



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Introduction to Blockchain

Nicolò Romandini

Post-doc @ DISI

nicolo.romandini@unibo.it

Outline

1. Introduction
2. Why the blockchain?
3. Background
4. The Bitcoin blockchain
5. Beyond Bitcoin
6. Distributed Ledgers Taxonomy
7. Hyperledger Fabric



Some dates

- **1992:** *Haber and Stornetta*
“How to Time-Stamp a Digital Document”
- **2008:** *Satoshi Nakamoto*
“Bitcoin: A Peer-to-Peer Electronic Cash System”
- **2014:** *Vitalik Buterin*
“Ethereum Whitepaper”

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

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.

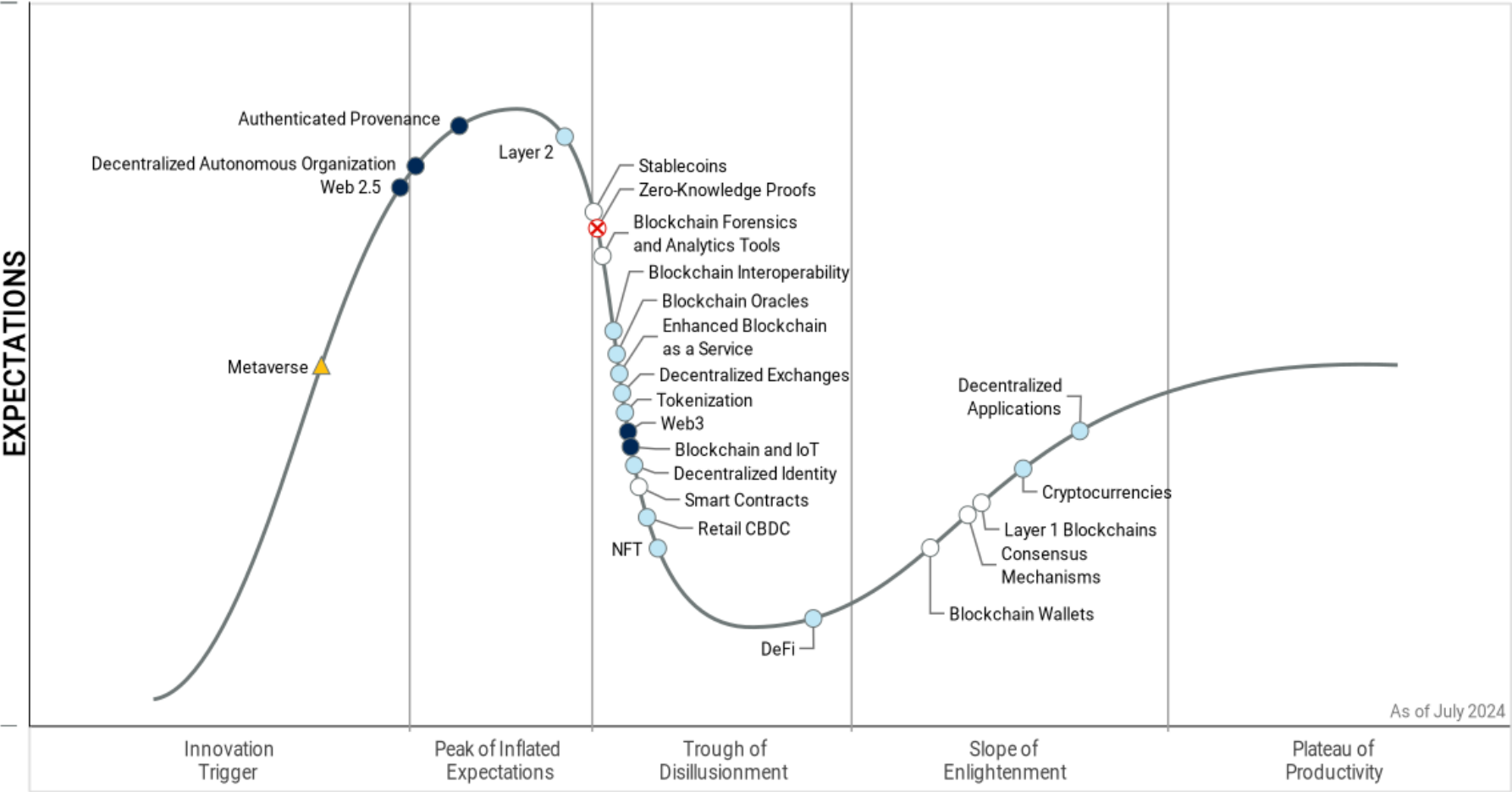
1. Introduction

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, and there is a broader cost in the loss of ability to make non-reversible payments for non-



Blockchain Hype Cycle

Hype Cycle for Web3 and Blockchain, 2024



Plateau will be reached: ○ <2 yrs. ● 2-5 yrs. ● 5-10 yrs. ▲ >10 yrs. ✗ Obsolete before plateau

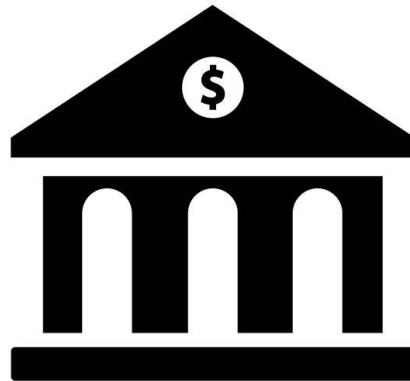
Why the blockchain?

Digital money evolution



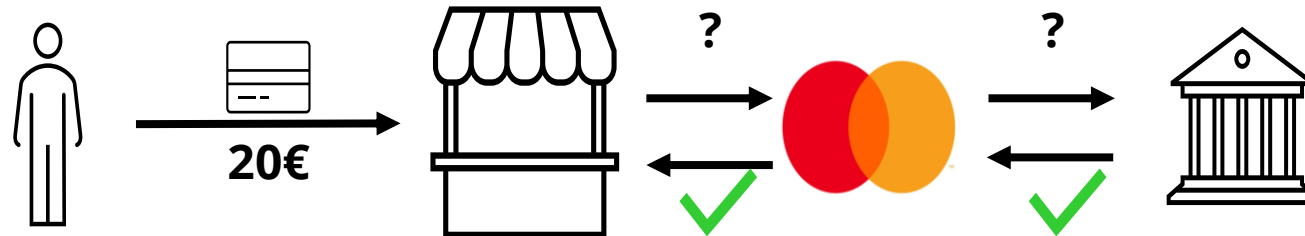
The trust problem

- The modern economic and monetary system is based on **trusted third parties** to guarantee the authenticity of documents and the validity of economic transactions between parties that do not trust each other.



Credit Cards

- Most common option for payments. Each buyer and each seller has to open a **credit line** in advance with a bank or a broker.
- Then, for each transaction, the third party **has to confirm** that the buyer has the funds to complete the operation.



Credit Cards

This “simple” method of payment has several issues:

- **Costs:** each intermediary can demand commissions for the service it provides, even large ones.
- **Timings:** procedures can depend on the opening hours for banks and offices.
- **Data ownership:** the user is not the real owner of his/her data or his/her account; services are provided according to banks/other entities policies.
- **Errors:** various human or technical errors, beyond the responsibility of the user, possible throughout the whole process.



Cash Money

Cash money has clear advantages compared to credit cards:

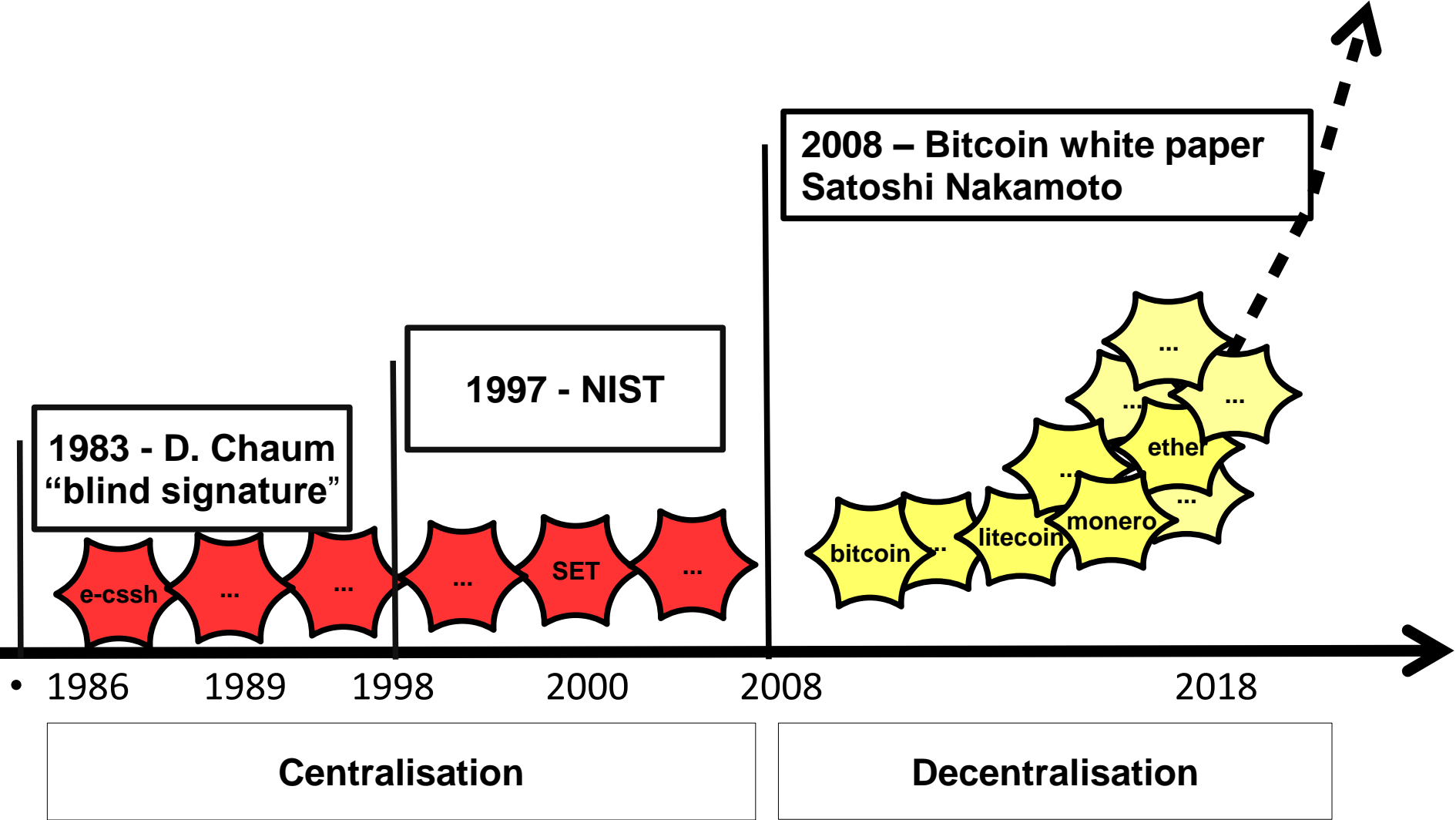
- it guarantees **full anonymity**.
- it does **not** need third parties.



However, there are other issues:

- we need to “**coin money**”, i.e. a third party that initially distributes the bills and that can be queried against their validity.
- it works only **offline**.

Digital Cash



Digital Cash

The problem with the first attempts to create digital cash was that in one way or another **they all needed a trusty third-party entity** to:

- validate transactions.
- coin new coins.

So, let's eliminate the intermediary.



Background

Information security and distributed systems



Information Security

- **Confidentiality:** protect information from unauthorized access and misuse.
Encryption
- **Integrity:** protect information from unauthorized alteration.
Hash Function
- **Authenticity:** establish the validity of information.
Digital Signature



Asymmetric Cryptography

Each entity has two keys linked **together**:

- a public key (**shared** in a *certificate X.509* issued by **Certificate Authorities**, identify the entity)
- a private key (kept **secret**)

RSA Key Generation

Output: public key: $k_{pub} = (n, e)$ and private key: $k_{pr} = (d)$

1. Choose two large primes p and q .
2. Compute $n = p \cdot q$.
3. Compute $\Phi(n) = (p - 1)(q - 1)$.
4. Select the public exponent $e \in \{1, 2, \dots, \Phi(n) - 1\}$ such that

$$\gcd(e, \Phi(n)) = 1.$$

5. Compute the private key d such that

$$d \cdot e \equiv 1 \pmod{\Phi(n)}$$



RSA Encryption

- Encrypt

$$c = m^e \bmod(n)$$

- Decrypt

$$m = c^d \bmod(n)$$

Very simple if you know the private key, but (almost) **unfeasible** otherwise.



RSA Digital Signature

A **digital signature** is a mathematical scheme for verifying the **authenticity** of digital messages or documents.

- Sign

$$s = m^d \bmod(n)$$

- Verify


$$m = s^e \bmod(n)$$

Producing a valid signature is (almost) **unfeasible** without knowing the private key.



Public Key Infrastructure

A **public key infrastructure (PKI)** is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke **digital certificates (X.509)**, which bind **public keys** with respective **identities**.

