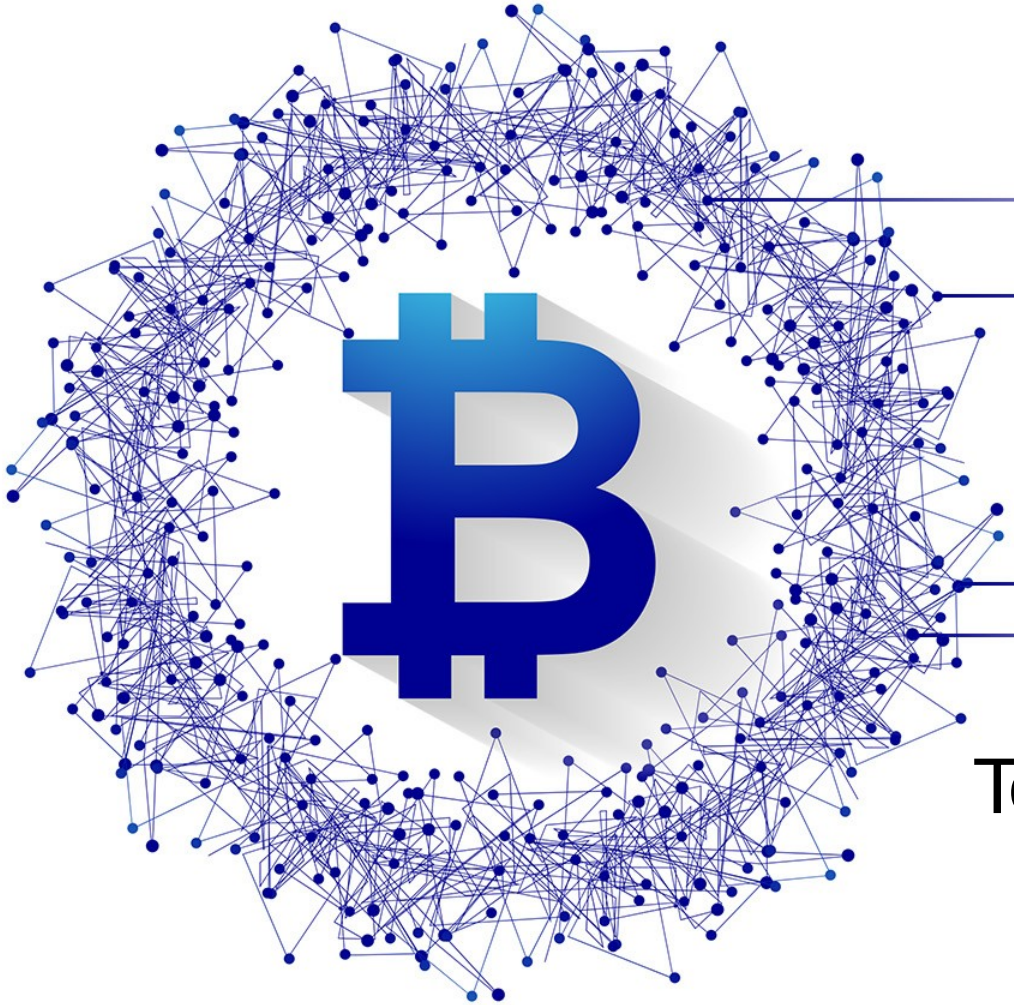
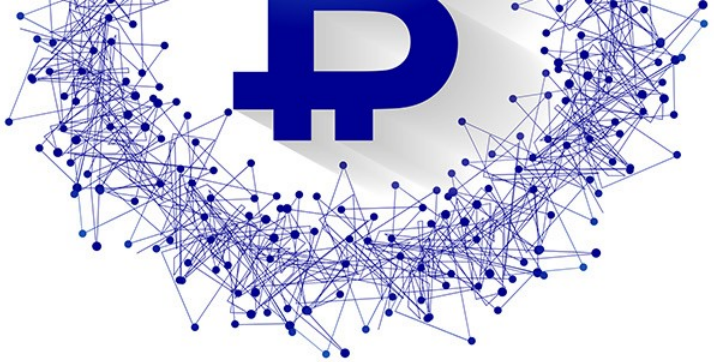


