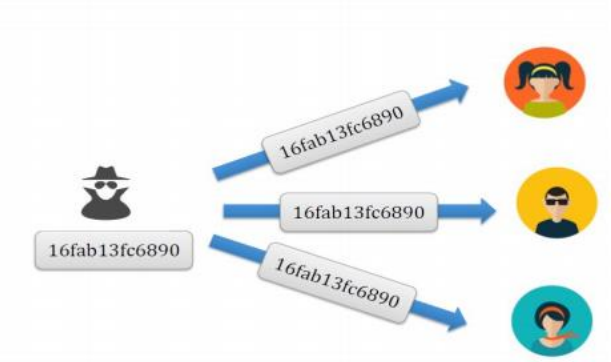
Blockchains

Basic Concepts:

- **A ledger** is like a bulletin storing operations (in order). Its properties are that it's an append-only list of events, is tamper-proof and that everyone must agree on its content (consensus). If it's organized as a list of blocks, it's called a blockchain. *Host it in the cloud*
- **Consensus** is the mechanism that defines who decides which operations will be added to the blockchain. It's an agreement on the same value: we call validity the act of agreeing on someone's proposal.
- **Tamper free** is the main characteristic requested by a ledger. To achieve it, compute the hash of each entry (block) and store in each entry the predecessor's hash. If one entry is tampered, it needs to recompute the hash of all the following ones (which will be computationally hard -> proof of work).

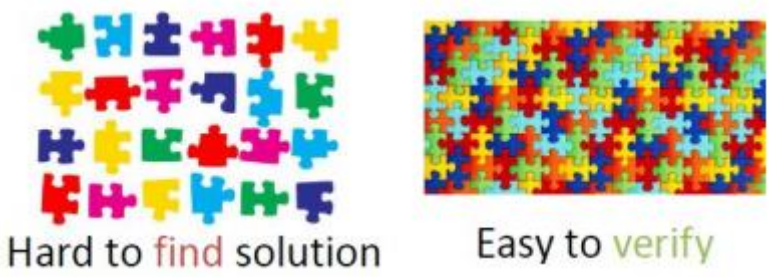
Several challenges in Blockchains: consistency must be maintained in the presence of different delays and since it's based on consensus, someone may cheat. This requires that the **majority must be honest**. Then we must prevent malicious users from **double-spending** (and other problems, like Sybil attacks, help this!)



So we must **define "majority"** in a context where everybody can easily join the network and assume multiple identities: counting won't be the way to go. We then use **the computing power controlled by each identity** (multiple ids in the same machine is not useful).

Proof of Work

The voting scheme must be hard to fake, so we introduce the PoW



the winner of the lottery will then decide which is the next node of the blockchain. "tickets are very expensive, the winner is paid later when validity is endorsed". This makes Sybil attacks pointless.

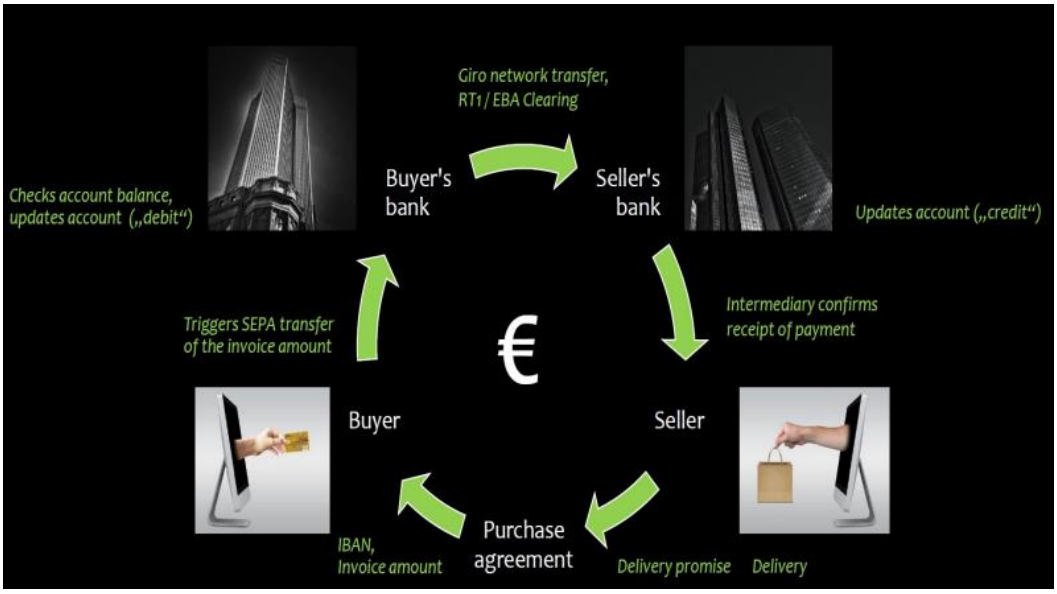
Since it's distributed, it may (rarely) happen that more winners are elected and this causes **forks**. Once the two chains grow, one of the two will eventually be longer and that one will be selected.