**Informazioni sul certificato**

Scopo certificato:

- Dimostra la propria identità ad un computer remoto
- Garantisce l'identità di un computer remoto
- 1.3.6.1.4.1.6449.1.2.2.79
- 2.23.140.1.2.2


* Per ulteriori dettagli consultare l'informativa dell'Autorità di




Rilasciato a: unibo.it **Issued to**

Rilasciato da: GEANT OV RSA CA 4 **Issued by**

Valido dal 16/11/2021 **al** 17/11/2022 **Valid from-to**

Percorso certificazione **Certificate chain**

 Sectigo (AAA)

-  USERTrust RSA Certification Authority
 -  GEANT OV RSA CA 4
 -  unibo.it



Hash Function

A **hash function** is a function that produces a **fingerprint** (digest, hash, ...) of fixed length from an arbitrary input.

- Easy to compute.
- Non-invertible.
- Deterministic and random output.
- Low probability of collisions.

blockchain -> SHA3-256 ->

45740502697d57cbc7e6522372d3247adf1ab8f1cdb0cda1f20a022bf3e153d0

Blockchain -> SHA3-256 ->

94074fd5892e84da500a78e4c02ff986c38815ad4063441a1caad310e89cf709

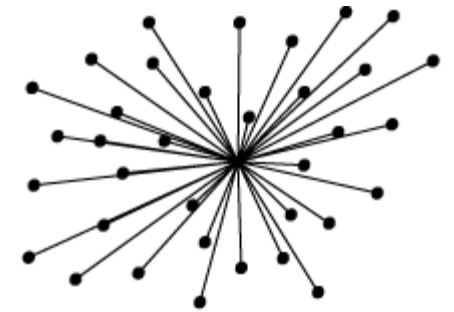


Distributed and Decentralized Systems

We can categorise systems according to **two** criteria: **where** the **computation** takes place and **who** makes the **decisions**.

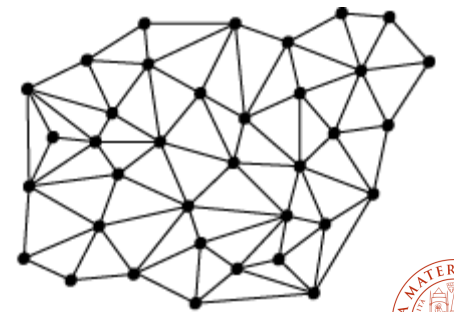
Computation

- Centralized: all operations are performed by a single node.
- Distributed: operations are spread across all nodes.



Governance

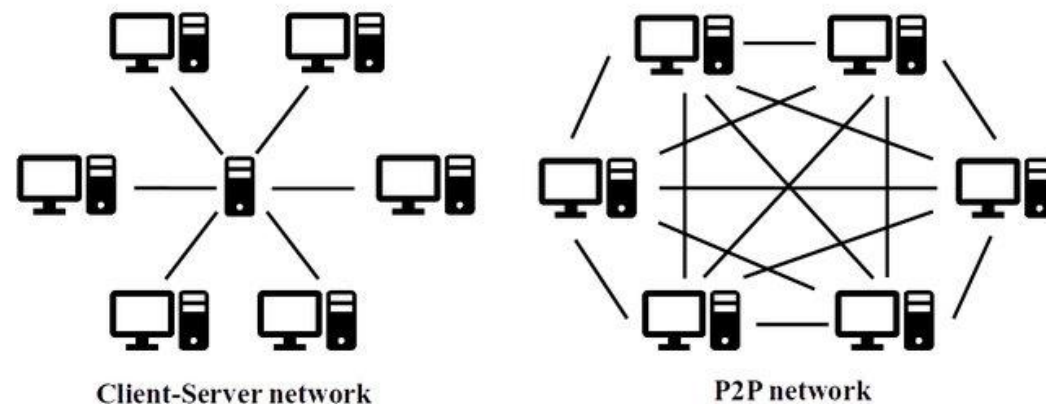
- Centralized: one single node decides for all.
- Decentralized: each node participates in the decision-making process.



Peer-to-Peer (P2P) Network

A **P2P** network is a network in which the computers of connected users act as **both client and server**.

Peers make a portion of their **resources** directly available to other network participants, **without** the need for **central coordination** by servers or stable hosts.



Communication Models

Two type of communication models:

Synchronous:

- All processes share the same clock.
- There is an upper bound on message delivery time.

Asynchronous:

- Weakest possible assumptions.



Failures in Distributed Systems

A **failure** (or fault) is a condition that causes a functional unit to **fail to perform** its required function.

- **Benign Failure:** when a node stops performing the operations it is supposed to perform. *Crash and omissions.*
- **Byzantine (arbitrary) Failure:** when a node starts to perform arbitrary operations, not present in the correct flow of operations.



Consensus

Distributed processes that have to **agree** on a single value (e.g., new status of the system).

Properties

- **Agreement:** If a correct process decides a value, then all correct processes eventually decide the same value.
- **Termination:** Every correct process eventually decides some value.
- ...



Consensus

Fisher, Lynch, Paterson (1985) - ***Impossibility of distributed consensus with one faulty process***: There is no deterministic protocol that solves consensus in an asynchronous system where at most one process can fail by crashing.

But is it often said that **blockchain solves** consensus? We will see that this is **not actually true**.



The Bitcoin blockchain

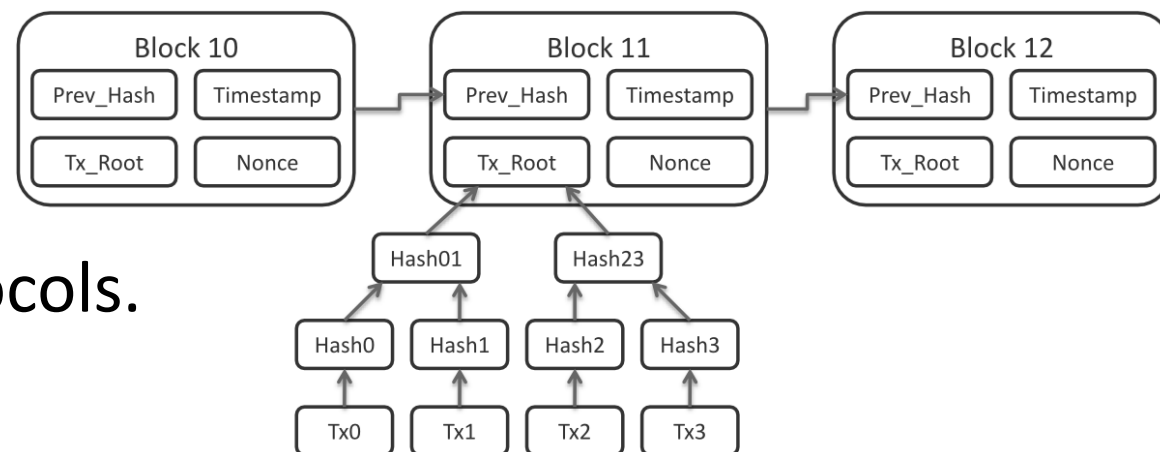


Structure

A blockchain is an **append-only** list of records, called **blocks**, that are linked together using **cryptography**.

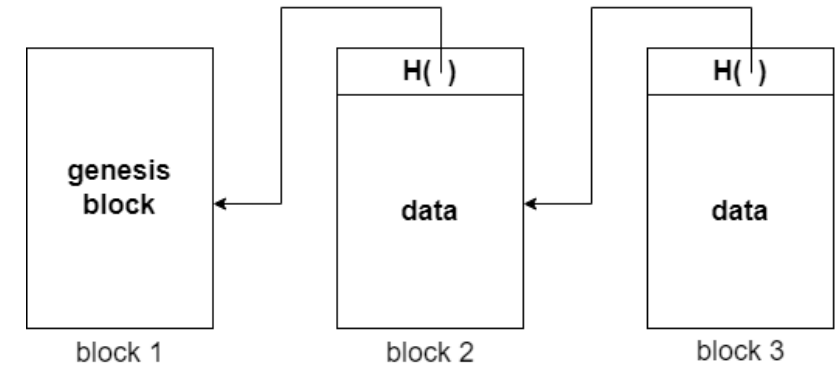
Blockchains are typically **managed by a decentralized peer-to-peer network** for use as a publicly **distributed ledger**.

- **Trust** is based on the use of strong **cryptographic** primitives and protocols.
- Everyone **has a copy** of the ledger.

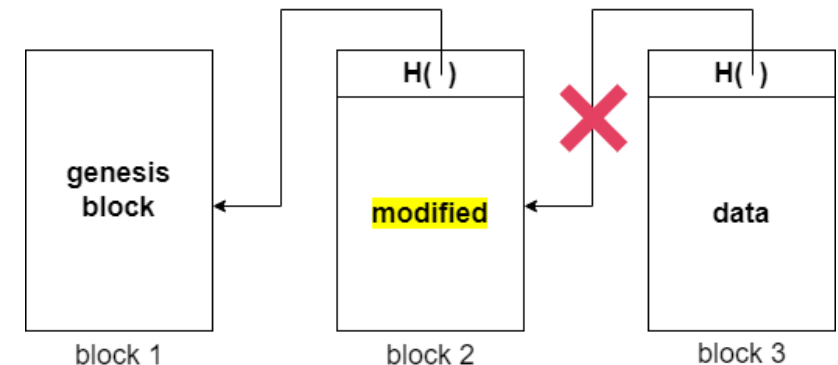


The chain

Chain of blocks linked by **hash pointers**.
Each block contains the hash of the
previous block.



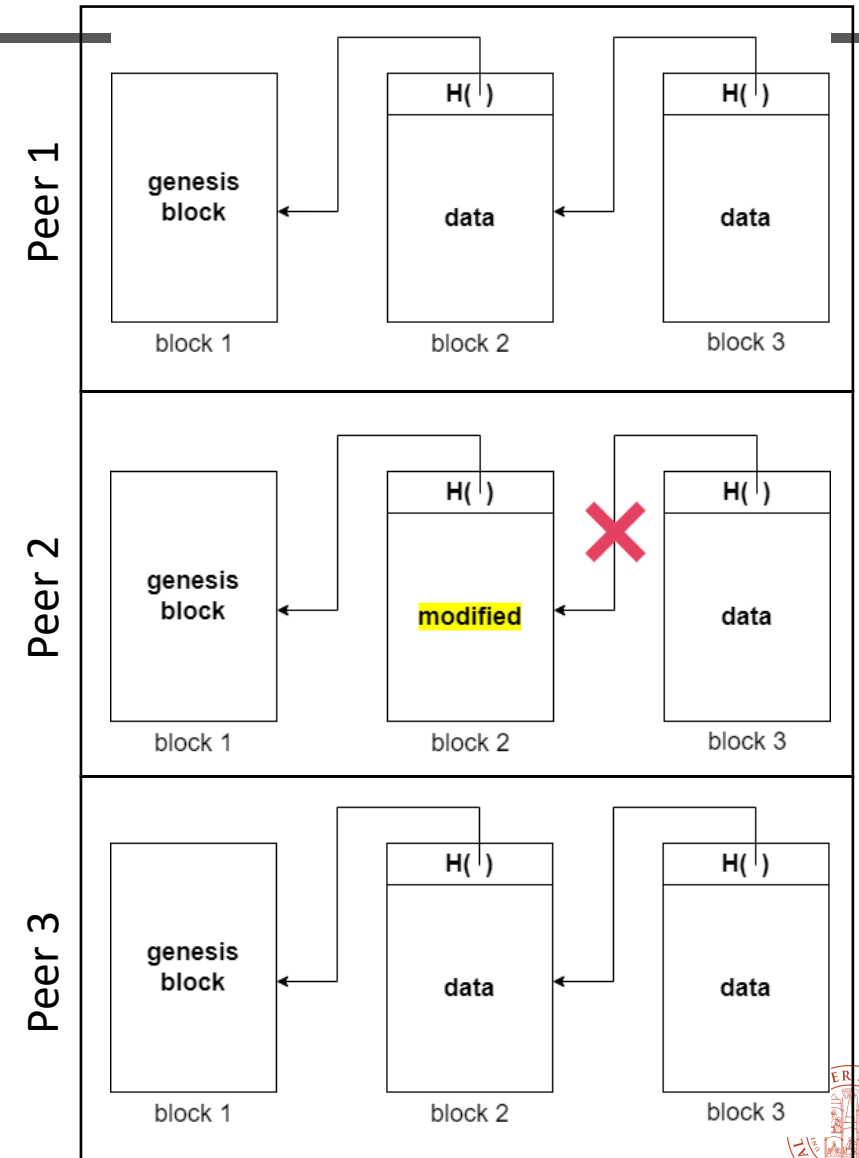
A change in a block produces a **break** in
the hash chain. To make the chain valid
again, it's necessary to **recompute the hash**
of all the next blocks.



The chain

Since the whole chain is **replicated** on the network, it is possible to find **the correct version** of the chain by asking it to the **majority** of the nodes (assuming that the majority of the nodes is **honest**).

