Bitcoin blockchain is a **decentralized, (peer-to-peer digital monetary system)** cryptocurrency
that uses cryptography to control its creation and management.
Bitcoin allows value to be transferred over the internet **without intermediaries** such as banks.
In 2008 Satoshi Nakamoto published the Bitcoin whitepaper: *"Bitcoin: A Peer-to-Peer Electronic Cash System"*

Bitcoin consists of three tightly connected layers:
1. **The currency (BTC)** – the unit of value, the smales unit of BTC is Satoshi: 1 BTC = 1 000 000 000 Sat.
2. **The network** – thousands-millions of computers (nodes) running Bitcoin software
3. **The protocol** – the rules that govern how Bitcoin works

Unlike traditional payment systems, Bitcoin removes trusted intermediaries such as banks and relies on:
- Blockchain
  - Cryptographic verification
  - Distributed consensus mechanism
  - Incentives for block mining (validation)

Fundamental Problem Bitcoin Solves
- Ownership verification
- Prevention of double spending
- Trustless operation

Before Bitcoin, the payment systems relied on:
User → Bank → Ledger → Payment Confirmation

This creates problems:
- central point of failure
- censorship
- trust dependency

Bitcoin replaces the central authority with cryptographic proof and distributed consensus mechanism.

Core Problem: Double Spending
Digital information can be copied two or many times. This creates the double spending problem.
Example:
Alice owns 1 BTC
Alice could make a copy of this bit string attempt to send the same coin to:
Bob and Charlie
Both transactions may appear valid unless a system ensures only one can succeed.

Bitcoin solves this problem using:

- ❖ **Cryptographic signatures and hash functions**
- ❖ **Public transactions ledger**
- ❖ **Timestamped blocks of transactions**
- ❖ **Proof-of-Work consensus mechanism**

Mathematical Foundations of Bitcoin
Bitcoin relies heavily on three mathematical domains:

Field	Role
Number Theory	modular arithmetic in finite fields
Cryptography	signatures and hashing
Probability Theory	mining security and attacks

Bitcoin ownership is controlled through public-key cryptography.

When a user spends coins, they must prove ownership of the private key corresponding to a public key embedded in the transaction output: performed by digital signature.
Bitcoin originally uses: Elliptic Curve Digital Signature Algorithm - ECDSA of **secp256k1** kindf of signatures and Hashing basedd on **SHA-256**

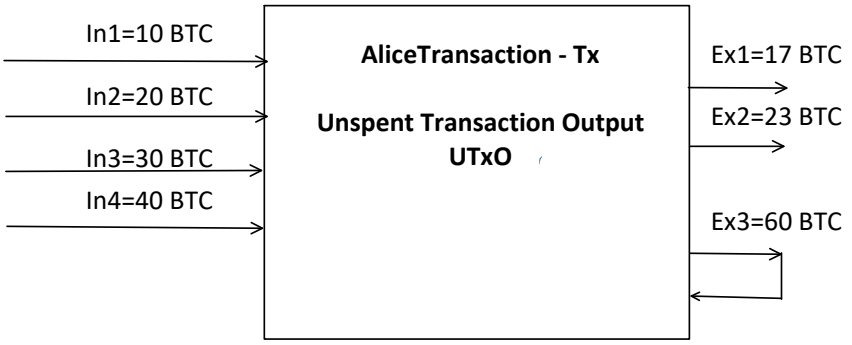
A cryptographic hash function H(x) must satisfy:
- Deterministic
- Preimage resistance
- Second preimage resistance
- ***Collision resistance***

SHA-256 Function
h = sha256(m)
Output:
256-bit digest
Bitcoin frequently uses double hashing:
h = sha256(sha256(m))