So all entries of the chain start as provisional and become stable merging over time.

Cryptocurrency

"virtual" currency that can be used for digital payments.

The traditional way of digital paying is a <u>trusted server</u> with <u>high transaction fees</u> and <u>no anonymity</u>.



Both parties trust their bank and banks trust each other.

Otherwise, there were less traditional ways that use an <u>intermediary service</u> *(like PayPal)*.

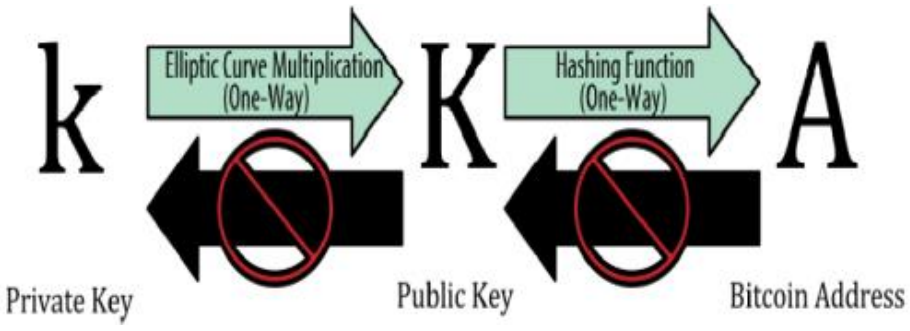
But then, without intermediaries..

➔ **Bitcoin** *https://bitcoin.org/bitcoin.pdf*

Purely p2p version of digital cash, no controlling authority and no server. A completely untrusted environment, which leads to the challenge of identifying evil intentions and finding consensus. Through **openness,** we do not need a trusted server, just a Bitcoin client. The **anonymity** is a **pseudo** one, and transactions are way smaller.

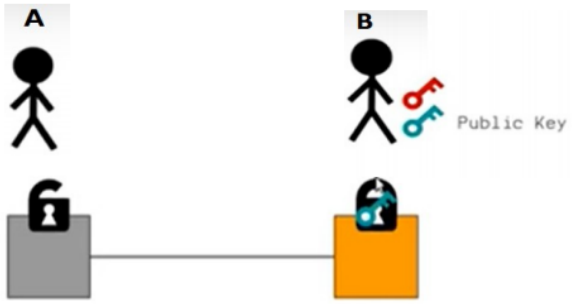
Decentralized Identity Management is used to represent **(addresses)**, creating random key-pairs

- **SK private (secret) key**
- **PK public key**: public name of a user, it speaks for the user's identity

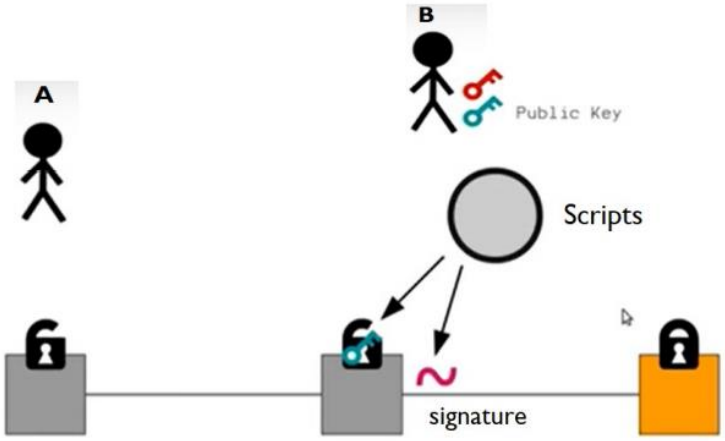


This means that anybody can make new identities at any time!