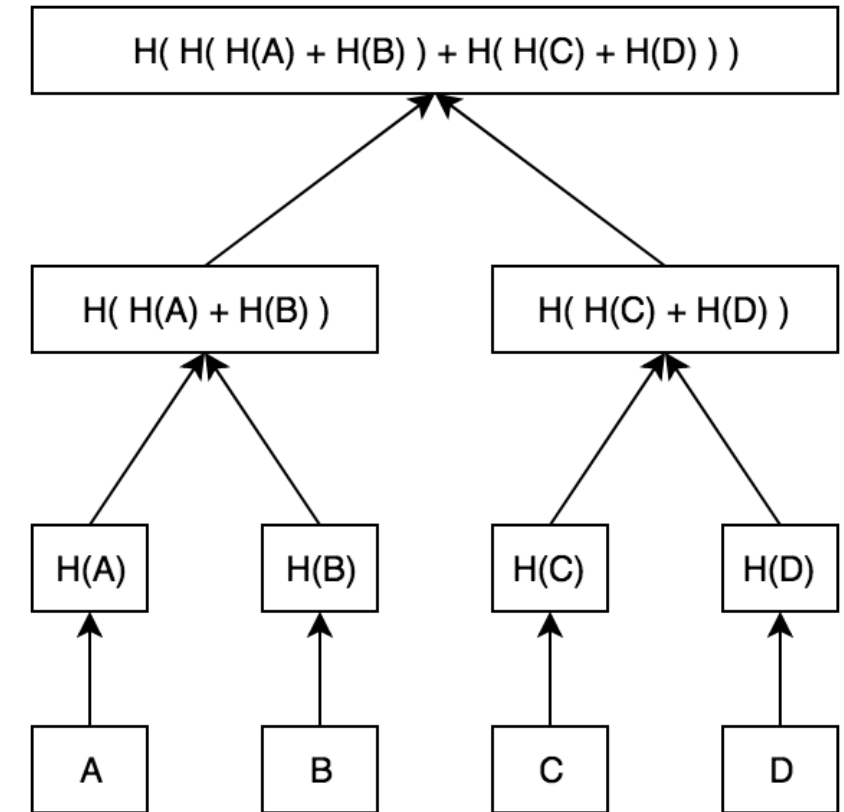
Assumption: **50% + 1** of nodes are benign.



Merkle Tree

A Merkle tree is a **hash-based** data structure.

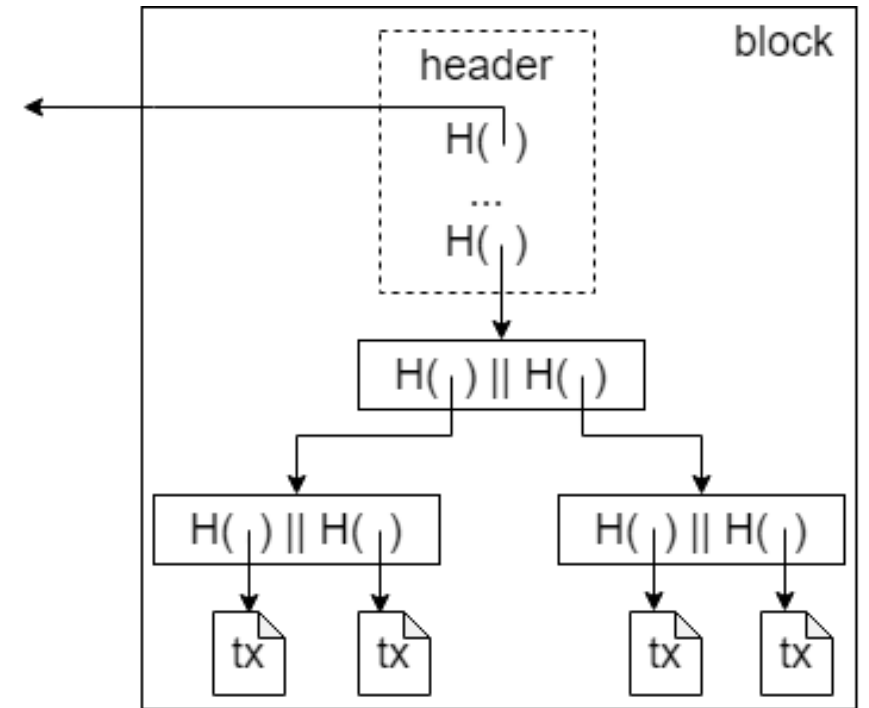
- The **leaves** of the tree are the data (ordered), whereas the **other nodes** contain couples of hashes.
- This structure makes it **simple** to prove that a certain data (does not) **belong**(s) to the tree.



The block

The block consists of **two** main parts:

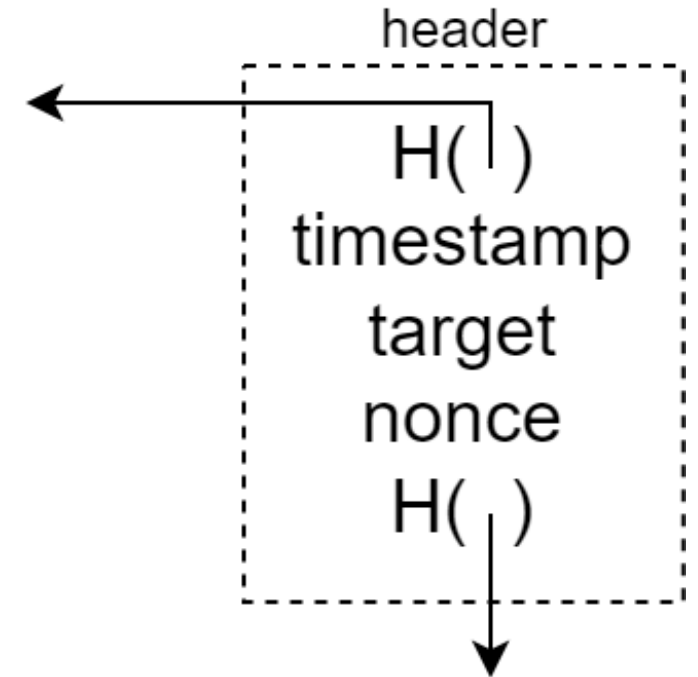
- **Header:** contains the hash of the **previous block**, the **root hash** of the Merkle tree and other information that we will see later.
- **Merkle tree:** the leaves of the tree are the transactions contained in the block.



The header

The **header** of a block consists of:

- The **hash** of the **previous** block.
- A Unix **timestamp**.
- A **target** value (256 bit). First s bits are 0.
- A **nonce** value (32 bit).
- The **root** of the Merkle tree.



The hash of the block is calculated by taking the **header of the block as input**.

Nakamoto Consensus

“**Nakamoto consensus**” is the algorithm used in Bitcoin network to achieve **trustless consensus** among participants.

The context in which Bitcoin operates is:

- **Asynchronous, distributed** and **decentralized**.
- The number of nodes is **not known** and they may be **Byzantine**.
- The **only assumption** is that the **majority** of nodes are **benign**.



Nakamoto Consensus

Peers have to **agree** on the **new state** of the system (i.e., the **next block to be added** to the blockchain).

Peers who generate new blocks are called miners. A miner **receives transactions** from clients. After **validating** them, it can start the process of generating a new block.

Intuition: each time, a block generated by a miner is **randomly chosen**.



Proof of Work (PoW)

To **discourage byzantine** miners and **DoS attacks**, Bitcoin requires miners to **solve a “puzzle” (mining)** in order to generate a new block. This is the so-called **Proof of Work**.

Solving the puzzle in a short time requires a large amount of **computational power**.

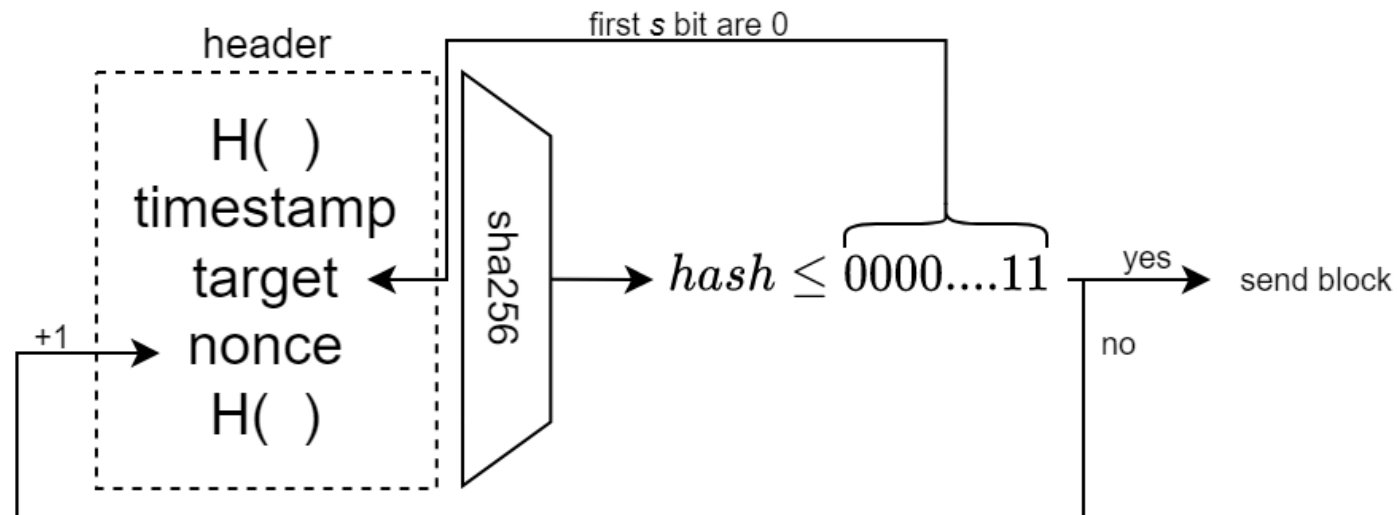
Also, if you solve the puzzle, you get a **reward** in Bitcoin (currently 6.25 BTC).



Proof of Work (PoW)

The puzzle consists of finding a **nonce value** such that the **hash** of the block is **less than** or **equal** to the **target value**.

- The best algorithm is **brute force**.
- The target **adjusts** every **2016 blocks** (roughly two weeks) to try and ensure that blocks are mined once every **10 minutes** on average.



Proof of Work (PoW)

Two possible scenarios:

- The miner **solves** the puzzle, **adds** the block to its chain and **sends** it to all other nodes.
- The miner **before solving** the puzzle receives a block from another node. In this case, it **stops searching** for the solution and **adds** the block to its chain.

Actually, a new block may arrive **at any moment (asynchrony)**.

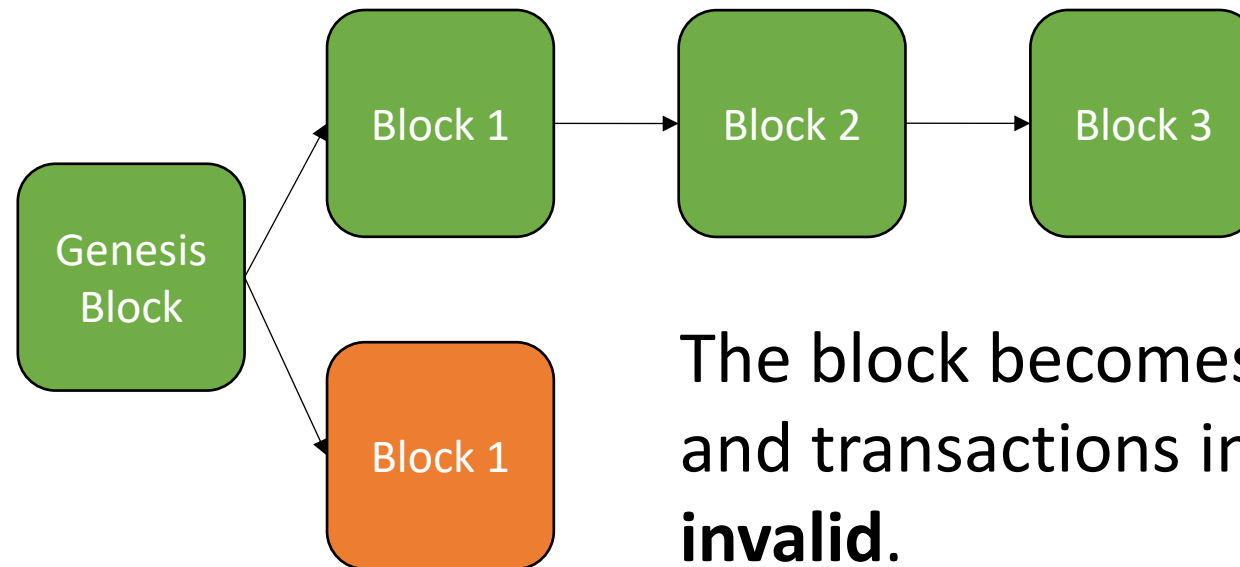
What happens if a miner **receives a block** when it has **already** added one to its chain?



Forks

The miner **keeps both blocks** as if they are both valid, resulting in a **fork**.