Exploring Bitcoin

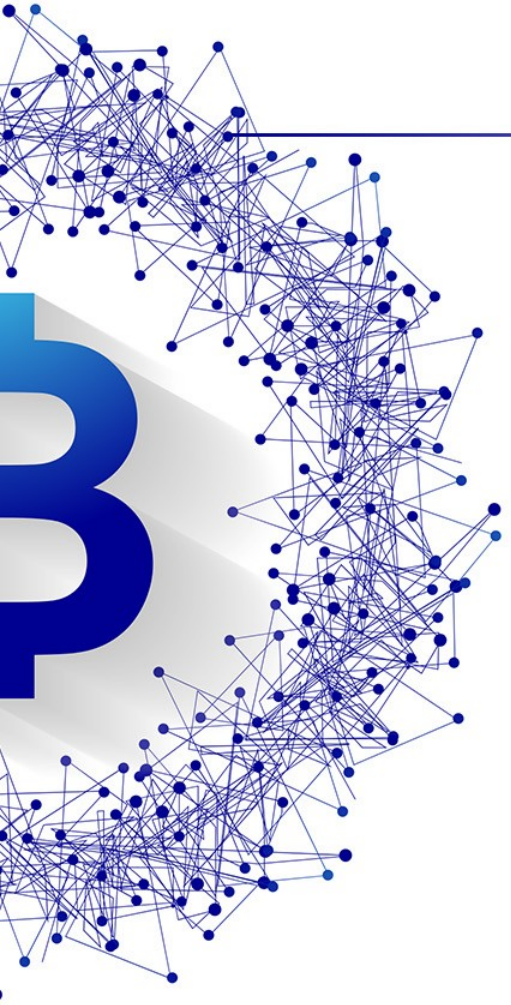
Tomasz Jadachowski

14.03.2019



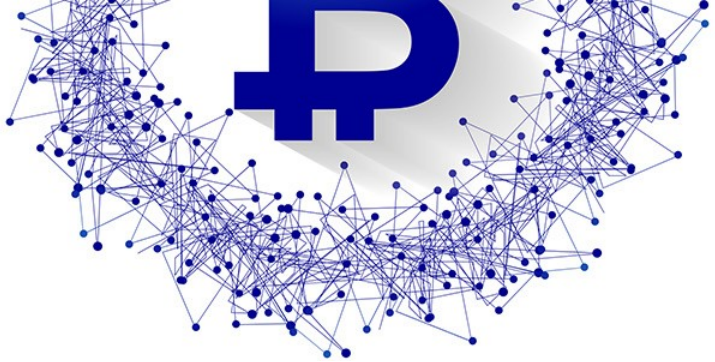
Content

- Bitcoin genesis
- Blockchain
- Mining
- Transactions
- Security
- Future

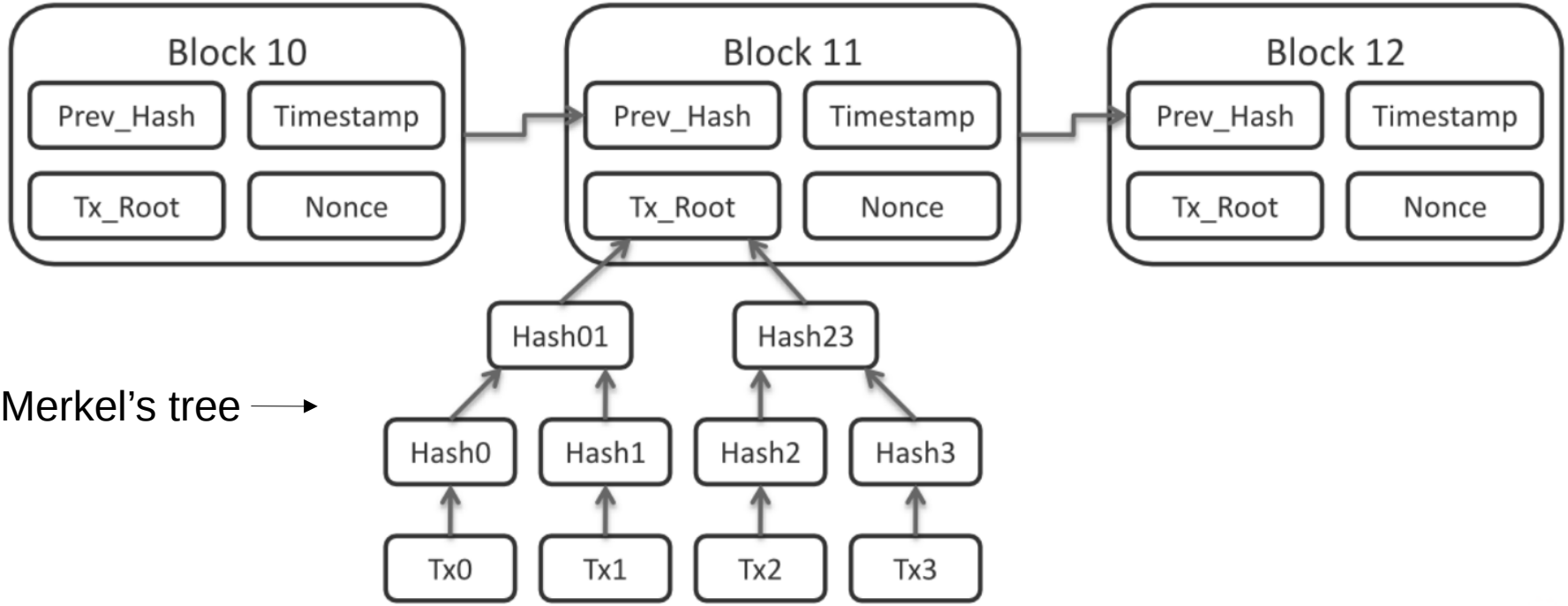


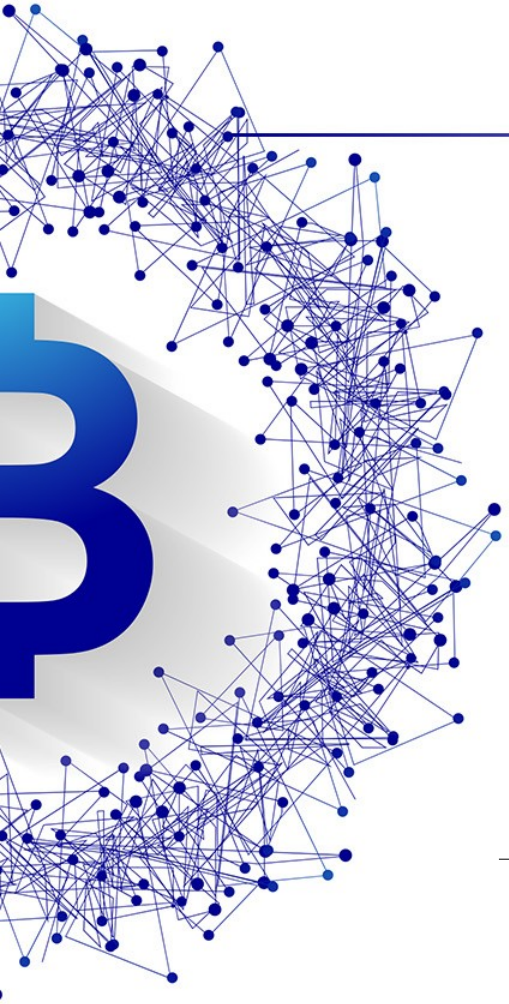
Introduction

- Bitcon (BTC) mysteriuos creator
- Digital money
- Censorship resistance
- Decentralization
