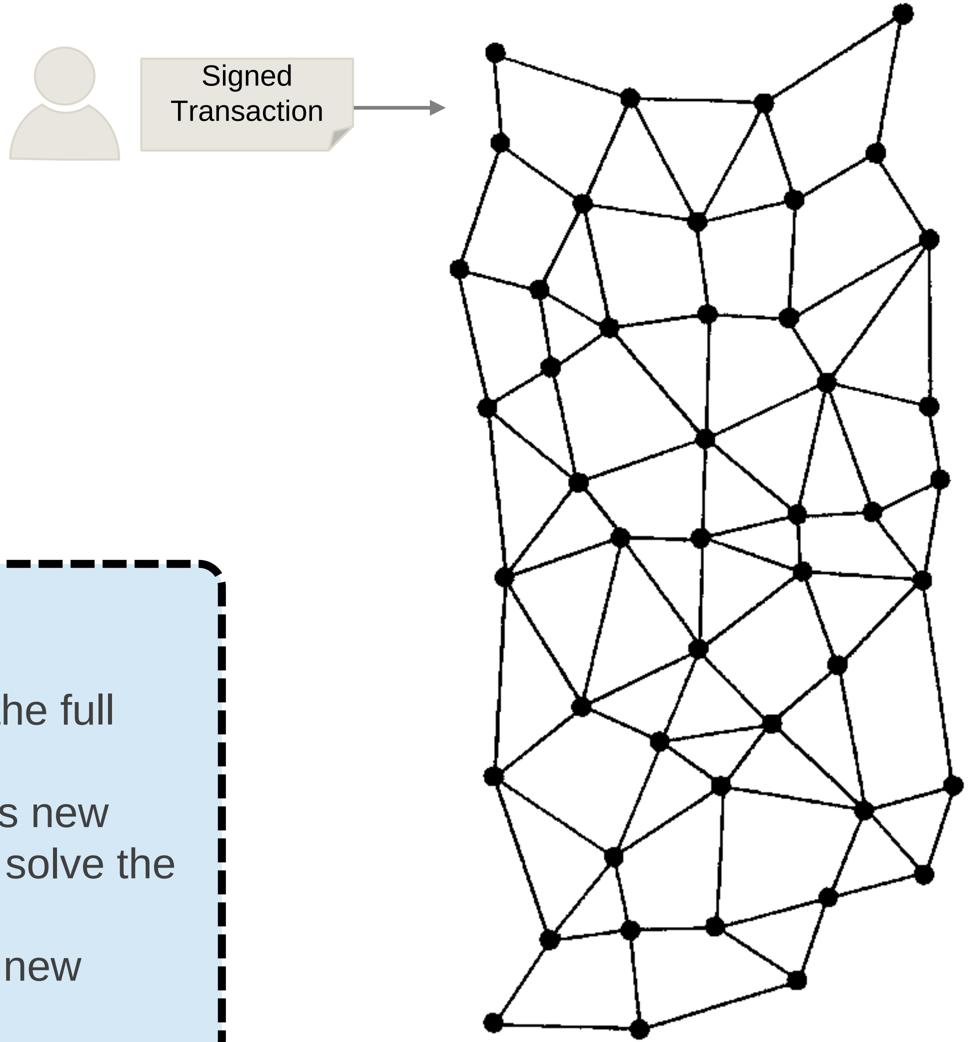
- Has private key to unspent transactions
- Owns the money
- Sends money by singing and publishing new transactions

Full Node

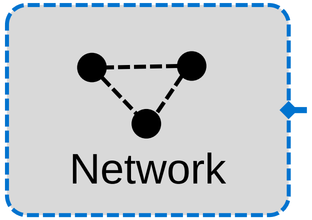
- Maintains the complete blockchain
- Validates every transaction and block
- Relays all new transactions

