# Basic Cryptography and Bitcoin

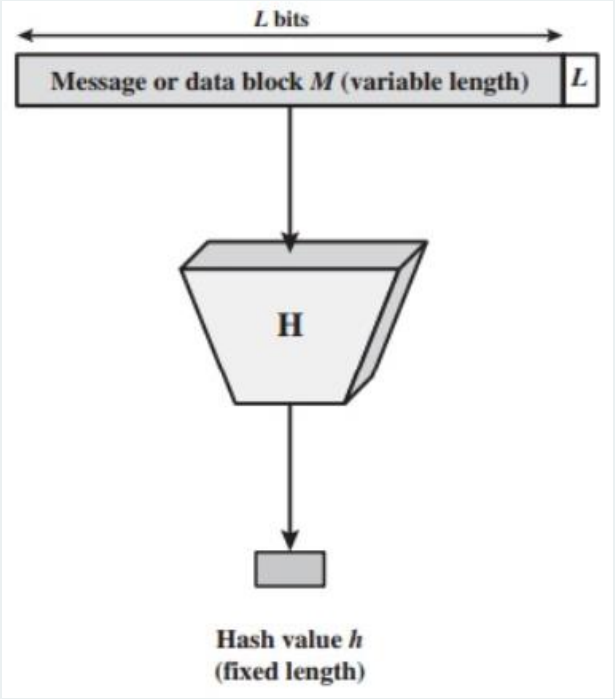
# Agenda

- Hash functions, symmetric and asymmetric encryption, private and public keys
- RSA and ECC
- Bitcoin basics
- Advanced scripting
- Exercise

https://github.com/sofitto/bitcoin-workshop

# Hash functions

- **Computationally efficient:** the computational time and power needed to transform input into output is very limited.
- **Hide information about the input:** in contrast to previous property, it is very hard to obtain information on the input given the output.
- **The output should look random:** minor changes in the input will generate  a completely distinct and well-distributed output. It is impossible to see a direct link between the input and the output.



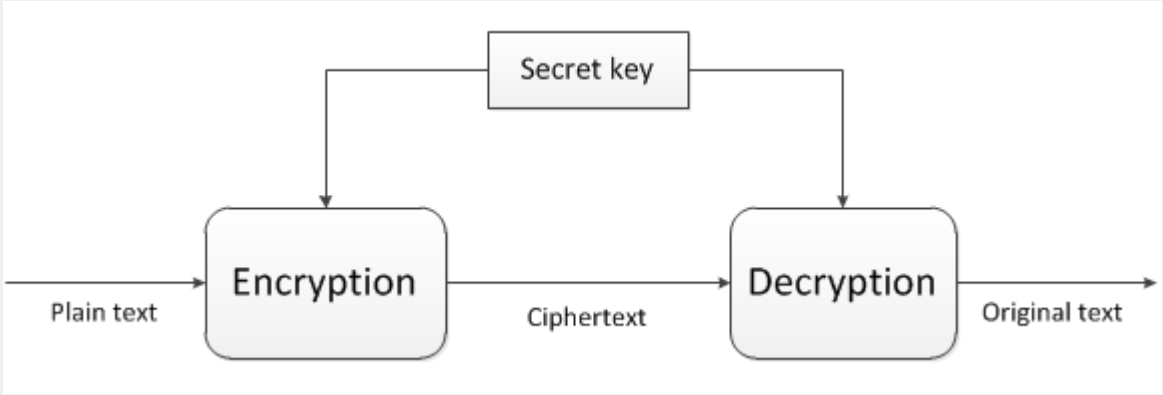
# Hash functions

| Algorithm | Message size (bits) | Digest size (bits) | Collision found |
|---|---|---|---|
| MD4 | $< 2^{64}$ | 128 | yes |
| MD5 | $< 2^{64}$ | 128 | yes |
| SHA-1 | $< 2^{64}$ | 160 | theoretical attack |
| SHA-224 & SHA-256 | $< 2^{64}$ | 224 & 256 | no |
| SHA-384 & SHA-512 | $< 2^{128}$ | 384 & 512 | no |
| SHA-512/224 & SHA-512/256 | $< 2^{128}$ | 224 & 256 | no |
| SHA3-224 & SHA3-256 | arbitrary | 224 & 256 | no |
| SHA3-384 & SHA3-512 | arbitrary | 384 & 512 | no |
| SHAKED-128 & SHAKED-256 | arbitrary | arbitrary | no |