- Secure store of value
- There could be max 21M Bitcoins
- Every Bitcoin could be divided to
- 100 million satoshis
- 0.00000001 BTC = 1 satoshi



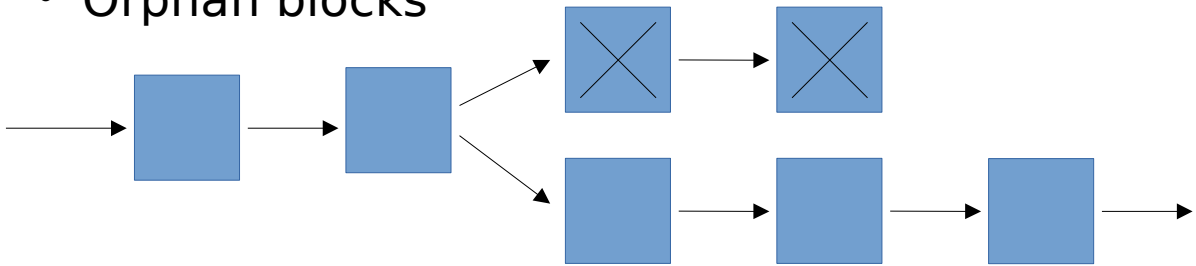
Blockchain


Block 10
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 11
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 12
Prev_Hash
Timestamp
Tx_Root
Nonce
Merkel's tree
Hash01
Hash23
Hash0
Hash1
Hash2
Hash3
Tx0
Tx1
Tx2
Tx3