The **rule** is that the **longest chain** is the one that individual nodes accept as a **valid** version of the blockchain.

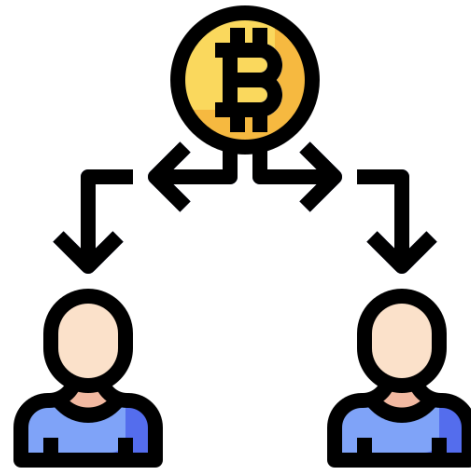


The block becomes an “**orphan**” and transactions in it are considered **invalid**.

Double-spending

Double-spending is the risk that a cryptocurrency can be **used twice** or more. In a context **without intermediaries** this is a serious **problem**.

Assumption: if you have already spent money on a transaction that is in the valid chain, the transaction validation process **rejects** the new transaction.



Double-spending

What an attacker might do is send **two different transactions** to two different miners.

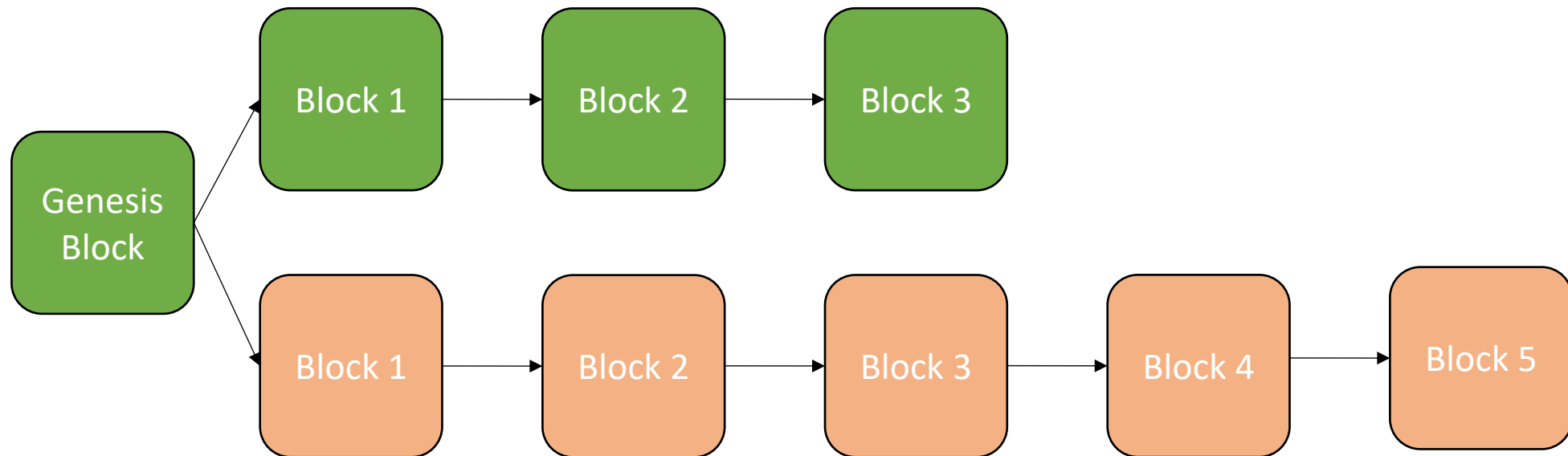
But **no problem! Eventually** one of the two blocks will be considered **invalid**.

The **rule** is to **wait for the block** in which the transaction is present **to be “buried”** by other blocks (about **6**).



51% Attack

It is possible to **exploit** the fork mechanism **to double-spend**, if **one entity owns 51%** of the **computational power** of the entire network.
In 2014 mining pool Ghash.io controlled 51% of all the processing power.



More on Nakamoto Consensus

How does the Nakamoto Consensus behave in relation to the **properties** we saw earlier?

- **Agreement: probabilistic.** If we stop at any time, we can always have forks.
- **Termination:** to have deterministic agreement we **cannot stop!**

The main **drawback** of Bitcoin is that **it can never stop** in order to work.



More on Nakamoto Consensus

“Every property is a combination of a **safety property** and a **liveness property**” (Alpern and Schneider)

- **Safety Property:** a safety property asserts that **nothing bad happens** during execution (**agreement**).
- **Liveness Property:** a liveness property asserts that **something good eventually happens** (**termination**).

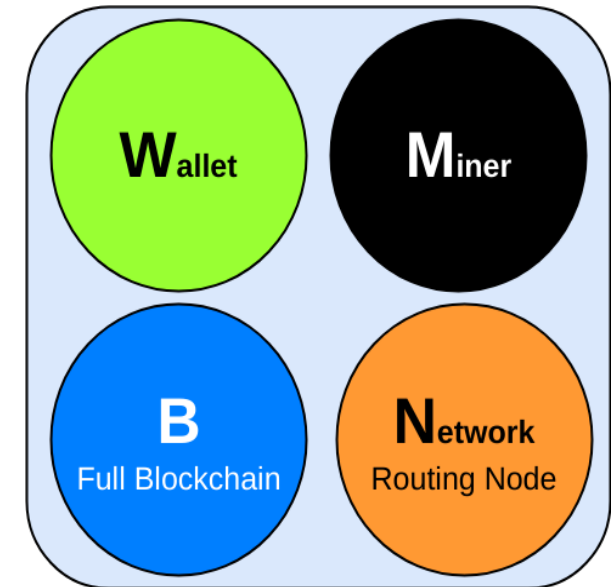
In Bitcoin, **liveness ensures safety!**



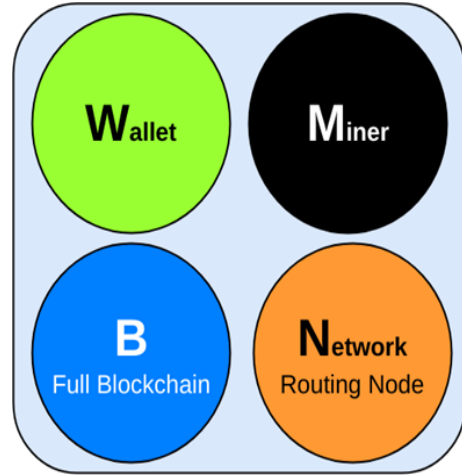
Role of Peers

The **peers** of the Bitcoin network can have the following **roles**:

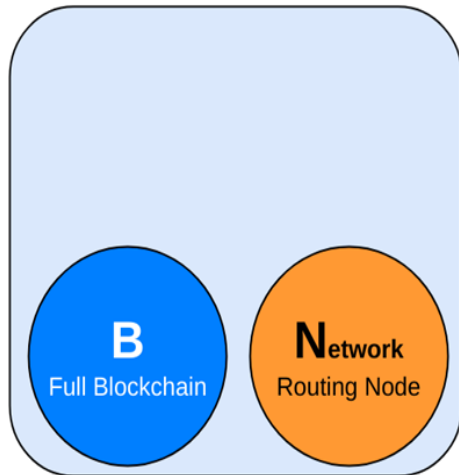
- **Wallet**: node that stores an identity.
- **Network**: node able to receive/forward blocks from/to other nodes.
- **Miner**: node that competes with others in the generation of new blocks by solving hash puzzles (mining).
- **Full blockchain**: node that stores the whole blockchain. That means not only the headers, but also the transaction data.



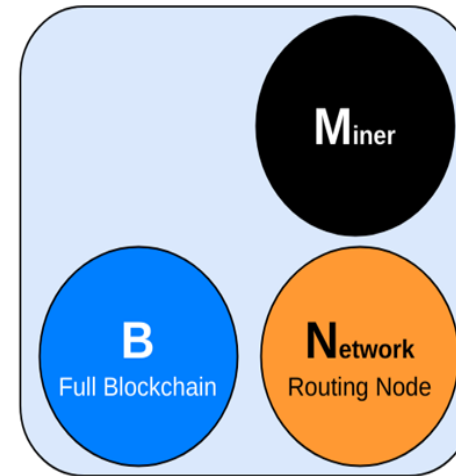
Combinations of Roles



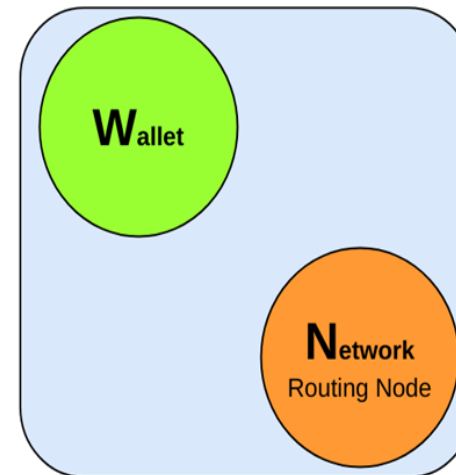
**Reference
Client
(Bitcoin Core)**



**Full Bitcoin
Node**



Miner Node



**Lightweight
Wallet Node**

Full Blockchain

Full blockchain nodes **store the whole blockchain** (block headers and data), i.e. about **450 GB**. They must **validate** each received block and deal with chain forks.

The other nodes can choose to

- **not store** the blockchain.
- act like full nodes, but **store only a recent portion** of the blockchain.
- **store a lightweight version** (headers only) of a recent portion of the blockchain, executing a “fast validation” of the transactions.



Identity and Address

An **identity** corresponds to a **public key**, which is an anonymous identifier.

However:

- **Re-using** the same key more than once allows anyone to **track the activity** of that key (the blockchain is public) and to infer the real identity of the owner.
- Every time a node broadcasts a transaction, the others **can store the IP address** of the message and **associate** it to the public key.

For a **true anonymity**:

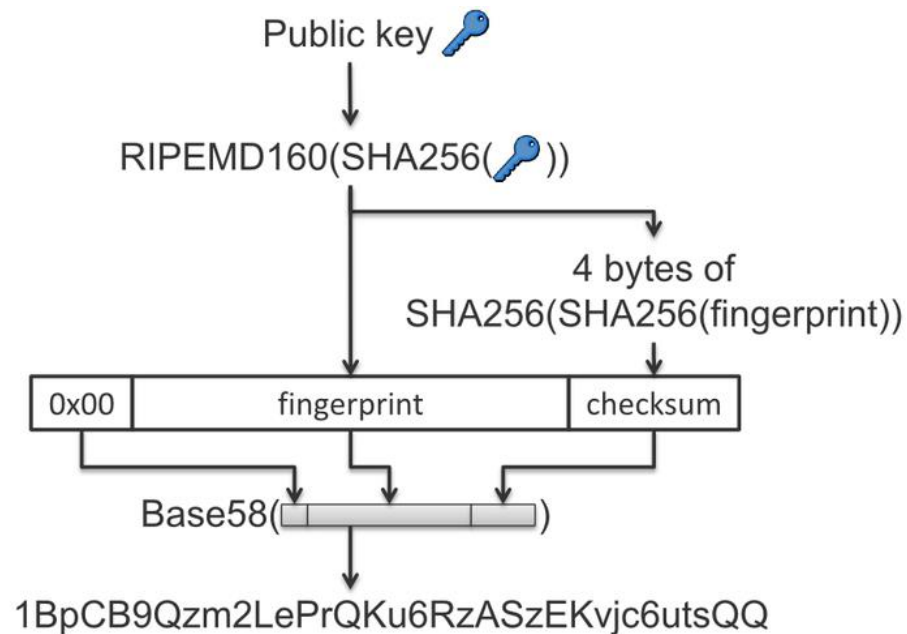
- it is necessary to frequently **change** the public key.
- it is necessary to **mask** the real IP address.



Identity and Address

Bitcoin transactions are between addresses.

An **address** is derived directly from a public key by following this procedure:

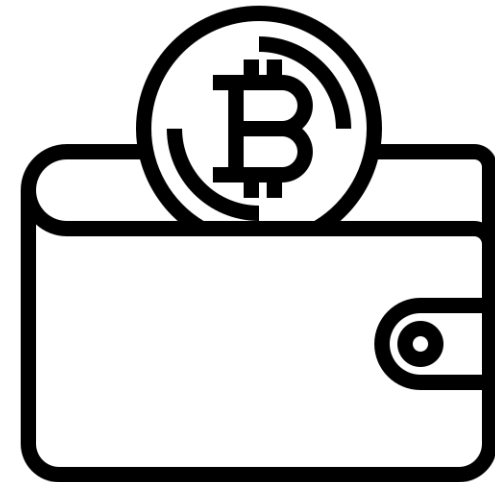


Wallet

A Bitcoin **wallet** is a device or program for **holding** and **sending** Bitcoins.

Bitcoin wallets **contain the private keys** needed to **sign** Bitcoin transactions. Anyone who knows the private key can control the coins associated with that address.

It does not contain Bitcoins!