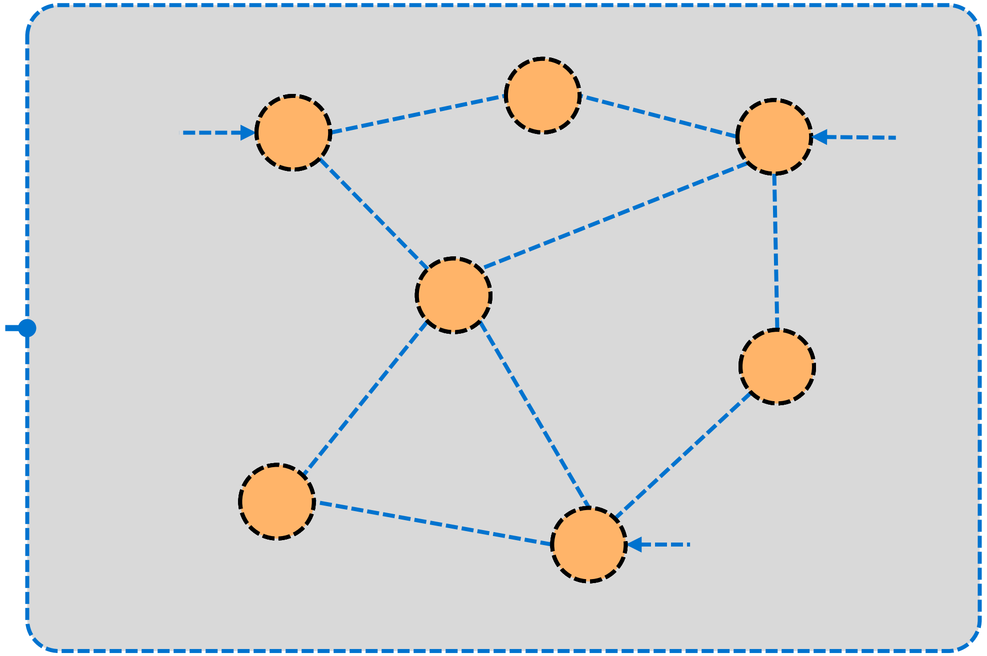
Miner

- Acts the same as the full node
- Additionally creates new blocks and tries to solve the mining puzzle
- Gets rewarded for new blocks

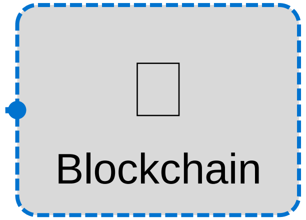


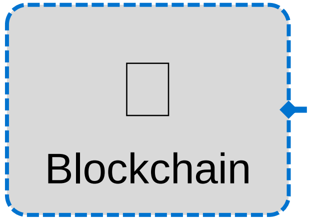
Reaching consensus in the network

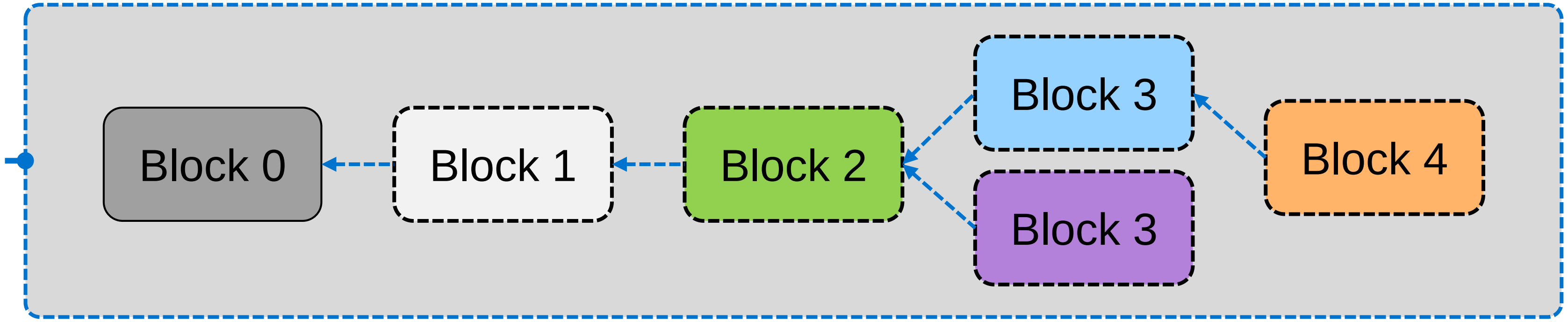




1. Only valid blocks
2. Longest chain wins
3. First arriving block first







Which node is allowed to issue a new block?