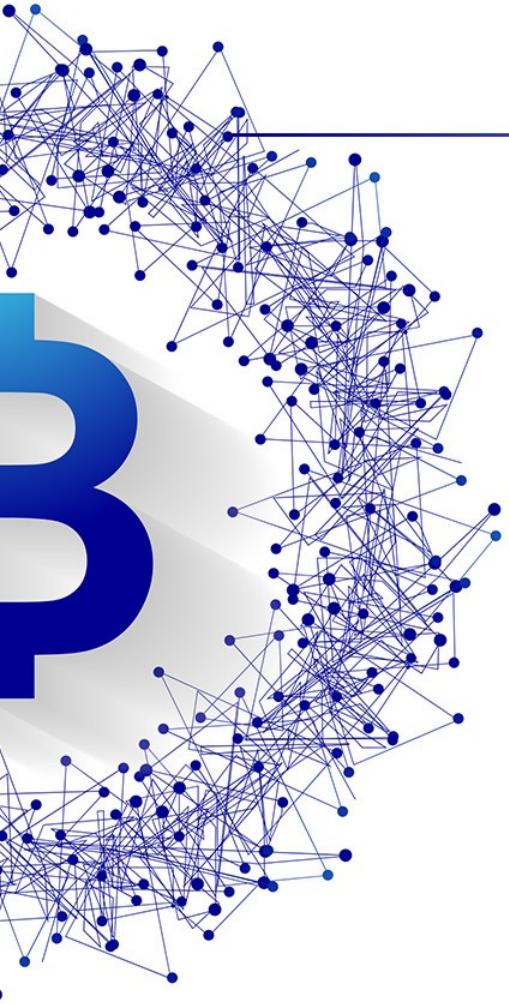


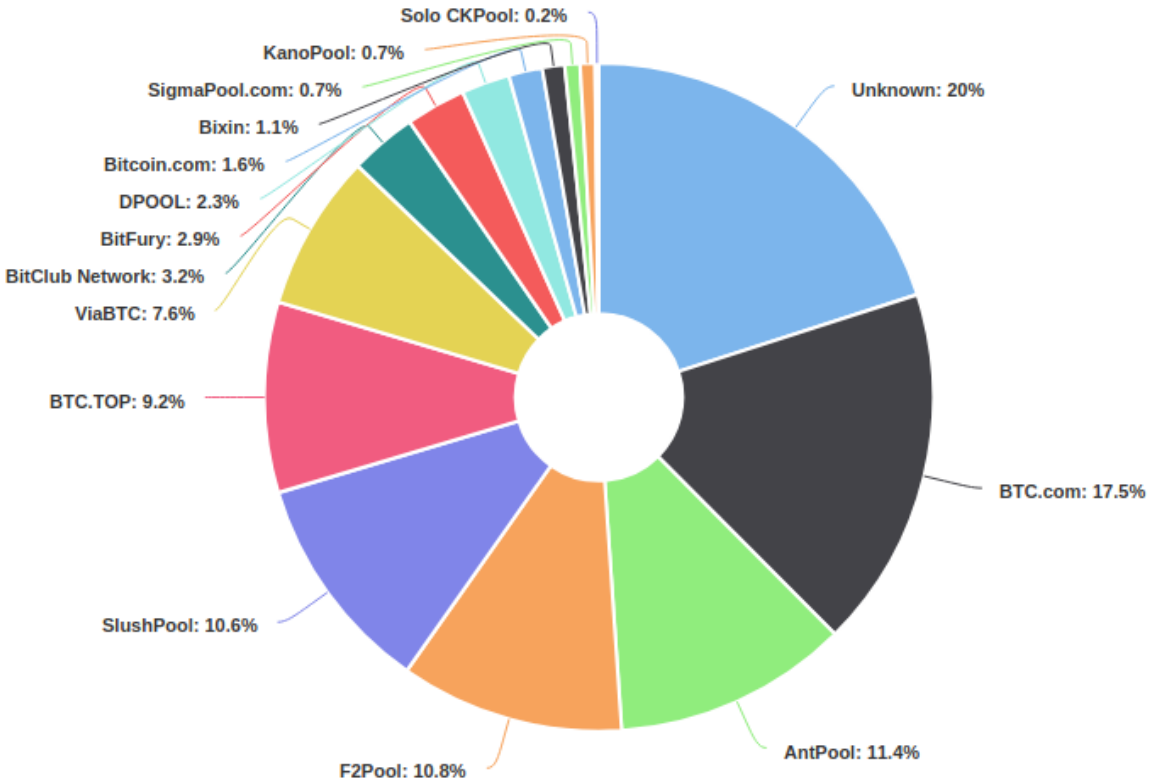
Mining

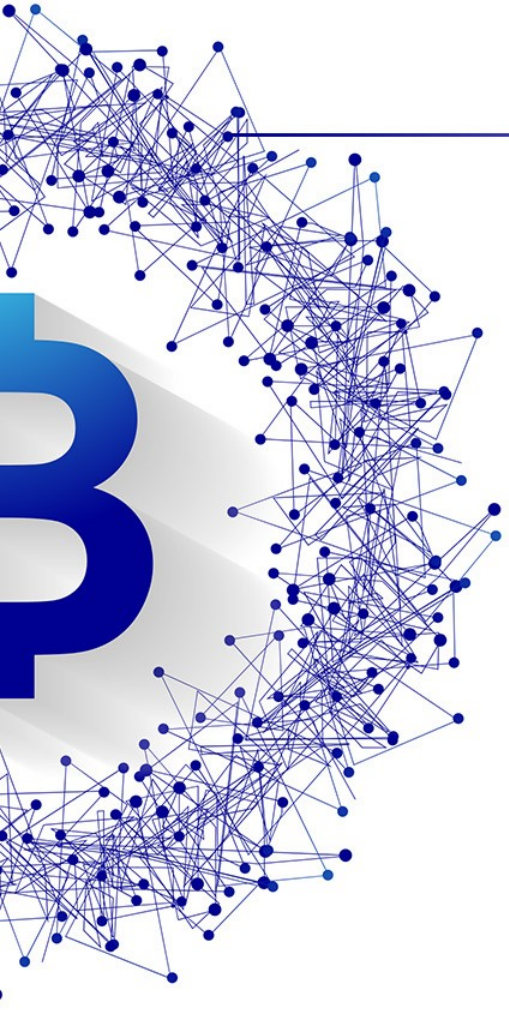
- PoW: sha256(sha256(header)) < Difficulty
- Block generation 10 minutes (winning miner will have reward in coinbase transaction)
- Difficulty adjustment every 2016 block
- Application specific integrated circuit (ASIC)
- Current networka hash rate ~40 eksa h/s
- Miners choose transactions to confirm
- Halving every 210000 blocks (4 years)
- Maximum block size 1Mb
- Orphan blocks