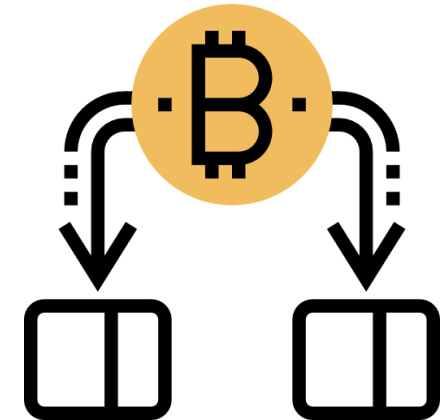


Transactions

A **transaction** is a **transfer** of Bitcoin value that is broadcast to the network and collected into blocks.

Two main types of transaction:

- a transfer of **existing** money among identities.
- **coinbase** (generation of new money).



Transaction between addresses

Intuition: cash passes from one owner to another. It is possible to reconstruct the history of owners since its creation.



ECB

Bank

Mario

Paolo

Bitcoin **does not save** “the coins” somewhere (like real cash in wallet), but simply **saves every transaction** on the blockchain.

To prove that I have the right to spend a quantity of Bitcoins, I only need to prove that **there is a transaction** on the blockchain **in my favour** with that quantity.

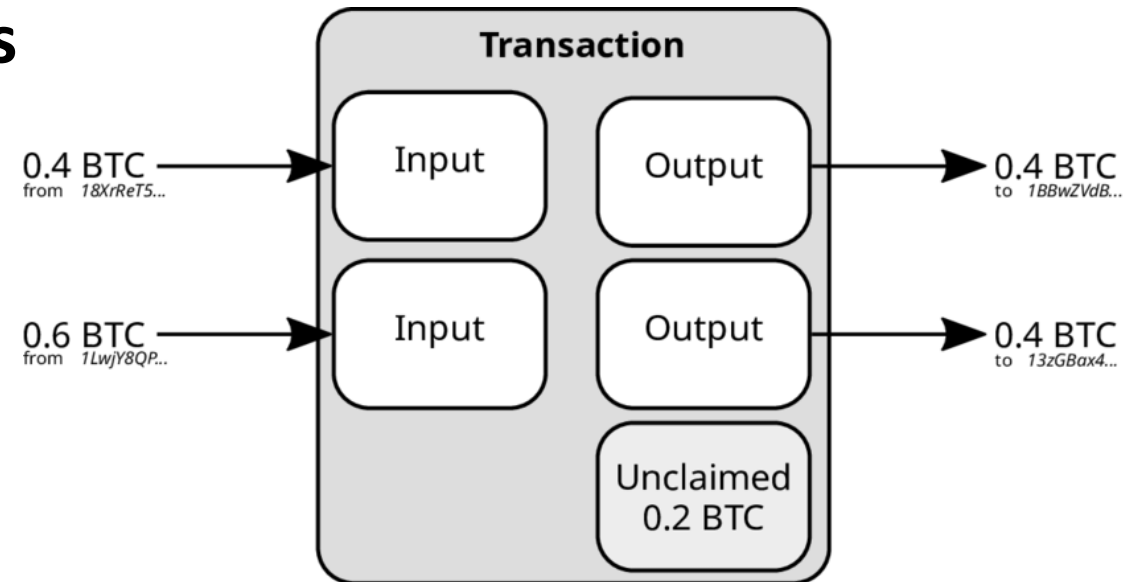


Input and Outputs

A transaction may have multiple:

- **Inputs:** reference to transactions on the blockchain that demonstrate my possession of those BTCs.
- **Outputs:** reference to the addresses of BTC receivers (also myself).

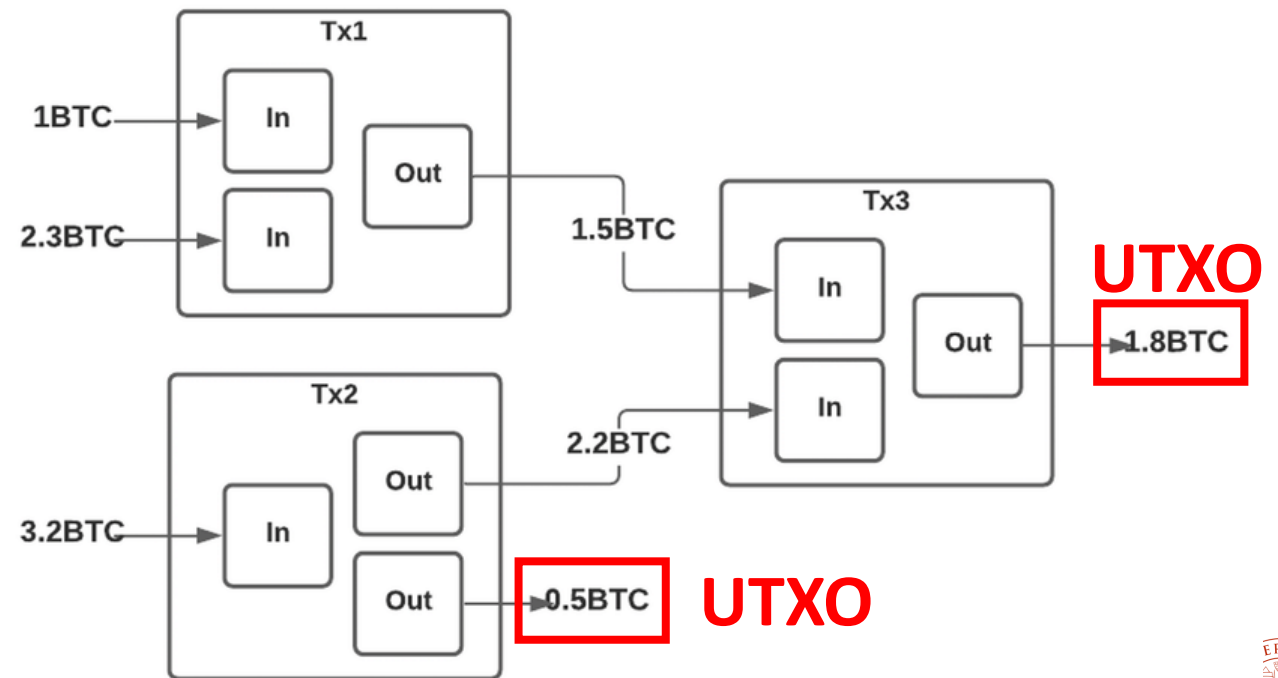
Inputs - Outputs = **Unclaimed BTCs**



Unspent Transaction Output

An unspent transaction output (**UTXO**) represents some amount of digital currency which has been authorized by one account to be spent by another, **but not yet spent**.

The **sum** of the amounts of all UTXOs is the **total supply of existing Bitcoin** at that point of time.



Scripts

Actually, input and output contain **scripts**, written using the Bitcoin **scripting language**.

The input script (**unlocking script**) is to **prove** that you can spend the BTC, the output script (**locking script**) ensures that only the intended receiver can spend the BTC.

Input Script (**scriptSig**): is the part of a transaction which contains the **required signatures** and the script which unlocks a UTXO for spending.

<signature>, <pubKey>



Output Scripts

Output Script (**ScriptPubKey**): is a script which controls how Bitcoin can be spent.

Two types of output scripts:

- Pay to Public Key Hash (**P2PKH**):

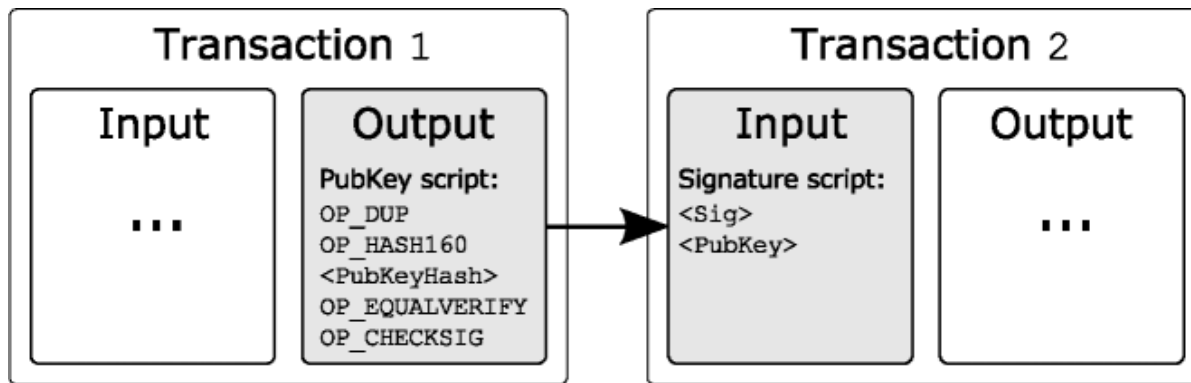
OP_DUP, OP_HASH160, <address>, OP_EQUALVERIFY, OP_CHECKSIG

- Pay to Script Hash (**P2SH**): for multisig.



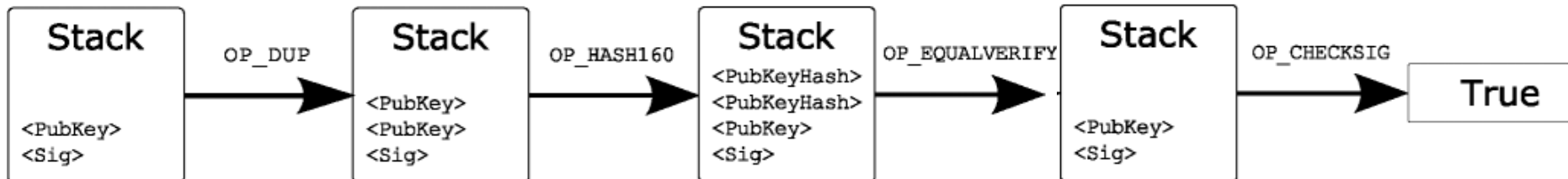
How to verify a transaction (P2PKH)?

To **verify** a transaction, simply **execute** the input and output script.
In this case, the output script is that of the transaction already in the blockchain!



Very difficult to calculate:

- PubKey from address.
- PrivateKey from PubKey.



Coinbase Transaction

A **coinbase transaction** is the **first** transaction in each block. The coinbase transaction assign the **block reward**, which is currently **6.25 BTC per block**, and also collects the **cumulative fees** (remember the unclaimed BTCs?) of all transactions in the block.

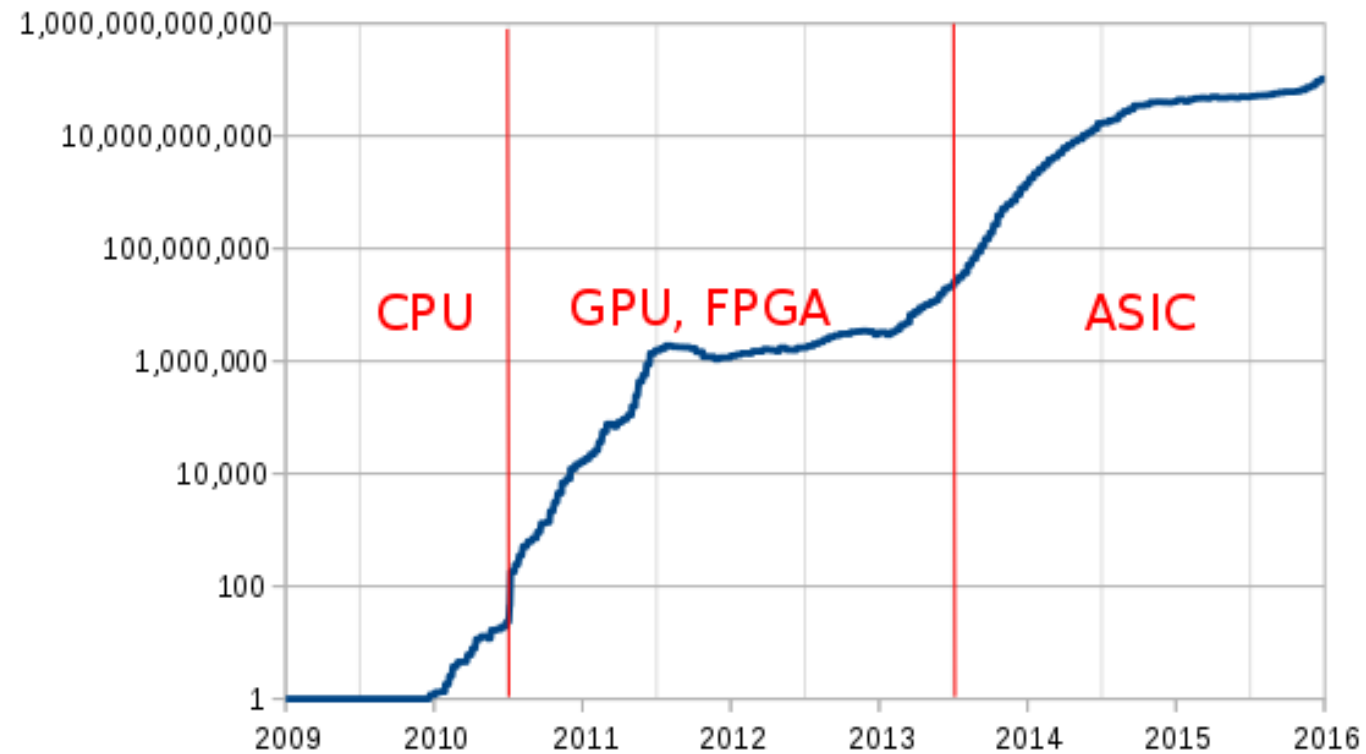
Why **fees**? Clients can give a **small incentive** to the miner, e.g. to process their transaction faster.

