Bitcoin.

Prior to 'emergence' of cryptocurrencies: commerce on the Internet relied on financial institutions (most often banks), which served as trusted third parties to process the payments.

Issues:

(1) transaction costs (not practical to transfer 5 cents, if you need to pay 10 cents for the transaction);

(2) most importantly, can we really trust the third party?

Most cryptocurrencies allow decentralized transaction: two willing parties can transact directly with each other without the need for a trusted third party.

The bitcoin system proposed by Satoshi Nakamoto is secure as long as honest nodes (participants) control more CPU power than any cooperative group (coalition) of attacker nodes.

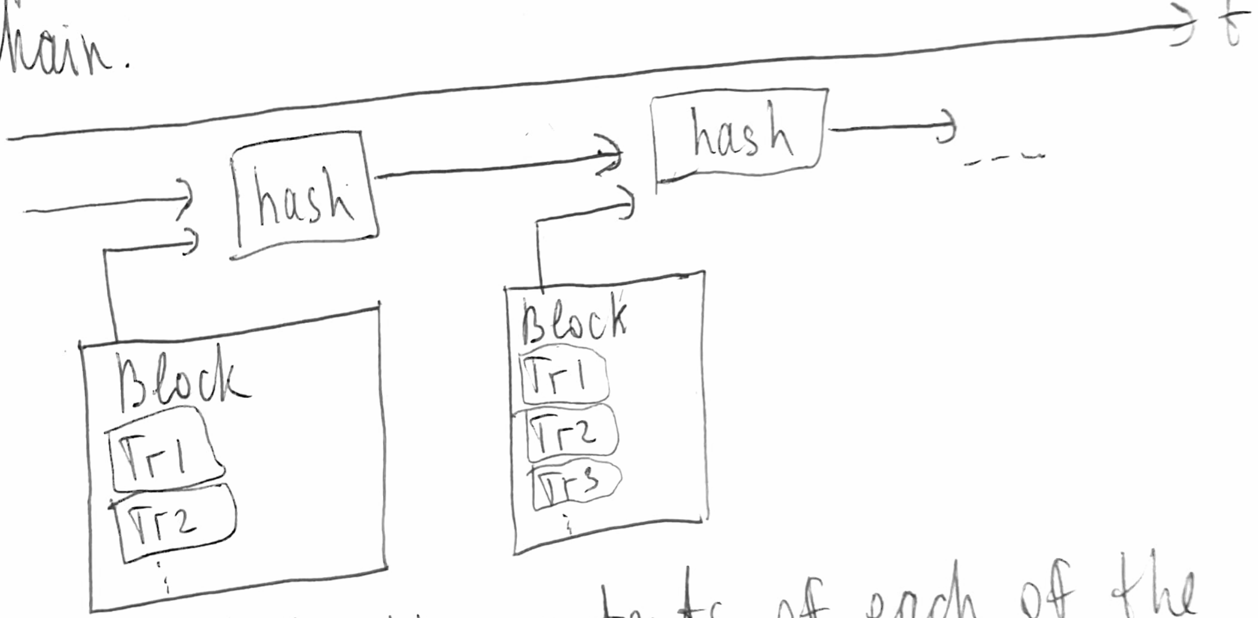
Def-n. An electronic coin is a chain of digital signatures. One owner transfers the coin to the next by digitally signing the hash of the previous transaction and the public key of the next owner, then adding the signature to the end of the coin.

Rmk. The digital signature is made with the ECDSA (elliptic curve digital signature algorithm), the curve E with defining equation $y^2 = x^3 + 7$ over $\mathbb{F}_p$ with $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ is known as secp256k1 (k for Koblitz). The hash function is SHA256 and the range of this function consists of 256 bits. The image is represented in the hexadecimal system: digits 0-9 and letters

$a = 10$
$b = 11$
$c = 12$
$d = 13$
$e = 14$
$f = 15$.

Issue: double-spending, i.e. the same digital coin can be sent to multiple users.

Resolution: add timestamps (timestamp server). Such a server works by taking a hash of a block of items (transactions) and hashes with a timestamp, then distributing the hash. Each timestamp includes the previous timestamp in its hash, thus forming a chain.

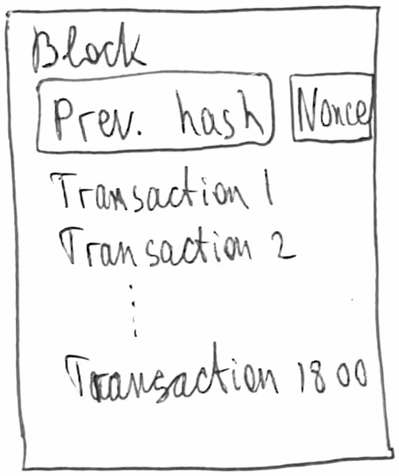


Let's describe the contents of each of the blocks in the blockchain. We will need to define a very important 'ingredient'.