Mining




Solo CKPool: 0.2%
KanoPool: 0.7%
SigmaPool.com: 0.7%
Bixin: 1.1%
Bitcoin.com: 1.6%
DPOOL: 2.3%
BitFury: 2.9%
BitClub Network: 3.2%
ViaBTC: 7.6%
BTC.TOP: 9.2%
SlushPool: 10.6%
F2Pool: 10.8%
AntPool: 11.4%
BTC.com: 17.5%
Unknown: 20%

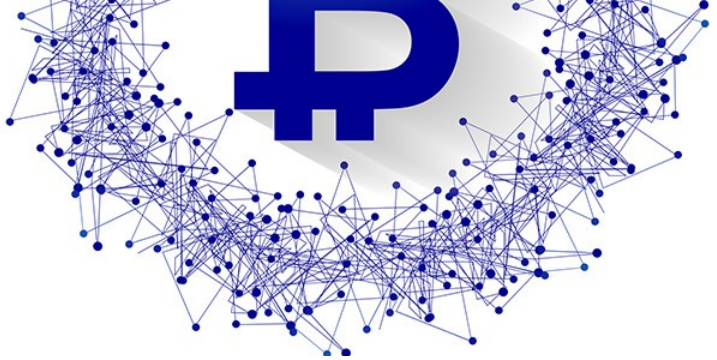


Elliptic curve

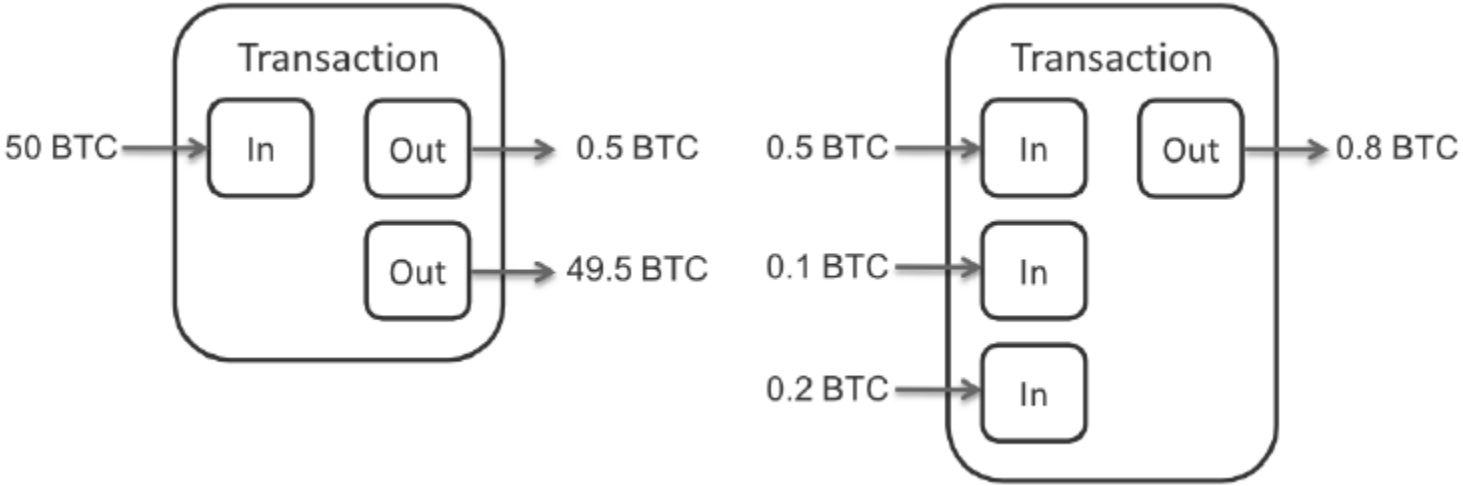
Bitcoin uses elliptic curve **secp256k1** (instead of recommended by NIST secp256r1, beause it was created by NSA and it is not clear why it has choosen parameters):
y^2=x^3+7
p= 2^256 - 2^32 - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1
N = 2^256 - 432420386565659656852420866394968145599
G – generator point