Constraints

- Keep the network fully decentralized ➔ Random selection of node

- Avoid a **51% attack** by a group of nodes allowing them to
  - prevent new transactions from gaining confirmations, to halt payments between some or all users, or
  - reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins.

- Avoid a **25% economic attack** by a group of nodes that change the incentives for other nodes so
  - they are incentivized to join the group of the attacker nodes or no longer work as a node

Approaches

- Proof of work ➔ Increase cost / difficulty of creating new blocks
  - Solve a time-consuming mathematical puzzle
  - Solve a puzzle that requires a lot of computer memory

- Proof of stake ➔ Increase cost of creating invalid blocks
  - Deposit an amount of money that gets transferred if an invalid block is detected

Comparison with established centralized solutions

Advantages

- Decentralized management (trust in one party ➔ trust in a system of multiple parties)
- Transparency of all transactions
- Traceability of the complete transaction history
- Pseudonymity of the wallet owners
- *Built-in* financial incentives for early adopters and network growth (➔ business model)

Opportunities

- Innovation thrust for IT solutions in the global finance system
- Lowered entry barriers for IT-savvy players with limited financial resources
- Impetus to re-evaluate established business models and economic mechanisms

The potential of blockchain technologies

They enable **intermediary-free** transactions of **digital, non-copyable goods** without the need to trust the other party.

Digital identities of people or **machines** can enter into secure transactions and all transaction details are stored **immutable** and **decentralized**.

Automated, programmable contracts can ensure contract compliance.

"We believe that the Blockchain will have the greatest influence on contracts, logistics and supply chain, healthcare, public administration, asset clearing, property and transactions."

\- Greg LaBlanc

Berkeley UNIVERSITY OF CALIFORNIA

Current and future use cases (B2B & B2C)

Industries

Finance

- Replacement of intermediaries in transactions
- Crypto currencies
- Microcredit and crowdfunding peer-to-peer
- ICOs: Start-up financing

Pharma und Medical

- Access to decentralized patient records
- Prevention of prescription abuse
- Prevention of counterfeit drugs

Automotive

- Supply chain tracking
- Digital identity of a car
- Digital mobility solutions (e.g. car sharing)

Energy

- Private suppliers of electricity
- Micro power networks
- Electricity trading

Cross-industry, peer to peer

Documentation

- Elimination of some notary services
- Traceable supply chain
- Service & maintenance protocols
- Digital certificates of origin

IP Management

- Preservation of patents, art or ideas on the blockchain (incl. time stamp)
- Securing 3D printer models
- Direct remuneration of license holders

Digital Identity

- Refugees pay by retina scan
- Direct remuneration of license holders
- Certificates of origin of digital certificates

Sharing Economy

- Transactions without a central platform provider
- Decentrally documented bartering
- Pay-as-you-use insurances

Historical development of the concept of *Smart Contracts*

- The term “Smart Contract” was coined long before blockchain technology emerged (Szabo, 1994)
- Initially described the formalization of processes in public networks like the Internet
- Possible applications:
  - DRM
  - Payment
  - Connection to the “real world” through sensor and actuators

➔ However, there was a lack of technology to realize these ideas

Szabo, Nick (1994): Smart Contracts.
https://web.archive.org/web/19971008001022/http://www.virtualschool.edu/mon/Economics/SmartContracts.html

Smart Contracts on blockchain-based platforms

In 2015, Vitalik Buterin revived the idea in his white paper “*Ethereum: A Next Generation Smart Contract & Decentralized Application Platform*“