Since this transaction is creating new Bitcoin, the coinbase transaction **is valid without any inputs**.



Some Numbers

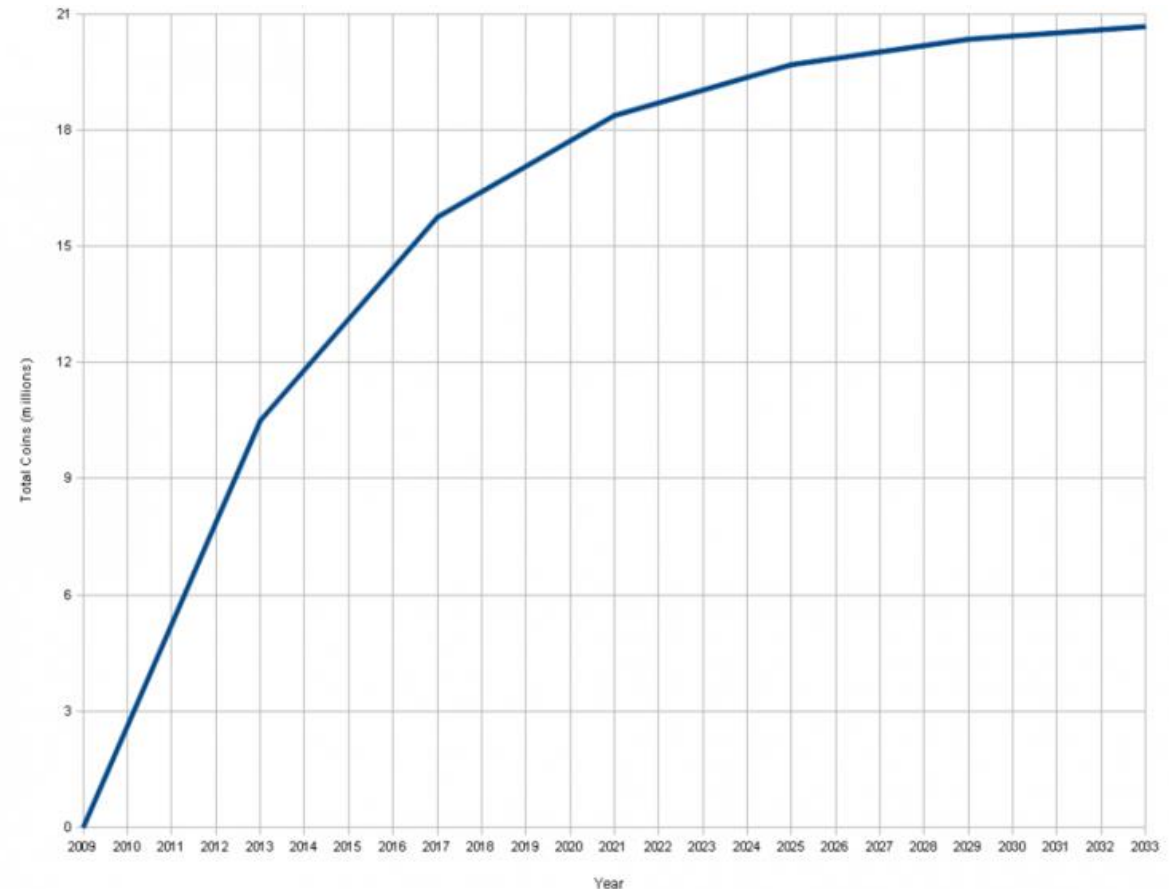
Mining difficulty (logarithmic scale)



Some Numbers

There will never be more than **21 million** Bitcoin.

- This value is **hardcoded** into the protocol.
- There are now **19M** Bitcoins in circulation.

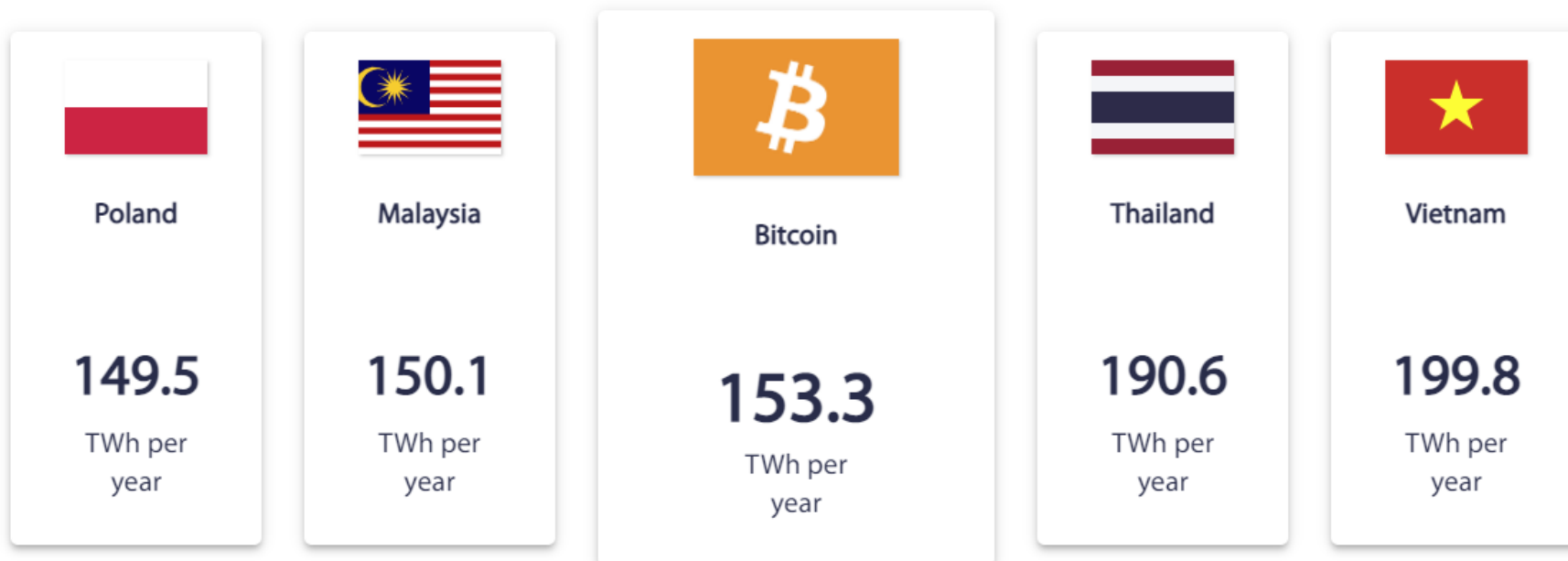


Some Numbers

Energy Consumption

Country Ranking

Country comparisons are, for better or for worse, the most common type of comparison. They are frequently used in the public debate to support positions of concern about the scale of Bitcoin's electricity consumption.



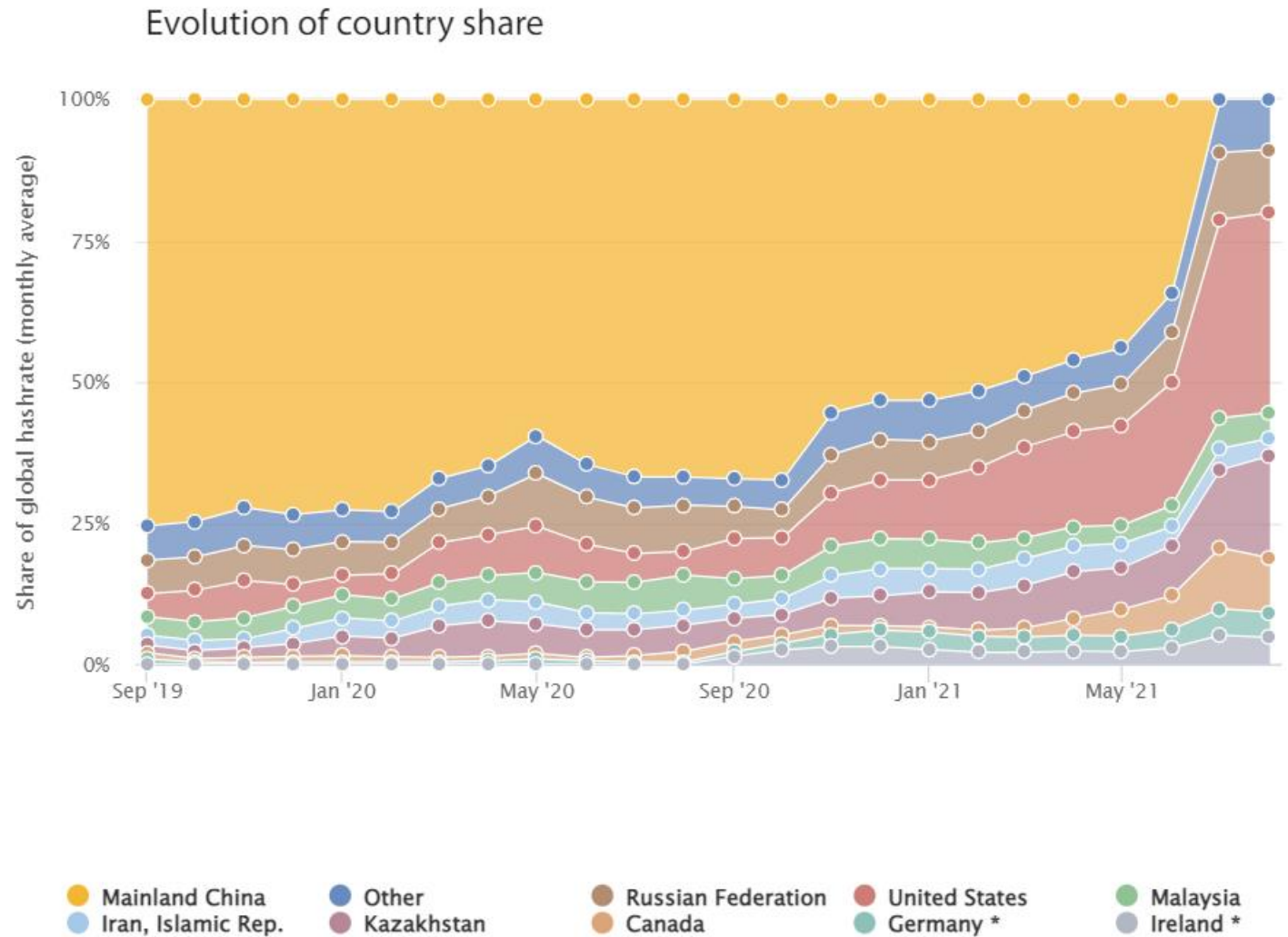
Source

U.S. Energy Information Administration, [Country Data](#), 2019 est. (or most recent available year)



Some Numbers

In **May 2021**, Chinese authorities ordered a **crackdown** on crypto mining and trading.



Beyond Bitcoin

Smart Contracts, dApps, DAO, NFTs, Oracles



Bitcoin Open Issues

Bitcoin and its blockchain are **problematic** under several aspects:

- **Non-technical issues:** need for huge storage capabilities, huge energetic consumption, concentration of computational power on few nodes.
- **Technical issues:** limited scalability, little data expressiveness, impossibility to express data relationship, lack of standardization, etc.



Technical Issues

- **Limited scalability:** the number of transactions per second is limited by the block size and by the block time.
- **Low efficiency:** At least 1h to safely confirm a transaction.
- **Data expressiveness:** in Bitcoin, data can be only economic transactions. Data relationships are hard to express.
- **Lack of standardization:** there are several variants of Bitcoin and cryptocurrencies, each with specific features.



Beyond Bitcoin

To overcome above open issues, starting from **2014**, with the public launch of the **Ethereum platform**, new blockchain and cryptocurrency implementations have started to appear.

In particular, they introduce some **new concepts** like:

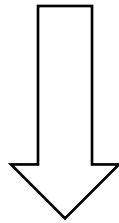
- **Smart Contracts.**
- **Decentralized Applications (dApps).**
- **Decentralized Autonomous Organizations (DAO).**
- **Non-fungible Tokens (NFTs).**
- **Oracles.**



Smart Contracts (SCs)

Bitcoin has a **scripting language**, but is **not Turing-complete**, so its functions are very limited.

It would be great to be able to **run any kind of code** (written in a Turing-complete language) exploiting the blockchain infrastructure.



A **Smart Contract** is a **computer program** stored and executed on a blockchain-based platform.