Bitcoin – SHA-256, RIPEMD-160

Ethereum – Keccak-256 (not NIST standard SHA-3)

# Symmetric encryption



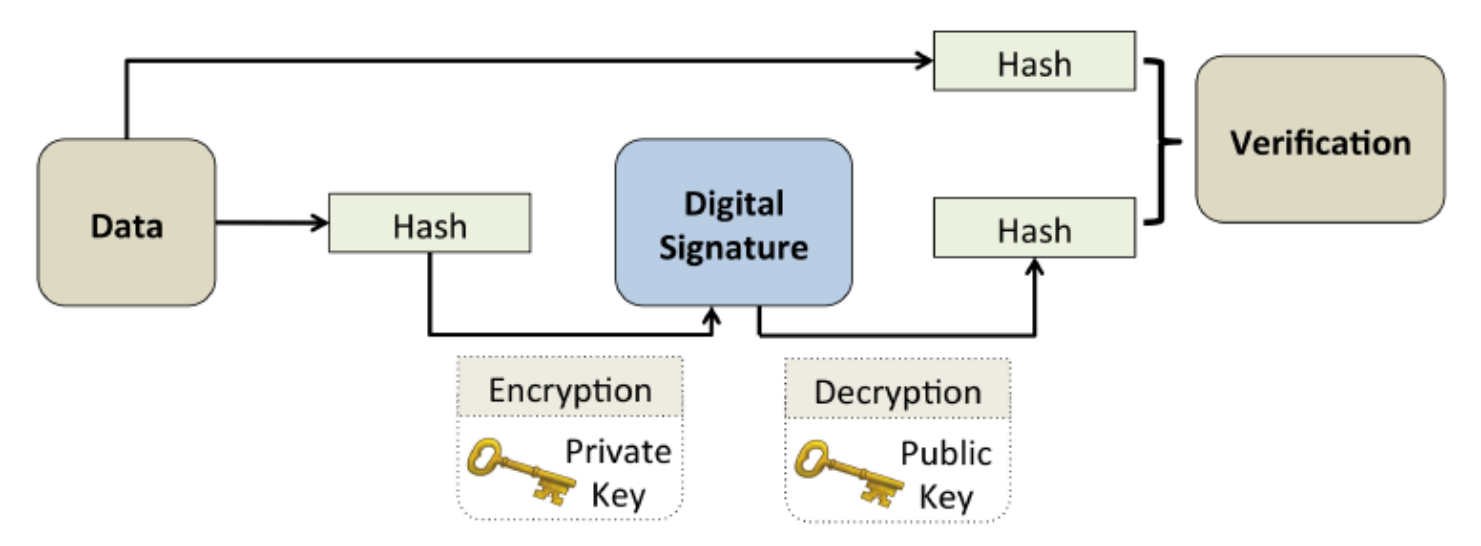

3DES, AES and many more…

Main weakness: the key should be shared between parties

# Asymmetric encryption

Encryption and decryption are performed with different keys: a **private** and **public** key.

The **private** key or signing key (SK) is known only to the owner of that key. The **public** key or verification key (VK) is visible to all users in the network.



# Asymmetric encryption

| Security(Bits) | Minimum size (bits) of Public Keys | | | Key Size Ratio | Protection From Attack |
|---|---|---|---|---|---|
| | DSA | RSA | ECC | ECC to RSA/DSA | |
| 80 | 1024 | 1024 | 160 | 1:6 | Until 2010 |
| 112 | 2048 | 2048 | 224 | 1:9 | Until 2030 |
| 128 | 3072 | 3072 | 256 | 1:12 | Beyond 2031 |
| 192 | 7680 | 7680 | 384 | 1:20 | |
| 256 | 15360 | 15360 | 512 + | 1:30 | |