Note: in some cases addresses mya also represent scripts. They're written in a simple non-Turing complete language, FORTH *(can't create infinite loops, must be validated via execution)*. Bitcoins get then locked/unlocked



- B notifies its public key to A
- A sends its bitcoin to the public key of B
  - unlocks its bitcoin, that were locked by a previous transaction
  - create a lock on the bitcoin sent to B
    - the lock contains the public key of B
  - only B it will be able to unlock the bunch of received bitcoins, with its private key



- Green publick key (lock)
- Red is signature (unlock)
- Scripts are used to specify the locking and unlocking code

Since all the information needed to execute a script is contained within the script, there is no state before execution (nor state saved after execution), so it's **stateless**. It's also **deterministic**, since it will execute the same way on any system, and it's also simple and compact.

Transactions

Bitcoin only records transactions, it's not account-based like centralized currencies. To find the total amount owned by an address, search the blockchain for its unspent output addresses.

Its lifecycle is that it starts with its creation, it's signed with one or more signatures indicating the authorization to spend the funds referenced by the transaction and then broadcasted on the bitcoin P2P network. Each **participant** (node in the network) validates and propagates the transaction until it reaches *(almost)* every node in the network. The transaction is **verified** by a mining node and included in a block of transactions recorded on the blockchain. Once there are sufficient subsequent confirmations, the transaction is a permanent part of the blockchain.

Escrow transactions (deposito in garanzia)

A wants something B has, B has to ship it to A. None of them wants to do the first move, because they do not trust each other -> introduce a third party that through a 2-of-3 multi-sig transaction solves the problem, **if** the third party doesn't collude. If all goes well, 3rd pt won't even be useful, otherwise it acts as a "judge".

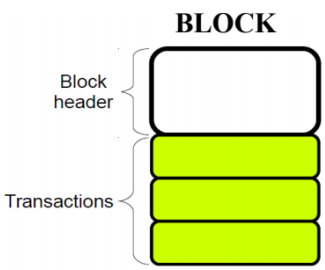
Note: if micropayments, combine all the small payments into a big payment at the end to not write many transactions in the blockchain.

- ➔ **Proof Of Burn**
- ➔ **Smart Contracts**

Nakamoto Consensus

There's an implicit approach to consensus: no voting and no collective message passing algorithm executed by the nodes, causing eventual consistency.

Double spending may cause inconsistencies in the ledger, so we introduce a MemPool *(collection of all bitcoin transactions awaiting confirmation)* where conflicts may happen and there's then a competition to add them. This means that the Nakamoto consensus is implemented like a lottery and this process is called **mining**.



Every miner will propose a block *(unit of work)* to be added to the ledger
The block header includes summary information, like the **version** of the protocol, **time, mhash, hashprev** (hash of all fields of the previous' block header), **target** and **nonce**.

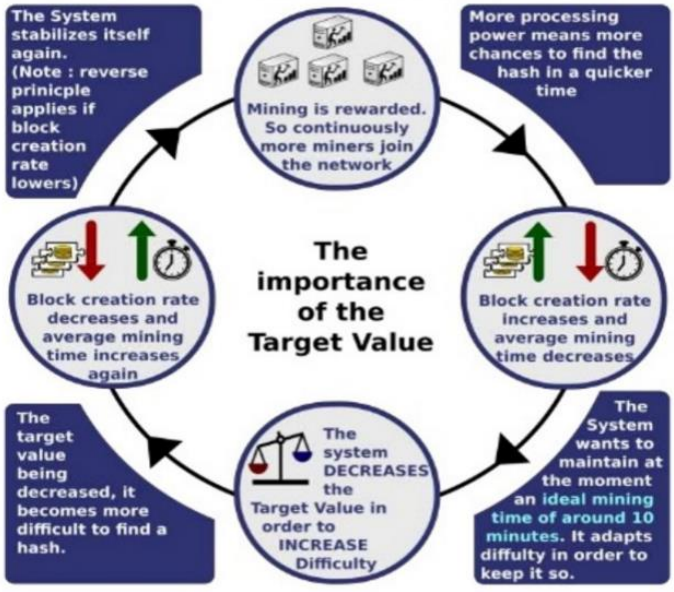
Mhash is the root of the Merkle tree of the transaction: it's built "on-demand". **Target** defines the required number of zeros at the beginning of the PoW hash. Blocks will then be linked through hash pointers, composing the blockchain.