Idea

- Replace the fixed data structures, algorithms and protocols of individual blockchain solutions (e.g. Bitcoin, voting, bidding, lottery, proof of ownership, …) by programs, written in a domain-specific programming language (e.g. **Solidity**) on top of a single public blockchain (**Ethereum**) and currency (**Ether**).
- Use cryptography to secure the immutability of the program code (**contract**)
- Wallet owners agree on the contract(s) to be used for their future interactions
- No or very limited access of the code to the “real world” (sensors, actuators)

➔ Similar to data-centric architectures for workflow management and automated case management

Smart Contracts

To be clear, at this point I quite regret adopting the term "smart contracts".
I should have called them something more boring and technical, perhaps something like "persistent scripts".

Vitalik Buterin, 2018

Human-readable Smart Contracts

Who should be able to understand or create Smart Contracts?

- There are attempts to link the immutable code with a human-readable, structured description of the contract
- Such document should describe:
  - Intention
  - Content
  - Context
  - Boundary conditions
  - Units (e.g. currencies)
  - Parameters
  - Input / Output
  - …

➔ Comparable to a requirements specification

The combination of machine-readable and human-readable contract is called a “Ricardian Contract”

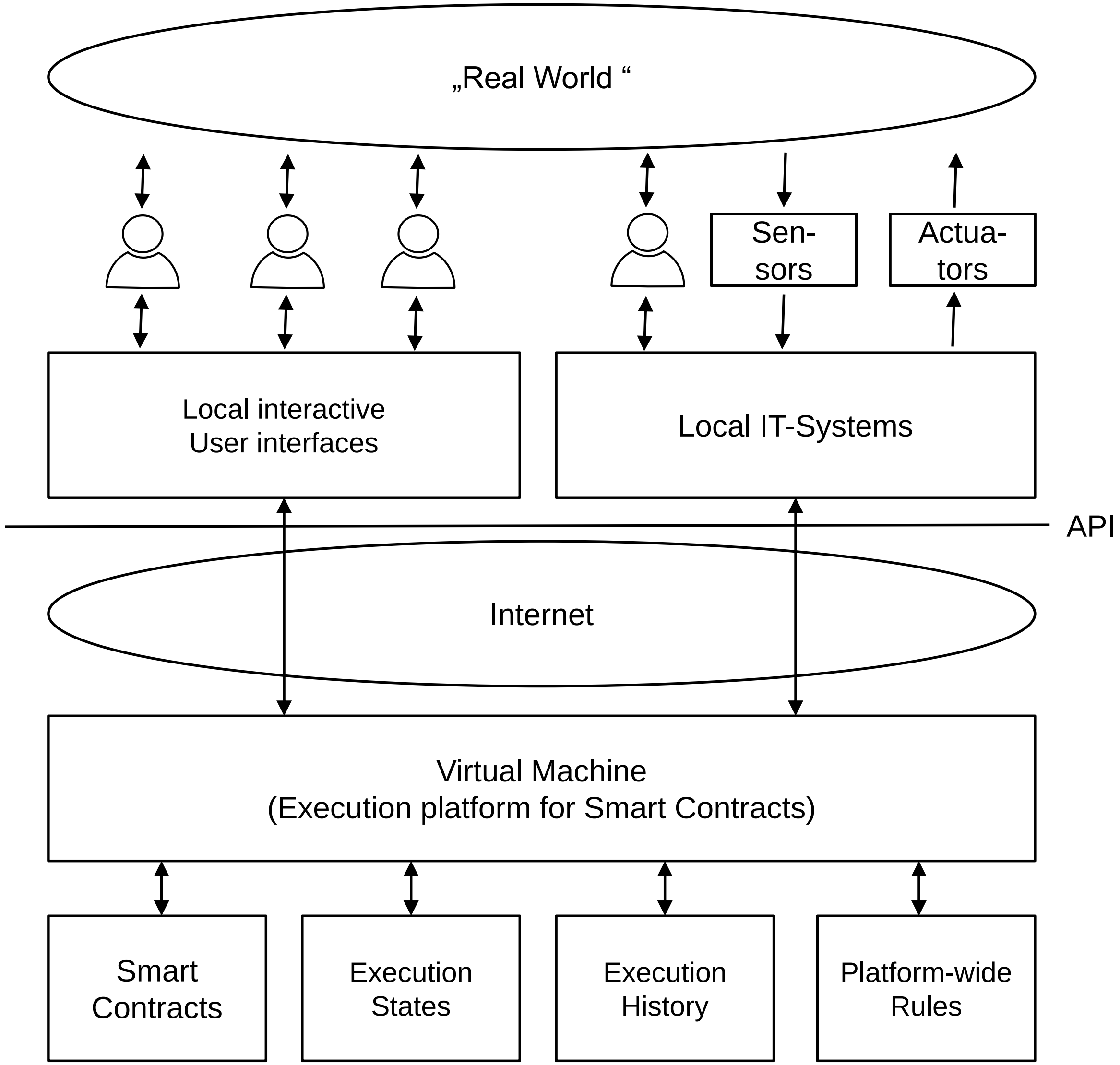
A reference architecture for a Smart Contract platform