Elliptic curve discrete logarithm problem, given points G and Q, find such integer x, that:
G * x = Q (x is priavate key, Q is public key)

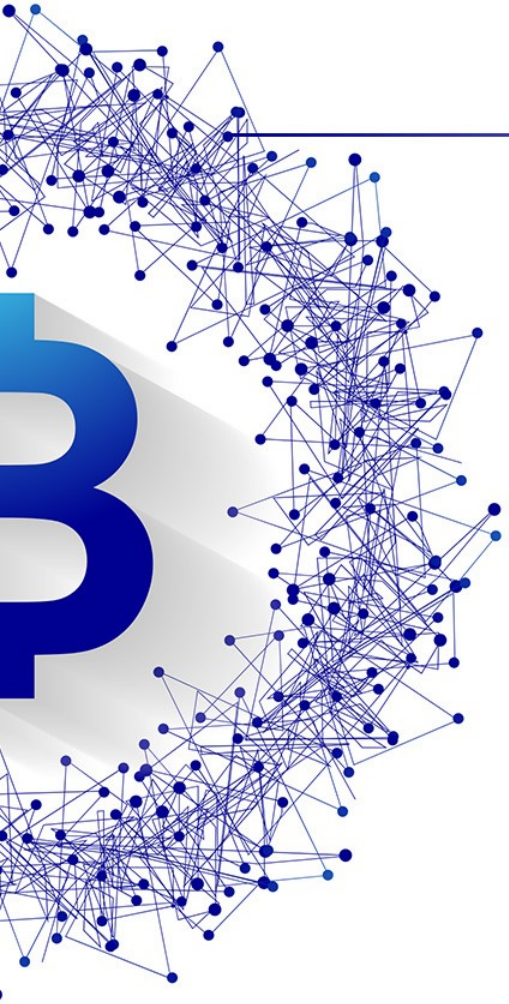
Why it is difficult? For example for every 0 < x < n, there exists y, which fulfill equation G * x * y = G.
Easy with quantum computers (Shor's algorithm).



Transactions



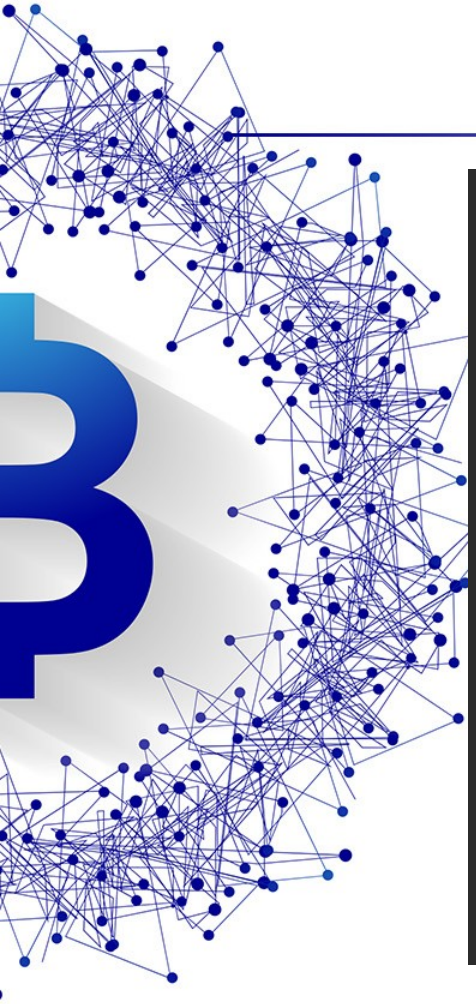
Transaction must have at least one input and at least one output. When inputs exceed payment value, usually new address is created to store change. If we choose to send change to original address, those funds will have exposed public key. More inputs and/or outputs will result in bigger transaction size.