# Recent problems with RSA

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS

COMPLETELY BROKEN —

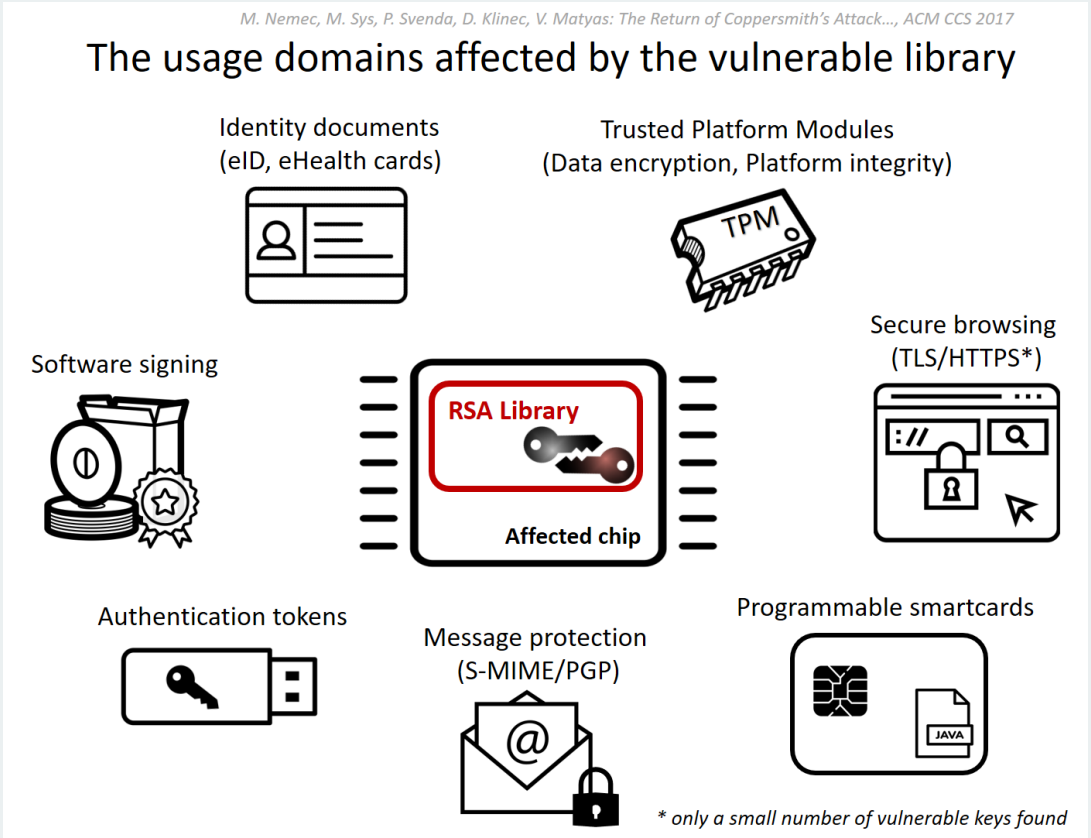
## Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

DAN GOODIN - 10/16/2017, 5:00 PM


Steve Jurvetson

Enlarge / 750,000 Estonian cards that look like this use a 2048-bit RSA key that can be factored in a matter of days.