Example:

Safe transfusion medicine

Blood bag supply chain involving

- Clinics
- Labs
- Logistic companies
- Blood donor centers



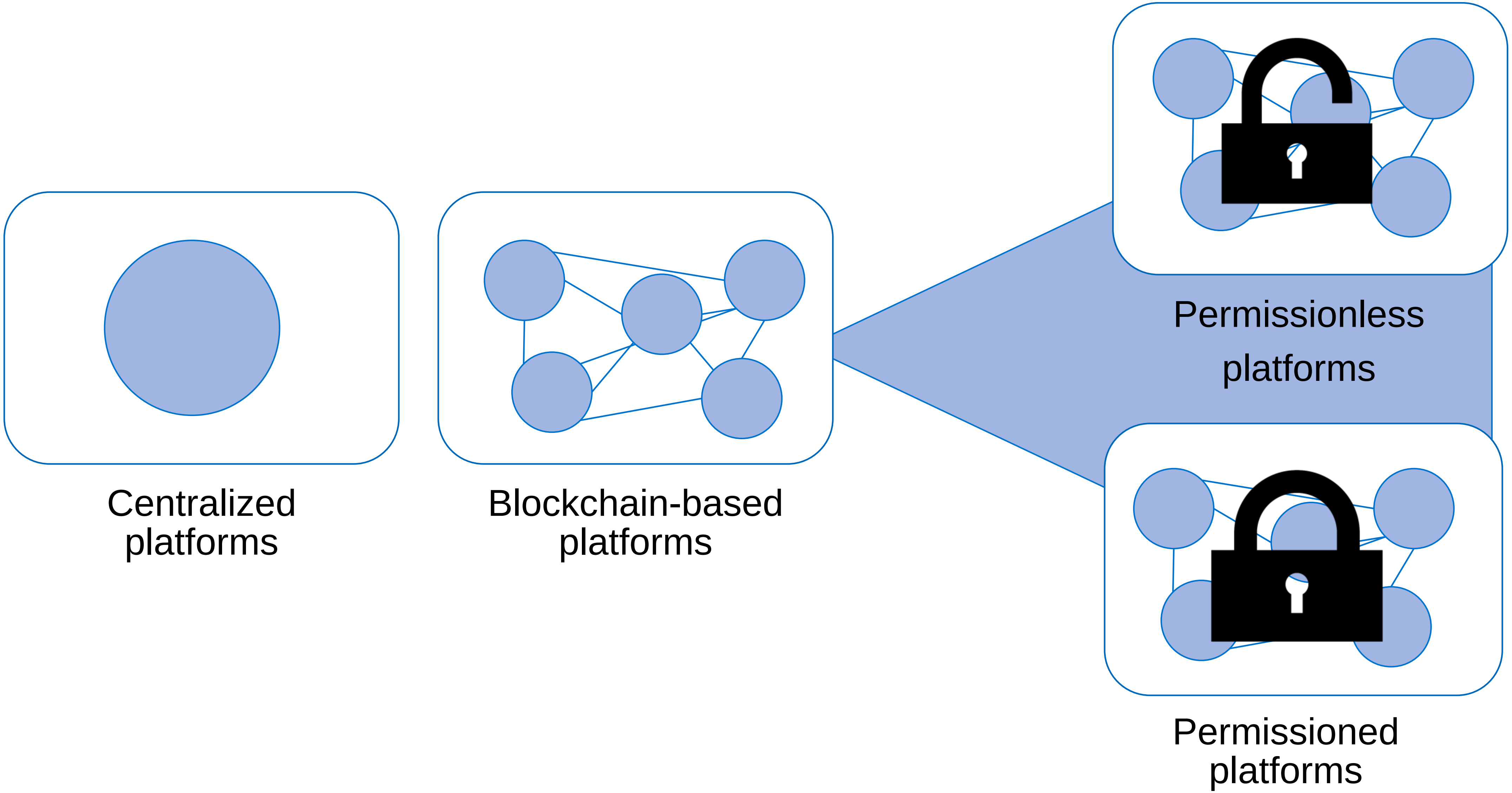
Working with a Smart Contract

- Deploy a contract template on the platform
- Create a contract instance with initial parameters
- Issue a single transaction with parameters:
  - Change the state of the smart contract instance
  - Validate that the transaction is valid in the current state
  - Perform (atomic) updates to the state variables of the contract
  - Invoke actions or queries of other smart contract instances
- Issue a query to access the state of one ore multiple smart contact instances
- Query the execution history

Characteristic properties of Smart Contracts

- **Deterministic execution**
  - No random numbers, current time, sensor values, …
  - Oracles
- **Terminating execution**
  - Domain-specific languages with limited expressiveness
  - Limited number of execution steps (Gas)