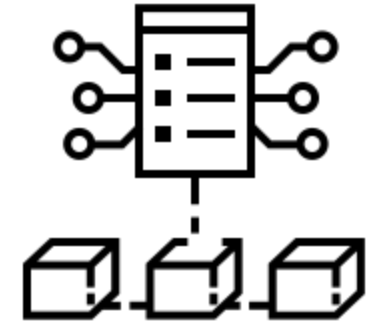
Stored on the blockchain = **Each node** in the network has a copy of it.



Smart Contracts

Having the blockchain as **enabler**:

- **no** need for **intermediaries** for execution.
- **distributed** execution.
- **fault tolerant**.
- SC and the data it contains are **immutable**.
- full **traceability** of the operations carried out on SC.
- **complete history** of the state of SC.



Smart Contracts

Each contract has an **address** that uniquely identifies it, **functions** that can be invoked and an **internal state** (e.g. the value of a variable at a given moment in time).

The internal state can be calculated at any time by **re-executing** the entire transaction history.

```
car contract:

query(car):
    get(car);
    return car;

transfer(car, buyer, seller):
    get(car);
    car.owner = buyer;
    put(car);
    return car;

update(car, properties):
    get(car);
    car.colour = properties.colour;
    put(car);
    return car;
```



Smart Contracts Lifecycle

- The first thing to do is to **deploy** the SC on the blockchain. The SC is thus distributed to **all nodes**.
- To **execute a function** it is necessary to perform a **transaction** to the address of the SC, specifying the function name and arguments.
- Each node **independently executes** the operations defined by the contract on its local copy.
- All of them have to get the **same result** to reach the consensus. **Otherwise** the transaction is **not valid**.



Smart Contracts Limitations

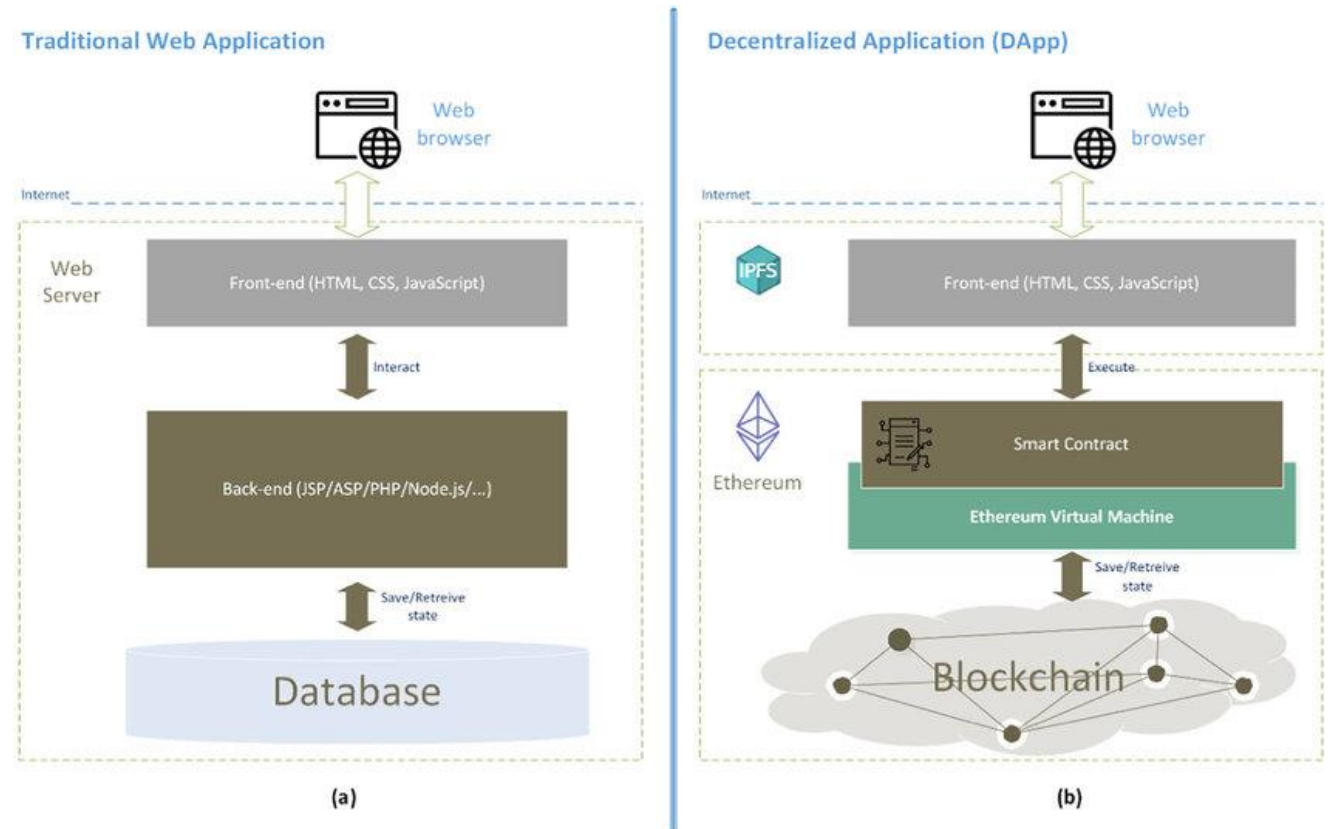
- In order to **achieve consensus**, it is essential that the SCs have a **deterministic execution**.
- A SC, contrary to what is often said, **cannot self-execute itself**. A transaction must be **explicitly** made to it.
- The **immutability** of the SC **prevents adding** functionality and/or **correcting** errors. To upgrade an SC, it is necessary to **deploy a new SC** and make sure that everyone uses it.
- Since the source code is **public**, anyone can easily **exploit the vulnerabilities**.
- The only data a SC can use is that **already present on blockchain**.



Decentralized Applications (dApps)

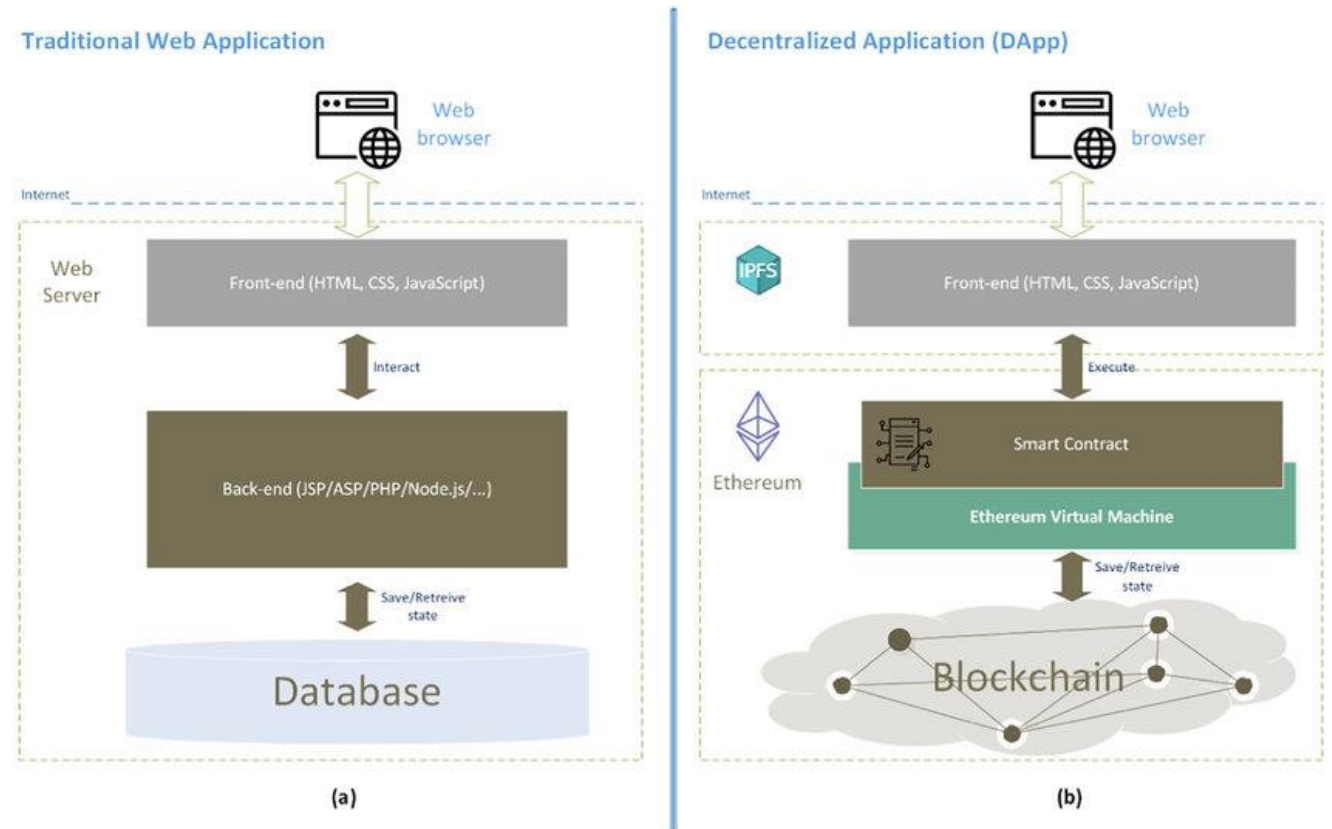
A decentralized application (**dApp**) is an application built on a **decentralized network** that **combines** a **smart contract** and a **frontend user interface**.

- **Frontend**: is the interface that users use to interact with the application, e.g. a **Web interface**.



Decentralized Applications (dApps)

- **Backend:** is the heart of the application, which implements the **application logic**. Is realised through **one or more** interacting **SCs**.
- **Database:** data storage is fully **decentralized** thanks to the use of the underlying blockchain.



Benefits of dApp Development

- **Zero Downtime:** thanks to the properties of the blockchain.
- **Privacy:** you don't need to provide real-world identity to deploy or interact with a dApp.
- **Resistance to Censorship:** no single entity on the network can block users from submitting transactions, deploying dApps, or reading data from the blockchain.
- **Complete Data Integrity:** data stored on the blockchain is immutable and indisputable.
- **Trustless Computation/Verifiable Behaviour:** SCs are public and deterministic.



Drawbacks of dApp Development

- **Maintenance:** the code and data published to the blockchain are difficult to modify.
- **Performance and Scalability:** huge performance overhead and low scalability due to consensus algorithms.
- **External Services:** sometimes some applications need external services to function. This eliminates many of the advantages of using the blockchain.



Decentralized Autonomous Organisation (DAO)

A Decentralized Autonomous Organisation (**DAO**) is an organisation constructed by **rules encoded as a computer program** built on top of a **blockchain**.

- **Controlled** by the organization's **members**.
- **No centralized leadership**.
- **Smart Contracts** are used to regulate the functioning of the organization.

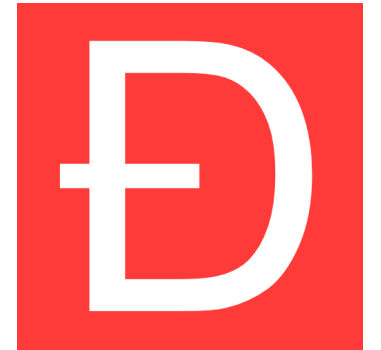
One of the pillars of **algocracy**: computer algorithms applied to **regulations, law enforcement**, and generally **any aspect of everyday life**.



The DAO

The DAO (stylized **Ð**) was a digital decentralized autonomous organisation and a form of investor-directed **venture capital fund** launched in 2016 on the **Ethereum** blockchain

- **Crowdfunding** campaign through a **token sale** (100 Dao token = 1 Ether).
- **1 token = 1 vote.**
- The DAO raised **\$150 million USD** worth of Ether.



The DAO

- Token holders could make a **proposal**, e.g. invest a certain amount of dollars in a project.
- Once the proposal was verified, all other investors **could vote on it**.
- After a certain period of time, **if** the 20% **quorum was reached**, the Ethers were **automatically deposited** in the SC related to the proposal.
- Any Ether generated by that proposal was then **put back** into the fund.
- In this way, the value of the Dao token could **increase** (in case of profits) or **decrease** (in case of losses).



splitDAO()

To protect **minorities** who did not want to spend their money on projects they did not believe in, it was implemented a system to **split the DAO** and transfer their funds into the “**child DAO**”.

- If no one decided to participate in the split, the proposer could **withdraw** its funds.
- **However**, the function suffered from a **vulnerability** related to **recursive calls**.