Def-n. A nonce (number only used once) is a number added to a hashed block in a blockchain, so that

$$h(h(\text{block}) \| \text{nonce}) < \text{target}, \text{ where}$$

target is a chosen number, which usually starts with a collection of zeros. The nonce is allocated 32 bits of memory. (~20).



How does the network run?

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. (Each) node tries to find a nonce, so $h(\text{block} \| \text{nonce}) < \text{target}$
4. When a node finds such a nonce ('demonstrates' proof-of-work), it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain using the hash of the accepted block as the previous hash.

<u>Rmk.</u> Nodes always consider the longest chain to be the correct one and keep working to extend it. If two nodes broadcast different versions of the next block simultaneously, some nodes receive one and some receive the other. Each node works on the blockchain with the block that it received, but saves the other branch (fork) in case it becomes longer. The tie will be broken, once the next proof-of-work is found and one branch becomes longer.

Incentive.

Proof-of-work requires a lot of CPU power (expenditures) for 'the people behind the nodes'. In order to motivate the nodes to propose new blocks, the first transaction in every block starts a new coin owned by the creator of this block.

<u>Rmk.</u> The process of finding a proof-of-work is also known as <u>mining</u> (in analogy with gold mining), the nodes participating in mining are called <u>miners</u>.

Honest chain vs attacker chain.

Next we would like to find the probability that an attacker catches up with the original blockchain given that his blockchain is n steps behind.

Thm. Let p be the probability that an honest node finds the next block first and q the probability that a 'bad' node (an attacker) does. We denote the probability that an attacker catches up from an n step deficit by p_n. Then

$$p_n = \begin{cases} 1, & p \leq q \\ \left(\frac{q}{p}\right)^n, & p > q. \end{cases} \qquad (\star)$$

Rmk. Notice that some node must produce the next block, hence, $p+q=1$ and $(\star)$ can be rewritten as

$$p_n = \begin{cases} 1, & p \leq \frac{1}{2} \\ \left(\frac{1-p}{p}\right)^n, & p > \frac{1}{2}. \end{cases}$$

Proof. We will first find $l_n := 1 - p_n$, the probability that the attacker doesn't catch up from an n step deficit (this is easier for technical reasons). We will adhere to the following strategy.

Step 1. Find l_n^t, the probability that an attacker fails to catch up from an n step deficit with an additional assumption that he (she) gives up after falling t steps (blocks) behind.

Step 2. Compute $l_n = \lim_{t\to\infty} l_n^t$ and $p_n = 1 - l_n$.

Notice that $l_0^t = 0$ (the attacker successfully catches up) and $l_t^t = 1$ (the attacker gives up). We also have the relations

$$l_n^t = p\, l_{n+1}^t + q\, l_{n-1}^t . \quad (*)$$

(the next gap is $n+1$ with prob. p or $n-1$ with prob. q)

Using that $p+q=1$, $(*)$ can be rewritten as

$$(p+q)\, l_n^t = p\, l_{n+1}^t + q\, l_{n-1}^t$$

$$p(l_{n+1}^t - l_n^t) = q(l_n^t - l_{n-1}^t) \quad \text{or}$$

$$l_{n+1}^t - l_n^t = \frac{q}{p} \cdot (l_n^t - l_{n-1}^t) \qquad (**).$$

Henceforth we will denote $c := \frac{q}{p}$.

Then $l_0^t = 0$, $l_2^t - l_1^t = c(l_1^t - l_0^t) = c l_1^t$

$$l_2^t = (1+c) l_1^t$$

Next let's check that $l_k^t = (1+c+c^2+\dots+c^{k-1}) l_1^t$ by induction on k with the help of $(**)$:

$$l_{k+1}^t = (c+1) l_k^t - c l_{k-1}^t = \qquad (**) \text{ for } n=k.$$

$$\underset{\uparrow}{=} (c+1)(1+c+c^2+\dots+c^{k-1}) l_1^t - c(1+c+c^2+\dots+c^{k-2}) l_1^t =$$

(using induction assumption for k and $k-1$)

$$= (1+c+c^2+\dots+c^k) l_1^t \underset{\uparrow}{=} \frac{c^{k+1}-1}{c-1} l_1^t$$

(formula for the sum of geometric series)

Now we find l_1^t, using that $l_t^t = 1$:

$$l_t^t = \frac{c^t - 1}{c-1} l_1^t = 1, \text{ so } l_1^t = \frac{c-1}{c^t-1}, \quad l_n^t = \frac{c^n-1}{c-1} \cdot \frac{c-1}{c^t-1} = \frac{c^n-1}{c^t-1}$$

This completes the first step.