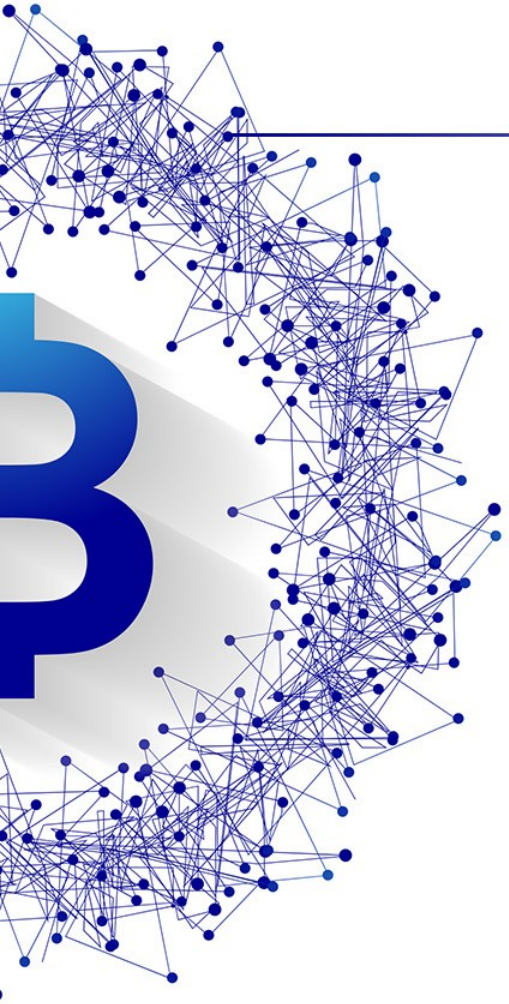
Transactions

- **P2PK (Pay to public key)** – no address, used in early BTC days
- **P2PKH (Pay to public key hash)** – address starts with 1, most popular
- **P2SH (Pay to script hash)** – address starts with 3
- **P2PWKH** and **P2PWSH** – address starts with bc1, coded with bech32, still need more adoption. P2PWSH addresses are longer (they use sha256 instead hash160 to create address digest)

Transactions

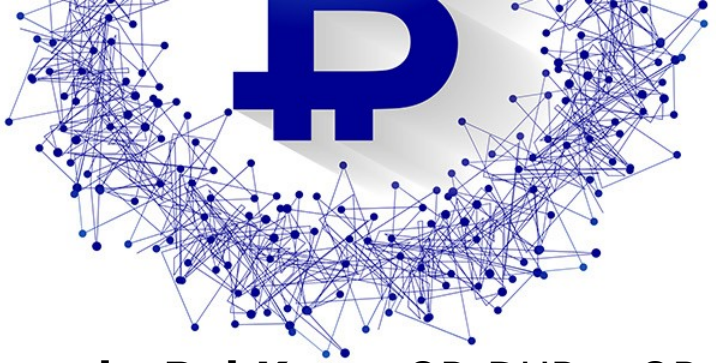


```
raw_transaction_data = (
    self.version +
    self.inputs_counter +
    self.get_encoded_inputs(
        positions=range(len(self.inputs))
    ) +
    self.outputs_counter +
    self.get_encoded_outputs() +
    self.lock_time
)
self.id = reverse_byte_hex(
    double_sha256(raw_transaction_data).hexdigest()
)
self.raw = raw_transaction_data.hex()
```



Transactions

Field	Description	Size
Version no	currently 1	4 bytes
Flag	If present, always 0001, and indicates the presence of witness data	optional 2 byte array
In-counter	positive integer VI = VarInt	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer VI = VarInt	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
Witnesses	A list of witnesses, 1 for each input, omitted if flag above is missing	variable, see Segregated_Witness
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

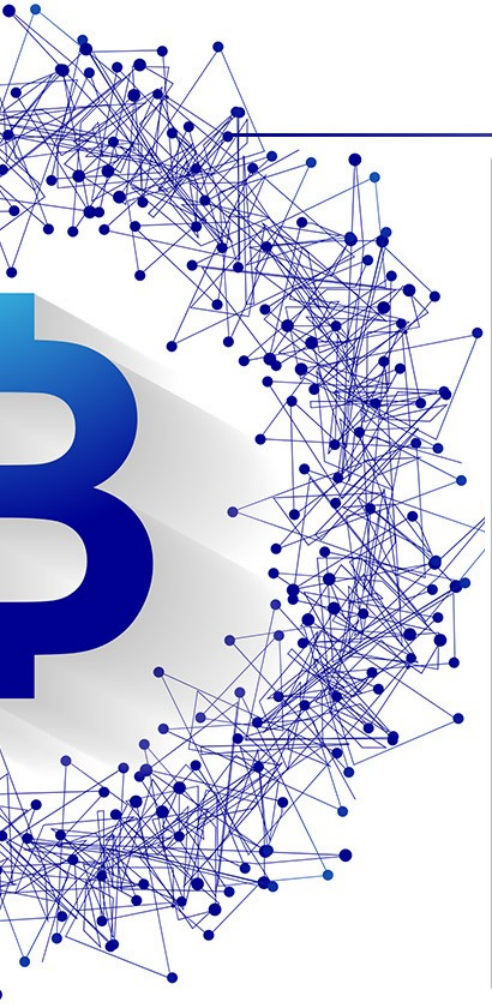


P2PKH transaction

scriptPubKey = OP_DUP + OP_HASH160 + <pubKeyHash> +
OP_EQUALVERIFY + OP_CHECKSIG
scriptSig = <sig> + <pubKey>

