Exploring Bitcoin

Tomasz Jadachowski
14.03.2019
• Bitcoin genesis
• Blockchain
• Mining
• Transactions
• Security
• Future
Introduction

- Bitcoin (BTC) mysterious 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

Merkel's tree
Mining

- PoW: $\text{sha256(sha256(header))} < \text{Difficulty}$
- Block generation 10 minutes (winning miner will have reward in coinbase transaction)
- Difficulty adjustment every 2016 block
- Application specific integrated circuit (ASIC)
- Current network hash rate $\sim 40$ eksa h/s
- Miners choose transactions to confirm
- Halving every 210000 blocks (4 years)
- Maximum block size 1Mb
- Orphan blocks
Bitcoin uses elliptic curve `secp256k1` (instead of recommended by NIST `secp256r1`, because it was created by NSA and it is not clear why it has chosen 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:

\[ G \times x = Q \]  (x is private key, Q is public key)

Why it is difficult? For example for every \( 0 < x < n \), there exists \( y \), which fulfill equation \( G \times x \times y = G \). 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
- **P2PKWH 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

```python
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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version no</td>
<td>currently 1</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Flag</td>
<td>If present, always 0001, and indicates the presence of witness data</td>
<td>optional 2 byte array</td>
</tr>
<tr>
<td>In-counter</td>
<td>positive integer VI = VarInt</td>
<td>1 - 9 bytes</td>
</tr>
<tr>
<td>list of inputs</td>
<td>the first input of the first transaction is also called &quot;coinbase&quot; (its content was ignored in earlier versions)</td>
<td>&lt;in-counter&gt;-many inputs</td>
</tr>
<tr>
<td>Out-counter</td>
<td>positive integer VI = VarInt</td>
<td>1 - 9 bytes</td>
</tr>
<tr>
<td>list of outputs</td>
<td>the outputs of the first transaction spend the mined bitcoins for the block</td>
<td>&lt;out-counter&gt;-many outputs</td>
</tr>
<tr>
<td>Witnesses</td>
<td>A list of witnesses, 1 for each input, omitted if flag above is missing</td>
<td>variable, see Segregated_Witness</td>
</tr>
<tr>
<td>lock_time</td>
<td>if non-zero and sequence numbers are &lt; 0xFFFFFFFFF: block height or timestamp when transaction is final</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>
**P2PKH transaction**

\[
\text{scriptPubKey} = \text{OP\_DUP} + \text{OP\_HASH160} + \text{<pubKeyHash>} + \\
\text{OP\_EQUALVERIFY} + \text{OP\_CHECKSIG}
\]

\[
\text{scriptSig} = \text{<sig>} + \text{<pubKey>}
\]

\[
\text{<pubKeyHash>} = \text{hash160(\text{encoded\_public\_key})},
\]
where \( \text{hash160(m)} = \text{ripemd160(sha256(m))} \)

\[
\text{addressData} = \text{b'}\text{x00}'+\text{<pubKeyHash>}
\]

\[
\text{checkSum} = \text{sha256(sha256(address\_data)[:-4])}
\]

\[
\text{Address} = \text{Base58 (address\_data + checkSum)}
\]

Exemplary address: 12ib7dApVFvg82TXKycWBNpN8kFyiAN1dr

Corresponding public key (point on elliptic curve):
\((969530635999237933560650239101067927402840670343920393195486342538445800007549, \\
24213599371259323050868340559734230940120001082991520973823206482901563403021)\)
## P2PKH Transaction

<table>
<thead>
<tr>
<th>Stack</th>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty.</td>
<td><code>&lt;sig&gt; &lt;pubKey&gt; OP_DUP OP_HASH160 &lt;pubKeyHash&gt; OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>scriptSig and scriptPubKey are combined.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code></td>
<td><code>OP_DUP OP_HASH160 &lt;pubKeyHash&gt; OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>Constants are added to the stack.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code> &lt;pubKey&gt;</td>
<td><code>OP_HASH160 &lt;pubKeyHash&gt; OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>Top stack item is duplicated.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code> &lt;pubHashA&gt; &lt;pubKeyHash&gt;</td>
<td><code>OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>Top stack item is hashed.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code> &lt;pubHashA&gt; &lt;pubKeyHash&gt;</td>
<td><code>OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>Constant added.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code> &lt;pubKeyHash&gt; &lt;pubKeyHash&gt;</td>
<td><code>OP_EQUALVERIFY OP_CHECKSIG</code></td>
<td>Equality is checked between the top two stack items.</td>
</tr>
<tr>
<td><code>&lt;sig&gt; &lt;pubKey&gt;</code></td>
<td><code>OP_CHECKSIG</code></td>
<td>Signature is checked for top two stack items.</td>
</tr>
<tr>
<td>true</td>
<td>Empty.</td>
<td></td>
</tr>
</tbody>
</table>
P2SH transaction

**script** = OP_HASH160 + <scriptHash> + OP_EQUAL

**unlocking_script** - complementary script, 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!