- **Isolated Execution**
  - Sequential
  - Serializable
  - Eventually consistent
- **Limited use of third-party library code**
  - Verifiability
  - Security

Implementations of the reference architecture



Centralized platforms

- Efficient
- Scalable
- Standardized components

- Need trust:
  - Availability
  - Sufficient resources
  - Fair conditions (e.g. pricing)
  - No manipulations

There are established non-technical measures to ensure trust, e.g. cooperatives, third-party (or state) supervision, Open Source licenses, etc.

Blockchain-based architectures

- Consensus of consortium on shared set of facts
- Transparency of all transactions
- Traceability of the complete transaction history
- Pseudonymity of the wallet owners
- Financial incentives for network growth
- Optimized for cross-organization collaboration
- Decentralized and redundant

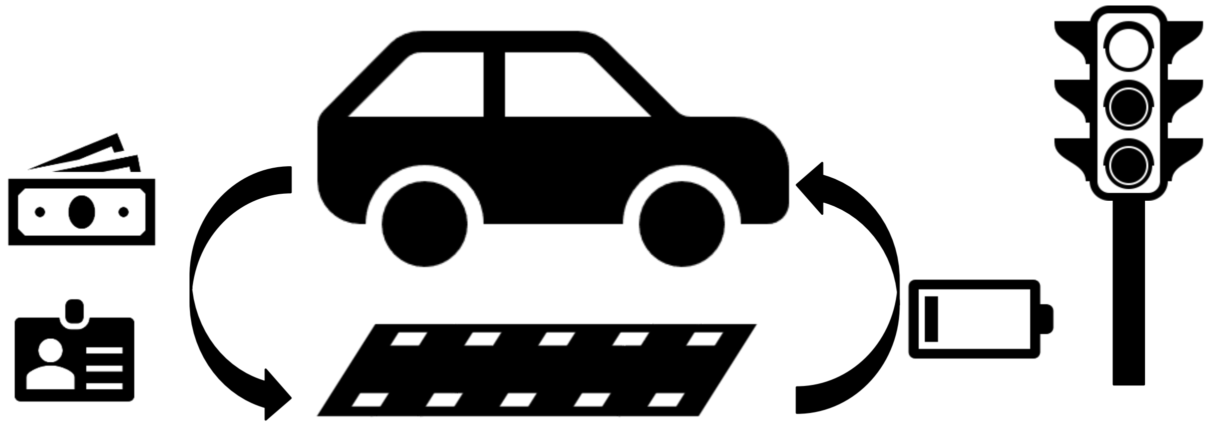
SE perspective:
- Data-centric
- Query support