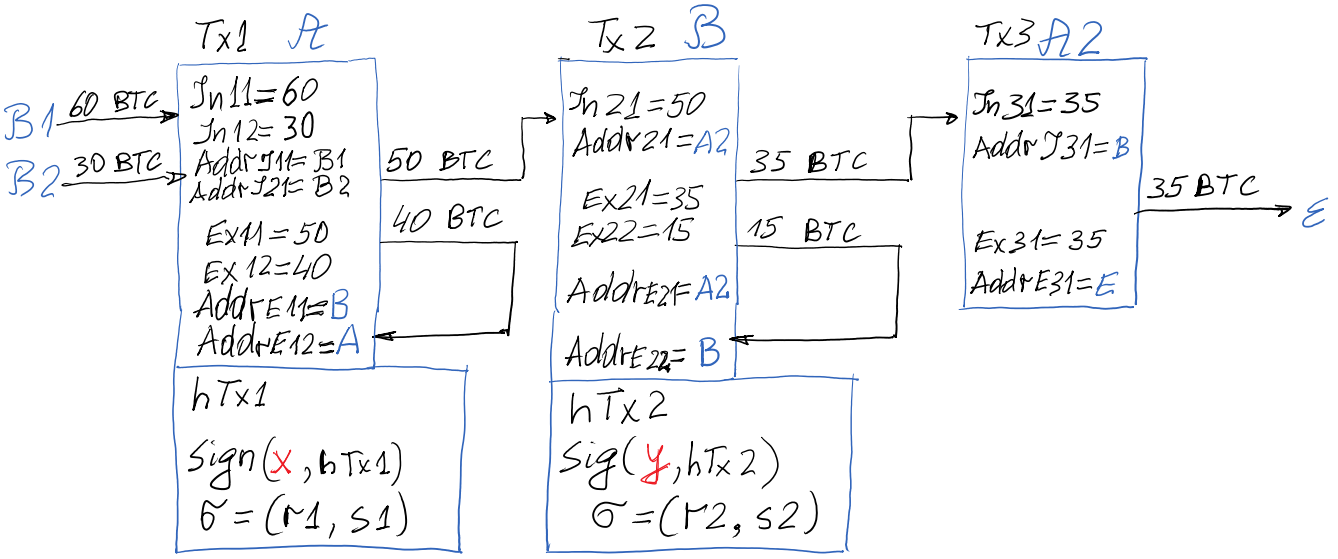
>> sha256('Hello Bob')
ans = 2D6A12FFC0A952FBD09F8909DE4E0E4B20BA2B906CD12AC22BBEF4EE5BD9003E
>> h=sha256(ans)
h = 476D03360CCD1CF6F25B2B4F8038E9B526BF6F0F7386F2781CA92D31DC362873

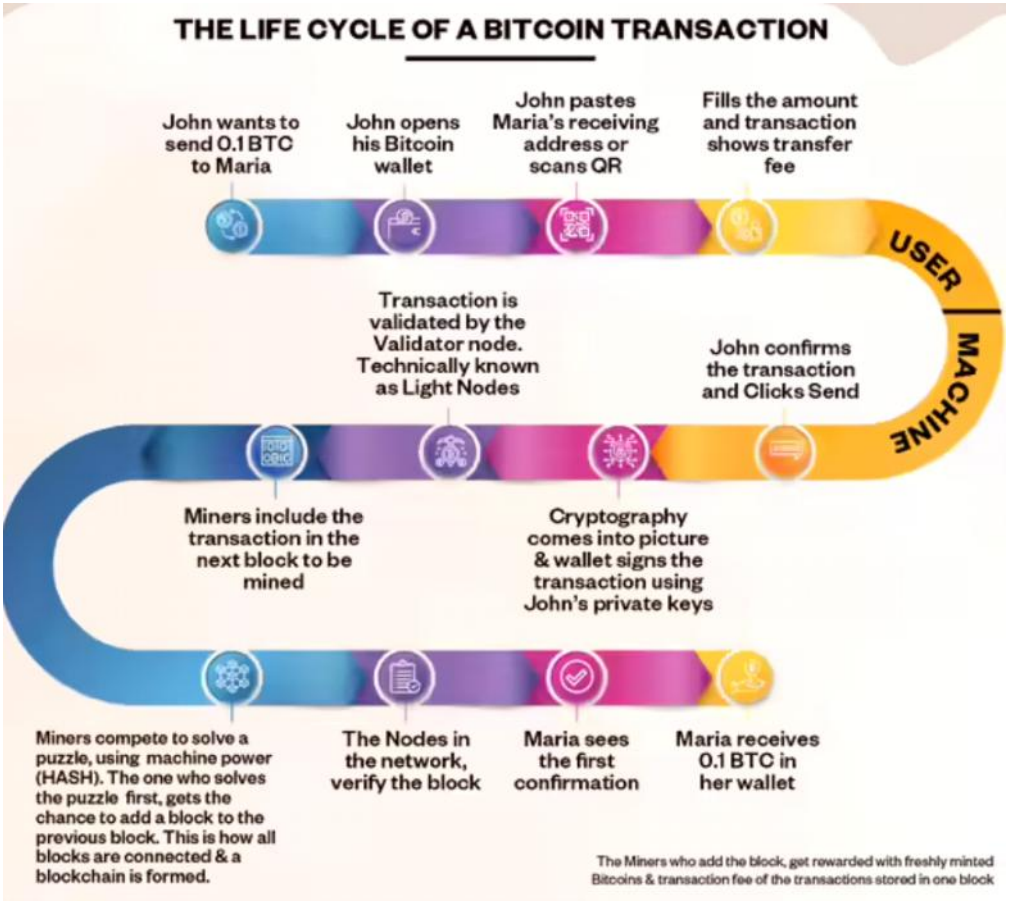
Hash function is used for:
- Address computation
- Transaction IDs
- Transactions signing
- Block hashing
- Proof-of-Work (PoW)