https://crocs.fi.muni.cz/public/papers/rsa_ccs17

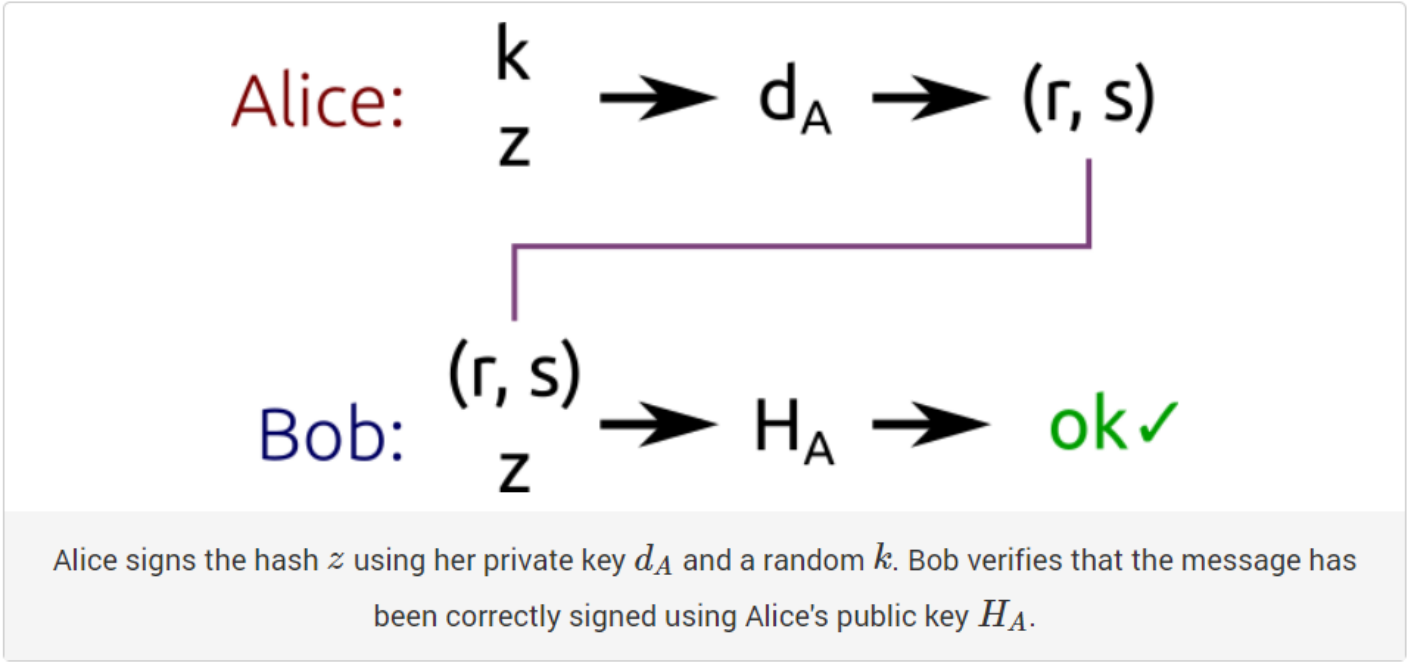
https://arstechnica.com/information-technology/2017/10/crypto-failure-cripples-millions-of-high-security-keys-750k-estonian-ids/

sofitto

# ECDSA “Under the Hood“

1. Take a **random integer** $k$ chosen from $\{1, \ldots, n-1\}$ (where $n$ is still the subgroup order).
2. Calculate the point $P = kG$ (where $G$ is the base point of the subgroup).
3. Calculate the number $r = x_P \bmod n$ (where $x_P$ is the $x$ coordinate of $P$).
4. If $r = 0$, then choose another $k$ and try again.
5. Calculate $s = k^{-1}(z + rd_A) \bmod n$ (where $d_A$ is Alice's private key and $k^{-1}$ is the multiplicative inverse of $k$ modulo $n$).
6. If $s = 0$, then choose another $k$ and try again.

The pair $(r, s)$ **is the signature**.