- Transaction costs
- Platform governance
- System complexity
- Immature technology

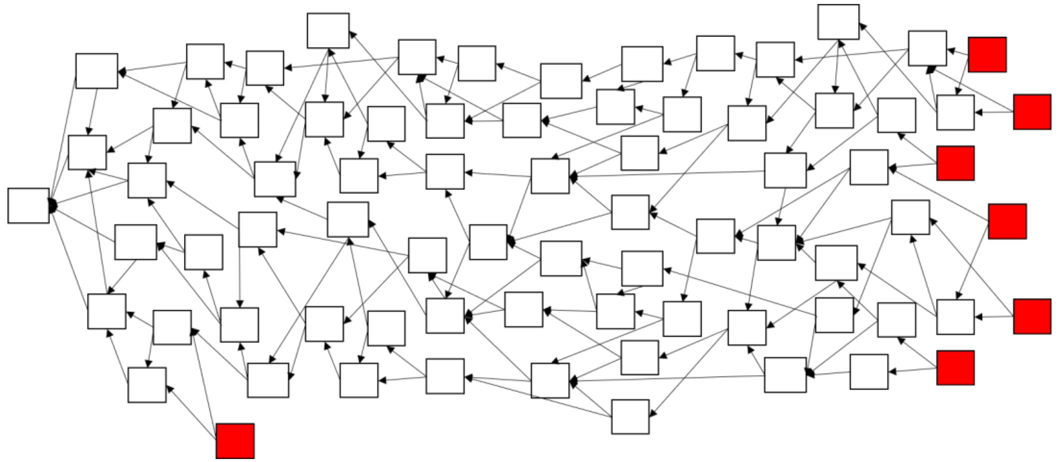
IOTA and the Tangle

Idea
- Focus on cross-enterprise Internet of Things use cases that do not require strict transactional semantics.

Example: Inductive charging:
- Instant authentication
- Trustless micro-payments
- Fast
- Scalable
- Immutable



Core Concepts
- Replace the linear chain by a directed acyclic graph. Do not use blocks, but transactions only.
- Atomic exchange of information and payments
- Each transaction has to validate two other transactions.