How does Bitcoin Works:
Bitcoin relies on blockchain and works through a sequence of processes combining cryptography, networking, distributed consensus mechanism, and incentives for mining.
Every time a transaction happens, it gets written onto a "block." That block is linked to the previous one, forming a chain. This record is copied onto thousands-millions of computers around the world (called **nodes**).
Also, anyone can participate in the ecosystem by downloading Bitcoin's open-source software and create bitcoin based applications.
Among the **nodes** there are **full node** in the Bitcoin network and thes **full nodes** independently stores a block chain containing only blocks validated by that node. When several nodes all have the same blocks in their block chain, they are considered to be in consensus. The validation rules of these nodes follow to maintain consensus are called consensus mechanism.
Bitcoin transactions are operating using so called **Unspent Transactions Output** - **UTxO** mechanism:
The sum of transaction Incomes (In) must be equal to the sum of transaction Expenses (Ex) or the sum of transaction Inputs must be equal to the sum of transaction Outputs.



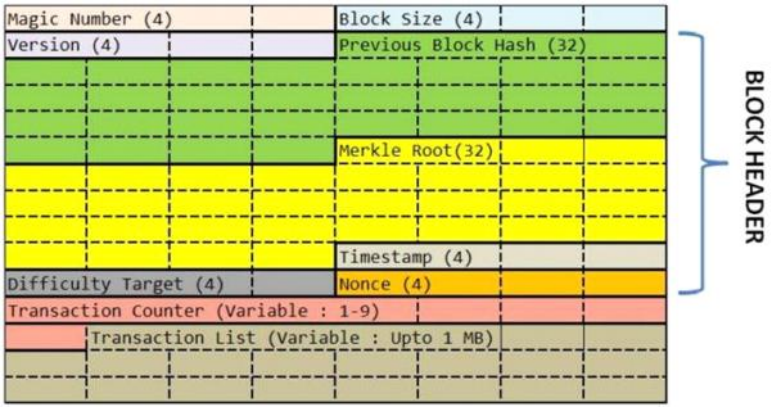
Transactions flow





Transactions are included in blocks

Bitcoin block structure

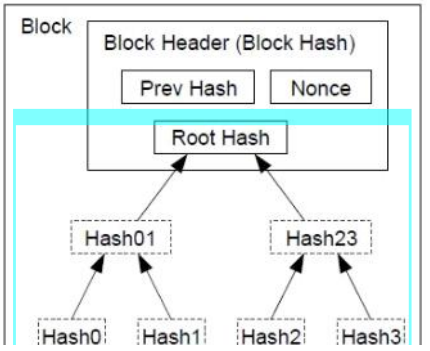


1 rectangular represents 1 Byte = 8 bits
32 Bytes == 64 hex numbers == 256 bits
Nonce is a number represented by 4 Bytes

Transactions list is represented by Merkle Tree which is represented by Root Hash.
This Root Hash is placed into the block's header along with the hash of the previous block and a random number called a nonce to be explained later.