We consider 3 cases:

① $p<q$ (or $p<\frac{1}{2}$), $c=\frac{q}{p}>1$:

$$l_n = \lim_{t\to\infty} l_n^t = \lim_{t\to\infty} \frac{c^n-1}{c^t-1} = 0 \quad (c^t \xrightarrow[t\to\infty]{} \infty).$$

$p_n = 1-l_n = 1.$

② $p=q=\frac{1}{2}$, $c=1$.

$$l_n^t = (1+c+\dots+c^{n-1})\, l_1^t = n\, l_1^t$$

$$l_t^t = t\, l_1^t = 1, \text{ so } l_1^t = \frac{1}{t}.$$

$$l_n^t = n\, l_1^t = \frac{n}{t}$$

$$l_n = \lim_{t\to\infty} \frac{n}{t} = 0.$$

$p_n = 1-l_n = 1.$

③ $p>q$ (or $p>\frac{1}{2}$), $c<1$

$$l_n = \lim_{t\to\infty} \frac{c^n-1}{c^t-1} = 1-c^n \quad (c^t \xrightarrow[t\to\infty]{} 0)$$

$$p_n = 1-l_n = 1-(1-c^n) = c^n = \left(\frac{q}{p}\right)^n$$

$\boxtimes$

<u>Rmk.</u> If $p > \frac{1}{2}$ (the 'computational power' of honest nodes exceeds $\frac{1}{2}$), the probability of the 'bad nodes' catching up decreases exponentially with the number of steps they fall behind.

<u>Rmk.</u> The situation we discussed is very similar to the gambler's ruin problem in statistics. The first mention of this problem dates back to the letter from Blaise Pascal to Pierre Fermat in 1656. The analogue of our problem can be stated as follows. A gambler is down n dollars and has unbounded credit (can keep losing as much money as he wants). Each round of the game he either wins 1 dollar (with probability q) or loses 1 dollar (with probability p). The game ends if he breaks even.

Let's play out the following scenario. The sender is an attacker and would like to (secretly) create an alternative block, where he records the transaction of the coin to himself instead of the receiver. The attacker's hope is to make the 'fraudulent' continuation of the blockchain longer, so that it gets accepted.

The attacker transfers the coin to the recipient (the transaction is broadcasted to all nodes) and starts mining an alternative chain (with this transaction substituted by the one with him sending the coin to himself).

<u>Rmk</u>. The timestamp 'issue' will be bypassed if the attacker mines faster.

Let's assume that the receiver waited until n blocks were added to chain. How likely is he to have successfully received the coin?

The attacker's potential progress up to that point will follow a Poisson distribution with parameter (expected value) $\lambda = \frac{q}{p} \cdot n$

In order to find the probability that the attacker can still catch up, we compute

$$\text{catch up probability} = \sum_{k=0}^{\infty} \begin{pmatrix}\text{probability of mi-}\\ \text{ning } k \text{ blocks}\end{pmatrix} \cdot \begin{pmatrix}\text{probability of}\\ \text{catching up given}\\ \text{that deficit}\end{pmatrix} =$$

$$= \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} \cdot \begin{cases} c^{n-k}, & k \leq n \text{ (see the thm above)} \\ 1, & k > n \text{ (already ahead)} \end{cases} =$$

$$= \sum_{k=0}^{n} \frac{\lambda^k}{k!} e^{-\lambda} \cdot c^{n-k} + \sum_{k=n+1}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} = \sum_{k=0}^{n} \frac{\lambda^k}{k!} e^{-\lambda} \cdot c^{n-k} +$$

$$+ 1 - \sum_{k=0}^{n} \frac{\lambda^k}{k!} e^{-\lambda} \quad \Big(\text{we used that } \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} = 1\Big) =$$

$$= 1 - \sum_{k=0}^{n} \frac{\lambda^k}{k!} e^{-\lambda} \left(1 - c^{n-k}\right).$$

Examples:

$n \backslash q$	0.1	0.3
0	1	1
5	0.0009	0.177
10	0.0000012	0.04166

Rmk. Let T be a random variable representing the time interval between mining subsequent blocks (we assume that mining follows a Poisson process with parameter λ). Let's find the corresponding probability distribution.

$$F_T(t) := P(T \leq t) = 1 - P(T > t) = 1 - \underset{(\text{on } [0,t))}{P(X=0)} = 1 - e^{-\lambda t},$$

so $f_T(t) = F_T(t)' = \lambda e^{-\lambda t}$ is the pdf of our distribution.

Such a distribution is an exponential distribution with parameter λ.

The corresponding mean value is

$$\mathbb{E}(T) = \int_0^\infty \lambda \cdot t \cdot e^{-\lambda t}\, dt \underset{\text{integration by parts}}{=} -te^{-\lambda t}\Big|_0^\infty + \int_0^\infty e^{-\lambda t}\, dt =$$

$$= -\frac{1}{\lambda} \cdot e^{-\lambda t}\Big|_0^\infty = \frac{1}{\lambda}.$$