The Nakamoto Consensus selects at each round a node at random and that one will unilaterally propose the next block (broadcasting it) to be inserted in the ledger: all nodes will check the validity of the block and update their blockchains with the new chosen block.

POW AND TAMPER FREENESS

- image an attacker changing a transaction in a block
  - the root of the Merkle tree changes and so the block header
  - the nonce of the block is no more valid
  - re-execute PoW to re-compute the right nonce for the new block
- in the next block the hash pointer to the previous block changes as well
  - the nonce of the next block is no more valid
  - the PoW has to be re-executed for the next block
- the pointer of the successo-successor must be changed and so on...
- an attacker would have to recompute the PoW for the entire chain
  - this would require an enormous computing power.

To make all work, the reward must dynamically vary in time (halve reward after 210000 blocks, 10 min each, scale difficulty [every 2016 blocks take into account the computational power of the network] to make it 10 mins nearly all the time).

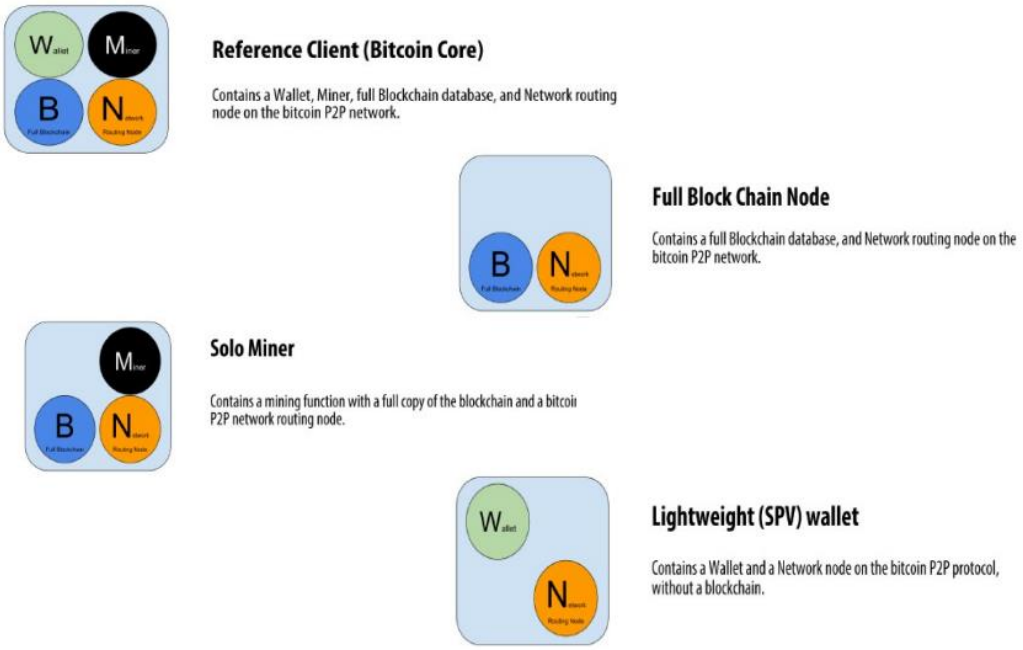


Two miners may win at the same time

➔ **Fork**, starts two different but legitimate branches. Longest wins

These type of forks are called temporary and aren't related to protocol upgrades (soft *["restrict the rules"]* if compatible with previous versions, hard *["loosen the rules"]* if not)

Actors



Mining blocks cost **a lot** in terms of power and time. That's why the solo mining approach is not that used, while **mining pools** are the way to go *(mutual insurance to lower risk)*. The pool may operate through a centralized mining operator *(must be trusted, uses a fee for itself)* or in a p2p way, using a private blockchain.

The pool manage must somehow know how much work each member of the pool has done and divide the revenue proportionally, even when participants may cheat

➔ **Mining shares:** proof of the work done by a miner, *"near-valid blocks", "near-optimal hashes"*

This leads to **Pay Peer Share:** instant payout guaranteed to the miner *(so workers may cheat sending invalid blocks*). So maybe **Pay Proportional:** every time a valid block is found, rewards from that block are distributed proportionally