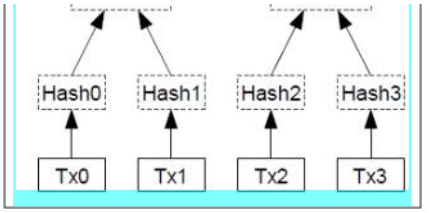
Merkle tree: is a graph consisting of leafs representing transactions data, graph vertices and edges.

Antraštė



representing transactions data, graph vertices and edges.

The block's header is then hashed with sha256 producing an output that will serve as the block's identifier.



Structure of a Bitcoin Transaction Blockheader.

Field	Purpose	Updated when...	Size (Bytes)
Version	Block version number	When software upgraded	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	A transaction is accepted	32
Time	Current timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current target in compact format	The difficulty is adjusted	4
Nonce	32-bit number (starts at 0)	A hash is tried (increments)	4

Bitcoin block mining ir based on **PoW** consensus mechanism referencing to Difficulty Target - **DT**.
Symbolically it can represented in the following way:
>> sha256('Block header data hashed using the arbitrary Nonce ...')
ans = 41F1801FC7E4B36D44DA242013CE7C6E8C111D19B5FB8795EF718A1030AD6980
Difficulty Target - DT defines the number of hexadecimal zero values required to mine a block.
For example **DT**=18.
This means that block is mined if the **Nonce** value is found such that h-value of block header has 18 leading hexadecimal zeroes.
Let such a **Nonce** is found, then the block is mind if e.g. miner obtains the following h-value:
>> sha256('Block header data hashed using the concrete Nonce of 4 bytes')
ans = 000000000000000000E7B8F4C1AAAC8D2D85A326BEE0B9E242E0E360B09C1347

The full lifecycle using Elliptic Curve Cryptography (**ECC**) can be divided into the following stages:

Key generation: is based on Elliptic Curve Digital Signature Algorithm - **ECDSA**
ECDSA Public Parameters are the following **PP**=(EC **secp256k1**; BasePoint-Generator ***G***; prime ***p***; param. ***a***, ***b***);
Prime ***p*** is of 256 bit length and defines the finite fielf $F_p = \{0, 1, 2, 3, \ldots, p-1\}$ where operations are performed **mod** ***p***.
Parameters ***a***, ***b*** defines the equation of Elliptic Curve (EC) over the finite fielf F_p:

$$y^2 = x^3 + ax + b \mod p$$

Private Key PrK = *z* generation.
Is generated at random as a hexadecimal number having 64 hexadecimal digits ***z*=randi(*p*-1)**
Where ***p*** - is a large prime having 256 bit length.
Public Key PuK = *A* generation.
Derived mathematically from private key **z**.
Bitcoin uses the elliptic curve: **secp256k1** it means that the number of Elliptic Curve (**EC**) points is equal to ***p***.

The public key is generated using **EC** generator ***G*** multiplication by number (scalar) ***z*** we denote by *:
PuK = *A* = *zG**
Public key ***A*** is an EC point and hence is represented by coordinates (*x*, *y*)
In **secp256k1** every coordinate ocuupies 256 bits.
Then The representation of ***A*** requires 512 bits.

Address creation
Private Key z → Public Key *A* (EC multiplication operation *) → sha256(*A*) → Bitcoin Address by taking 34 hexadecimal digits

Let user's **PuK**=(*x*,*y*) is represented by the following hexadecimal number
x = 52A6B8D367854D62EABF9F415A5E68505506727009472936417C747610270624
y = 3EF4698C496E1475B7ECB6A2E4C85B6C7C1C57AA124DFCA859AE3B6CBB3F4A75

To compute Address **PuK** coordinates *x* and *y* must be hashed
>> h=sha256('52A6B8D367854D62EABF9F415A5E68505506727009472936417C7476102706243EF4698C496E1475B7ECB6A2E4C85B6C7C1C57AA124DFCA859AE3B6CBB3F4A75')
h = 16A134BC52822E45372C9FE6411939B8B60747EAEA5FAF9AB40137FC97BD05F0

Bitcoin Address Example: B8B60747EAEA5FAF9AB40137FC97BD05F0