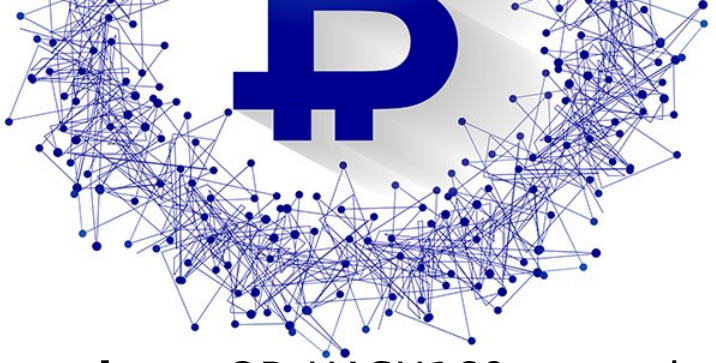
<pubKeyHash> = hash160(encoded_public_key),
where hash160(m) = ripemd160(sha256(m))
addressData = b'\x00' + <pubKeyHash>
checkSum = sha256(sha256(address_data)[:-4]
Address = Base58 (address_data + checkSum)

Exemplary address: 12ib7dApVFvg82TXKycWBNpN8kFyiAN1dr
Corresponding public key (point on elliptic curve):
(96953063599923793356065023910106792740284067034392039319548634253844580007549,
24213599371259323050868340559734230940120001082991520973823206482901563403021)



P2PKH Transaction

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

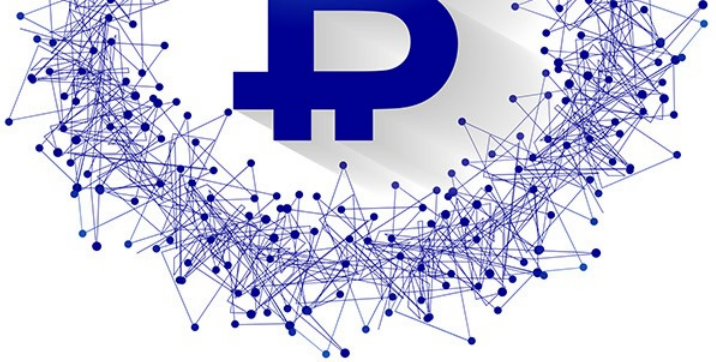


P2SH transaction

script = OP_HASH160 + <scriptHash> + OP_EQUAL
unlocking_script – complementary scirpt, concatenated with `script` must evaluate to true

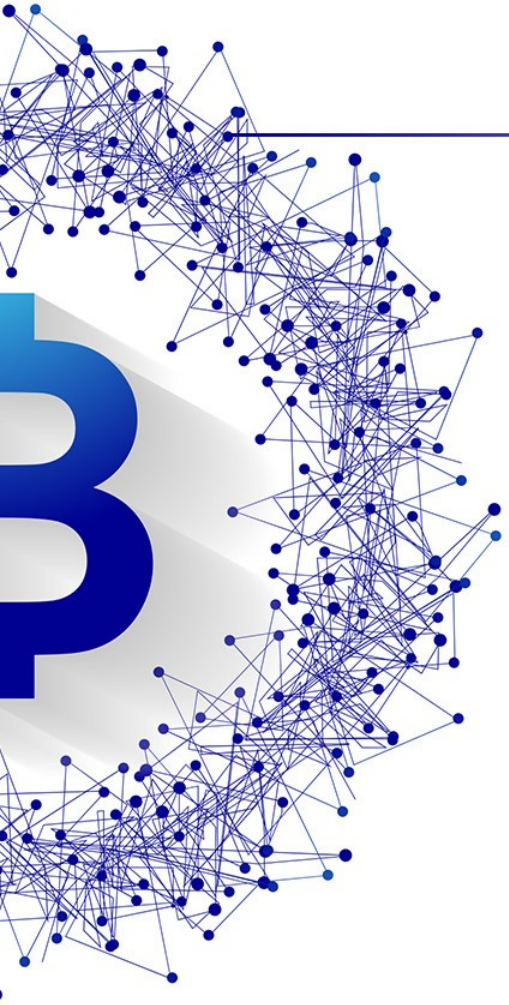
<scriptHash> = Hash160(SCRIPT)
addressData = b'\x05' + <scriptHash>
checkSum = sha256(sha256(address_data)[:-4]
Address = Base58 (address_data + checkSum)

Exemplary address: 37k7toV1Nv4DfmQbmZ8KuZDQCYK9x5Kpz
Corresponding script used to generate it:
OP_2DUP OP_EQUAL OP_NOT OP_VERIFY OP_SHA1 OP_SWAP OP_SHA1 OP_EQUAL
This script was created as a bount to find two different messages giving the same SHA1 hash value. Bounty was already claimed.



Future

- Lightning network
- Shnorr signature (the main reason that Bitcoin did not originally use Schnorr signatures is that Schnorr was not standardized, and was not available in common crypto libraries. An advantage of this method is that, if parties cooperate, we can generate a single signature that validates two or more separate transactions)
- Bulletproofs (zero knowledge proofs)
- Side chains



Thank you!