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {  
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])  
        throw;  
    uint reward = (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];  
    if (!rewardAccount.payOut(_account, reward)) // XXXXX vulnerable  
        throw;  
    paidOut[_account] += reward;  
    return true;  
}
```



The DAO Hack

- On 8 June 2016, a user made a **split proposal** called “lonely, so lonely”.
- No one joined in, so he was the only participant in the split.
- He **created a malicious script** so that it would **recursively** call the split function before updating his balance.
- He withdrew about **\$50M** from the fund.
- However, **a period of time** was needed to release the funds.



The Fork

The hacker **did not** commit any act **against the law**. The Smart Contract had a **vulnerability**!

- Eventually, the Ethereum network was **hard forked** to move the funds in The DAO to a recovery address where they could be exchanged back to Ethereum by their original owners.
- However, some continued to use the **original unforked Ethereum** blockchain, now called **Ethereum Classic**.



Non-fungible Tokens (NFTs)

A non-fungible token (**NFT**) is a **non-interchangeable** unit of data stored on a blockchain.

- The main use is to **digitally represent/identify** a physical entity.
- NFT ledgers claim to provide a **public certificate of authenticity** or **proof of ownership**.
- **Creation** and **transfer** via blockchain **transactions**.
- Legal rights conveyed by an NFT can be **uncertain**.
- The trading of NFTs in 2021 increase to more than **\$17 billion**, up by **21,000%** over 2020's total of \$82 million.



The Ukraine DAO

The **Ukraine DAO** is a **fund** to help Ukraine following the Russian invasion.

They decided to auction an **NFT** representing the **flag of Ukraine**.

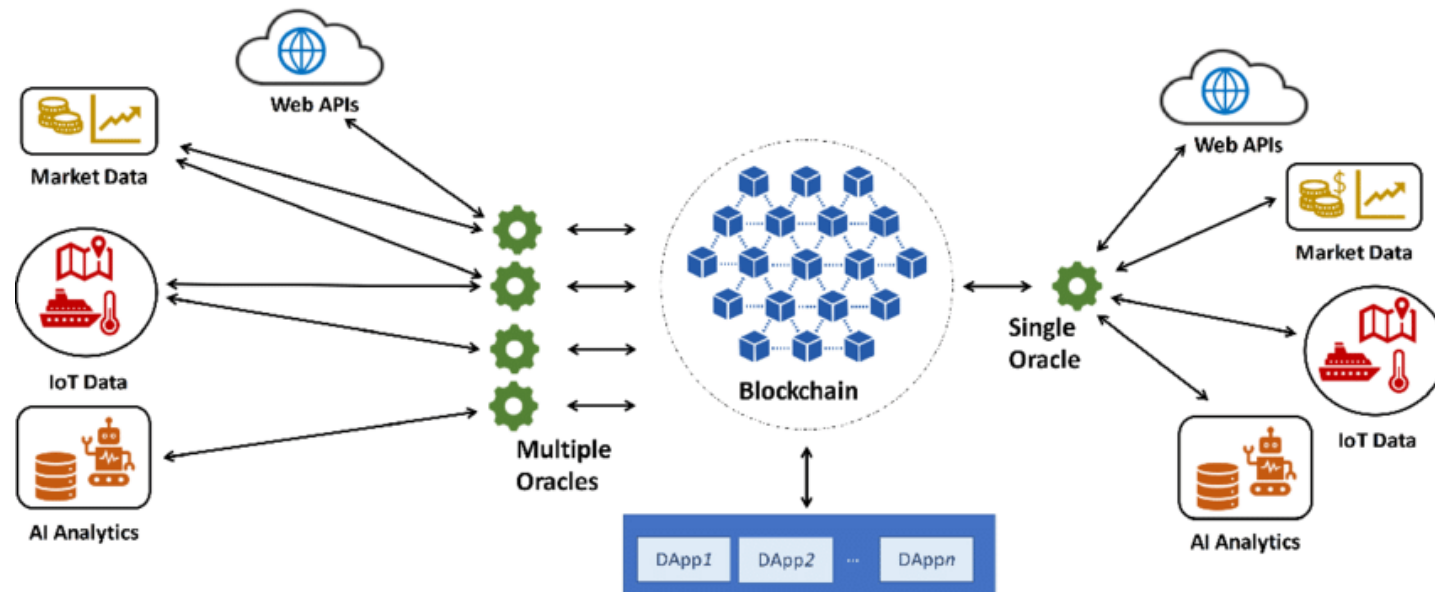
They raised more than **\$6M** in Ether.



Oracles

Blockchain oracles are entities that connect blockchains to **external systems**, thereby enabling smart contracts to execute based upon inputs and outputs **from the real world**.

However, oracles can be **single-points-of-failure** and **bottlenecks**, or prone to **man-in-the-middle attacks**.



Randomization

Thanks to an oracle, it is possible to obtain “**random**” behaviour within a Smart Contract.

Instead of generating a random number within the SC, the SC **requests** it from the oracle.

The oracle **will make sure** to return the **same value to all SCs** that are executing the same function.

Everyone will get the same result!



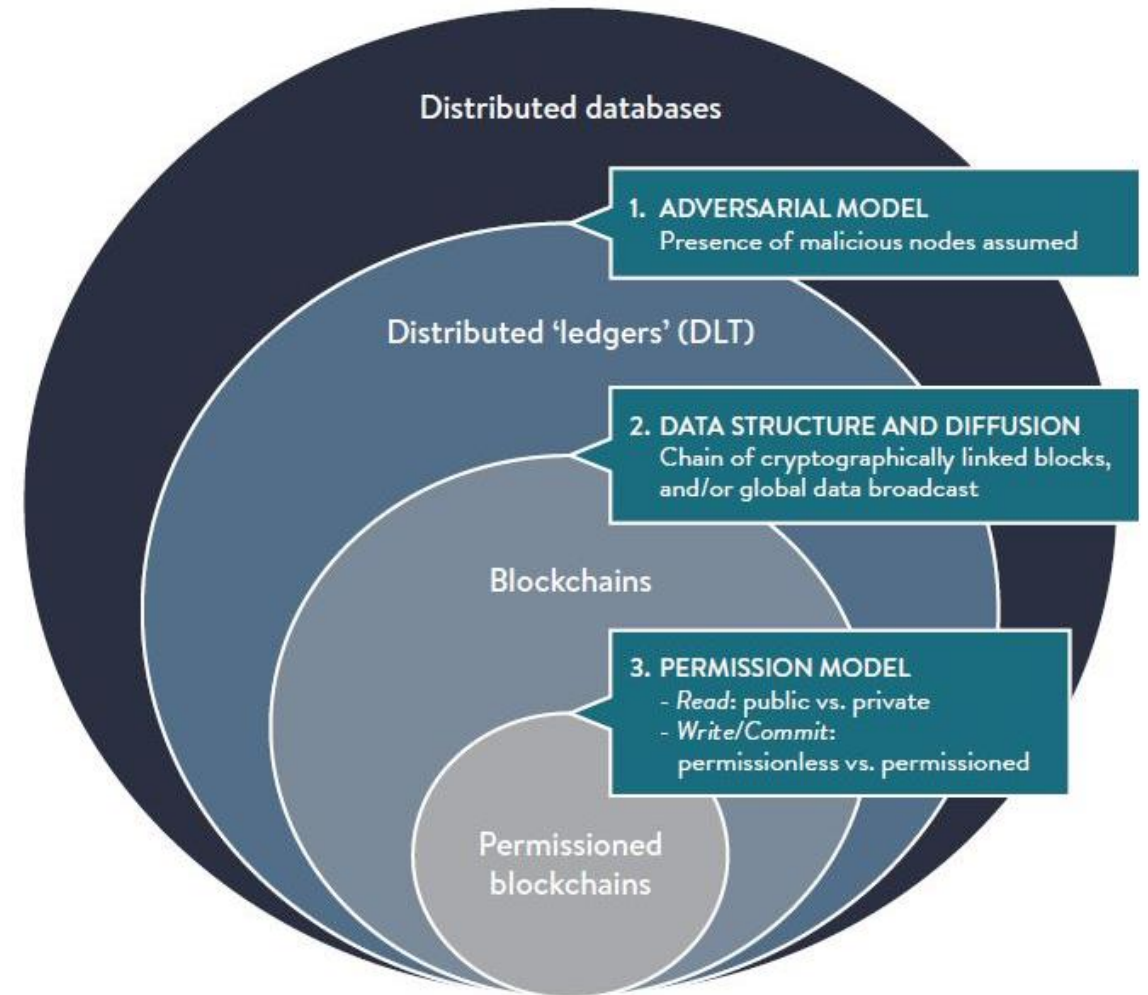
Distributed Ledgers Taxonomy



Distributed Ledger Technology (DLT)

A **distributed ledger** is a **replicated, shared, and synchronized** database geographically spread across multiple sites.

- A **peer-to-peer** network is required as well as **consensus algorithms** to ensure replication across nodes.
- The blockchain is actually a type of **distributed ledger** organised as a **chain of linked blocks**.

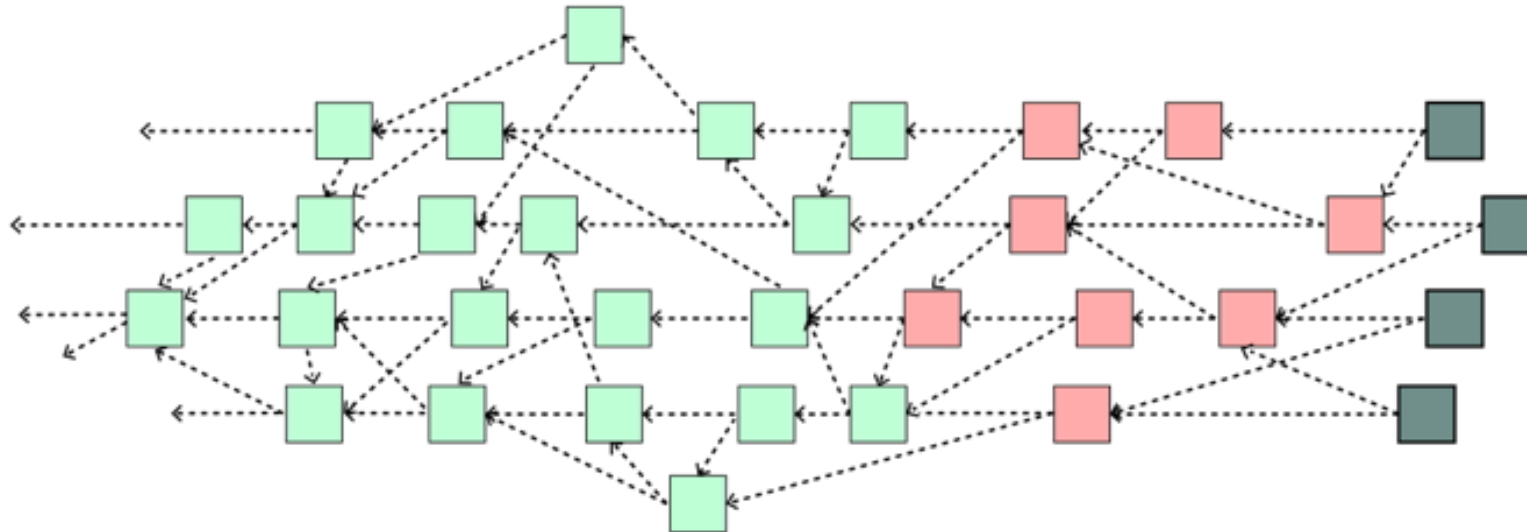


Source: Global Blockchain Benchmarking Study, by Dr Garrick Hileman & Michel Rauchs, 2017 - University of Cambridge

Other Types of DLTs

The IOTA **Tangle** is an innovative type of distributed ledger technology (DLT) that is specifically designed for the Internet of Things (**IoT**) environment.

It is a particular kind of **directed acyclic graph**, which holds **transactions**. Each transaction is represented as a **vertex** in the graph.



Permissionless vs Permissioned

- **Permissionless:** open network available for anyone to interact and participate in consensus validation. Fully decentralized across unknown parties.
- **Permissioned:** closed network. Designated parties interact and participate in consensus validation. Partially decentralized.



Permissionless – Pros & Cons

Advantages

- Broader decentralization.
- Highly transparent.
- Censorship resistant.
- Security resilience.

Pitfalls

- Less energy efficient.
- Slower and difficult to scale
- Less user privacy and information control.



Permissioned – Pros & Cons

Advantages

- Stronger information privacy.
- Highly customizable.
- Faster and more scalable.

Pitfalls

- Limited decentralization.
- Less secure.
- Less transparent to outside oversight.



Tokenized vs Tokenless

Tokenized ledgers: transactions involve some kind of purely digital asset (**token**) represented within the ledger.

Tokenless ledgers: distributed ledgers which **do not offer any asset as incentive** for being part of it or expect no payment for deploying smart contracts.

- They are typically **permissioned**, and thus strong trust has already been established during the registration process.



Tokenised Ledgers

Tokens typically serve **two** main purposes:

- They act as an **economic incentive** for the protocol participants to form consensus in **decentralized systems**:
 - Forming economic incentives within a consensus protocol **is really only relevant for decentralized systems** (a miner who mines a block outside of the consensus will waste power and forego any reward).
- They help preventing **spam** and **DoS attacks**:
 - Every operation carries a nominal fee to be executed.
 - Spammers can be thwarted by the enormous costs involved in creating a large number of transactions.



The Whole Taxonomy at a Glance

		Distributed		Localized
		Decentralized	Centralized	Centralized
Permissioned	Tokenized	<i>pointless</i>	<i>appl. unclear</i>	WoW Gold
	Tokenless		FINANCIAL APPL	Wikipedia
Permissionless	Tokenized	BITCOIN, ETHEREUM	RIPPLE, STELLAR	<i>appl. unclear</i>
	Tokenless	<i>subject to spam</i>		