Not yet ready for production

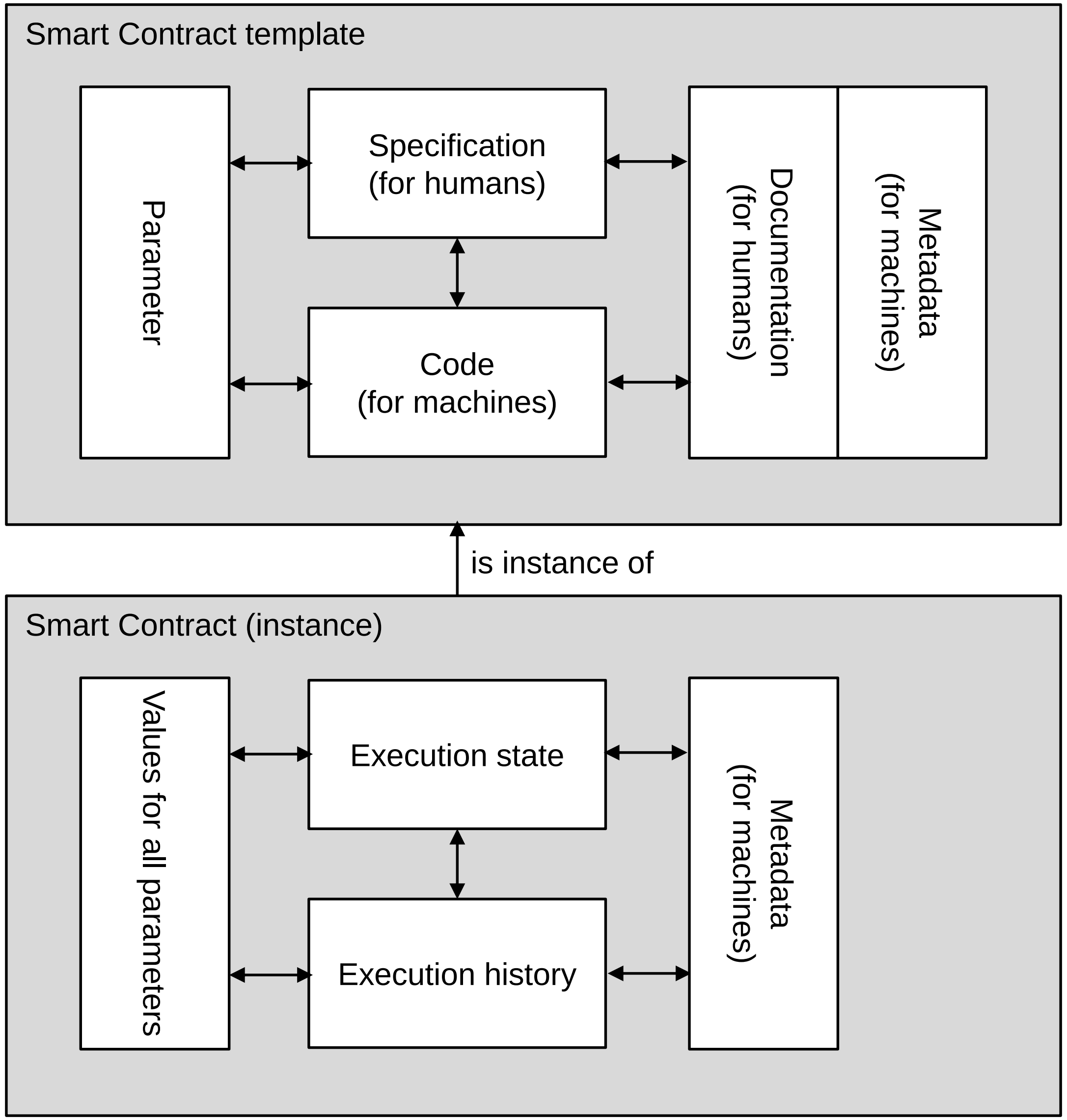
Hyperledger - Blockchain technologies for business

Hyperledger is an **open source** collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chain, manufacturing and technology.

- **Hyperledger Fabric**: A (federated, permissioned, private) blockchain based on a peer-to-peer network executing and validating chaincode.
- **Hyperledger Composer**: Development environment for smart contracts contributed by IBM.
- Several other (competing) projects

- Focus on business and cross-enterprise use cases
- Broad industry support
- Currently used mostly for education and internal proof-of-concept solutions

The structure of Smart Contracts



Smart Contract template

- Human-readable **specification** (typically text)
- Executable **code**
- List of **parameters**
- Optional **documentation**
- Machine-readable **metadata** (e.g. version, language, author, copyright)

➔ Analogous to a standard contract

Smart Contract instance

- Each instance can be referenced on the Smart Contract platform by a unique ID
- The execution state of the instance describes the values of all variables of the contract
- The execution history ensures traceability
- Optional metadata (e.g. timestamps) of the instance facilitate automatic administration

Blockchain-based architectures

Summary

Characteristics

- Focus on the precise, algorithmic descriptions of the rights and obligations of the contracting parties based on a shared state
- Trackability of all transactions
- Use of cryptographic methods
- Domain-specific languages
- Decentralized peer-to-peer execution environment

Drawbacks

- High execution costs
- Increased organizational and legal complexity
- Paradigm shifting for the development of new business models

Problems they do not address

- Correctness of input information
- Enforceability of rights outside the execution platform
- Correctness of the code