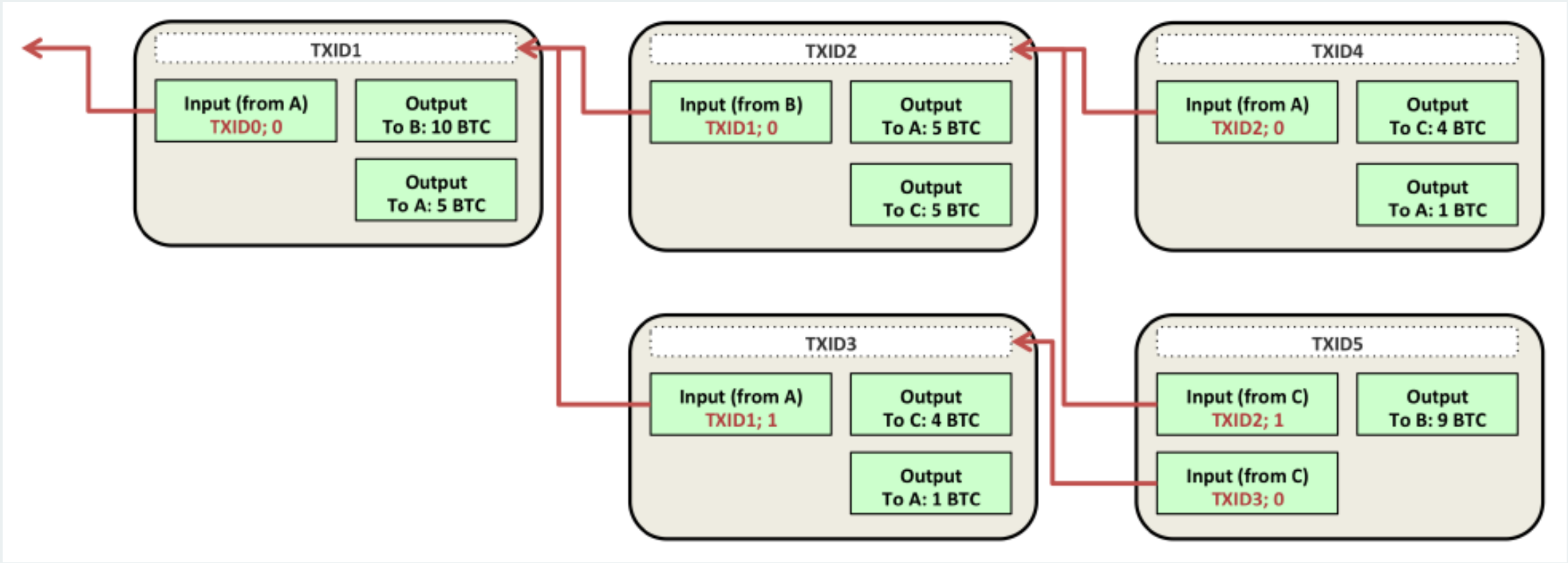

Alice signs the hash $z$ using her private key $d_A$ and a random $k$. Bob verifies that the message has been correctly signed using Alice's public key $H_A$.

Sony PlayStation Hack:

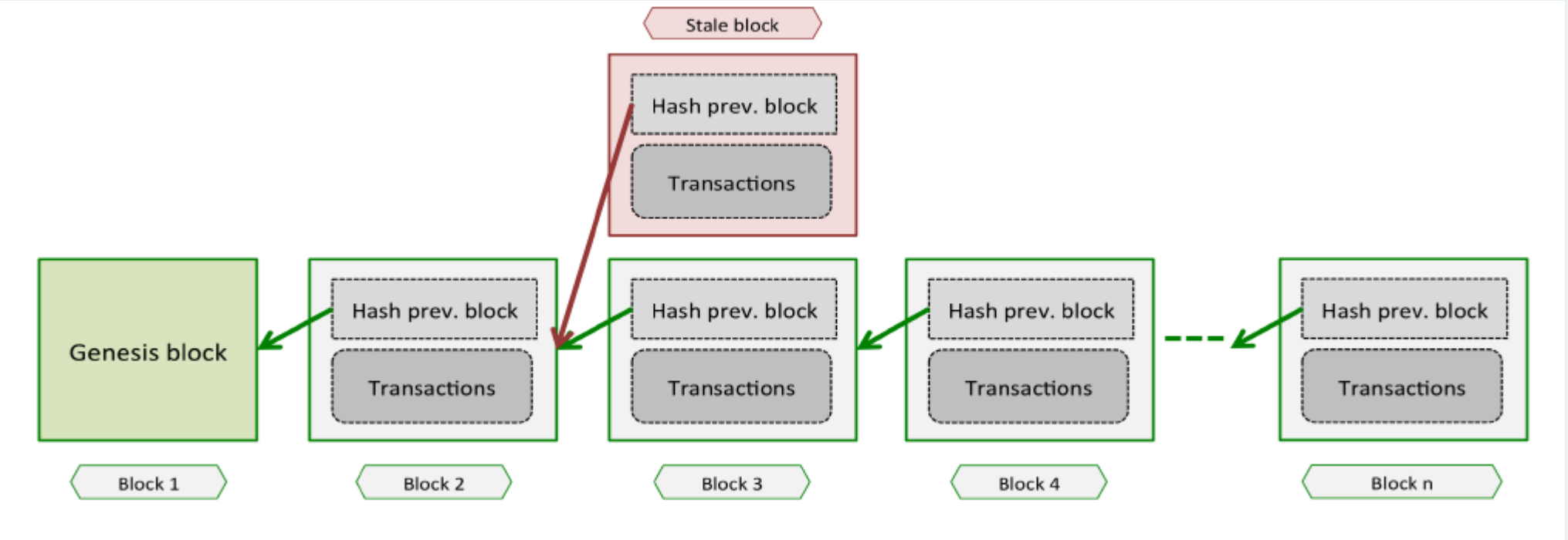
https://youtu.be/4loZGYqaZ7I?t=35m45s

http://andrea.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/

# Bitcoin Blockchain (transactions)



# Bitcoin Blockchain

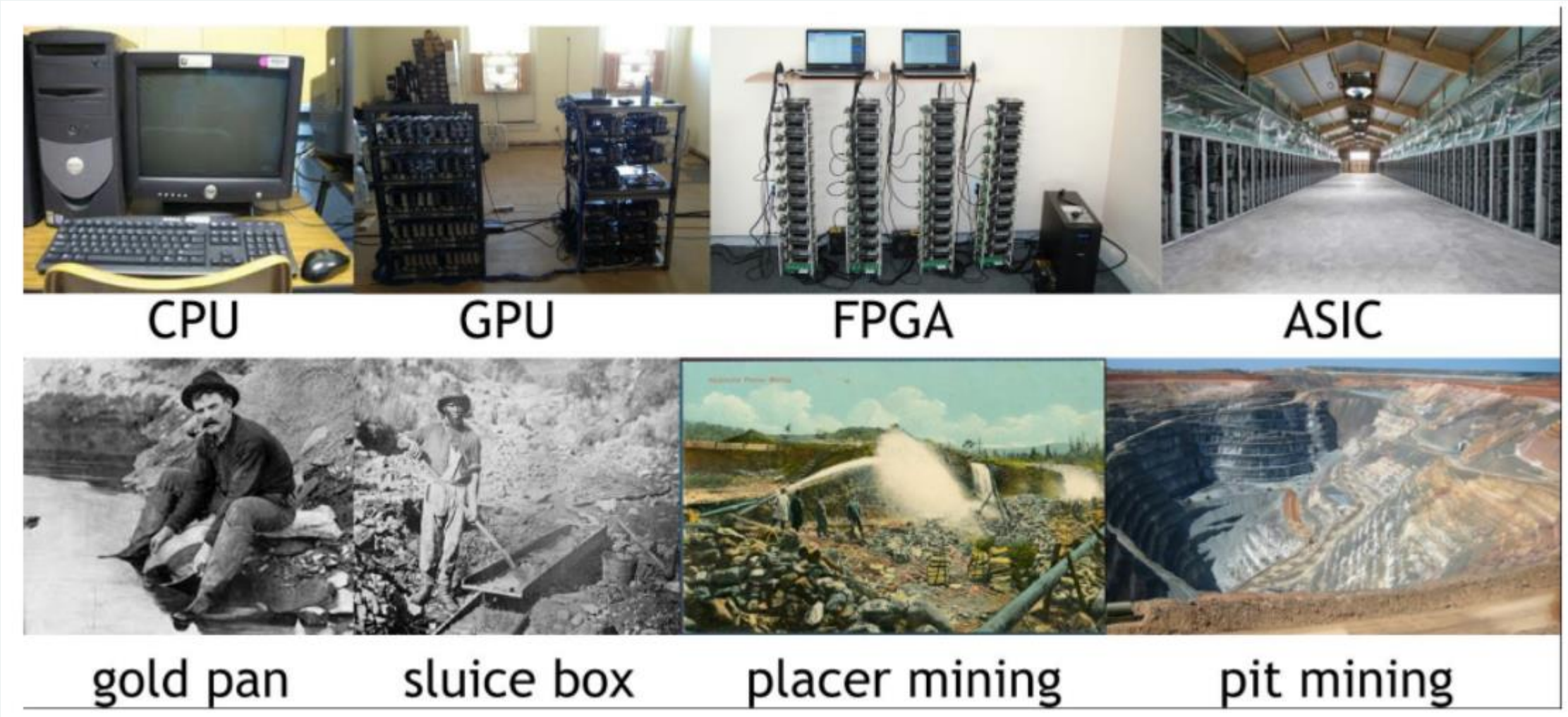



- Everybody has a copy of the database
- The database is (almost) append-only (i.e. data written is immutable)
- The consistency of the database is easily verified as the system follows specific rules

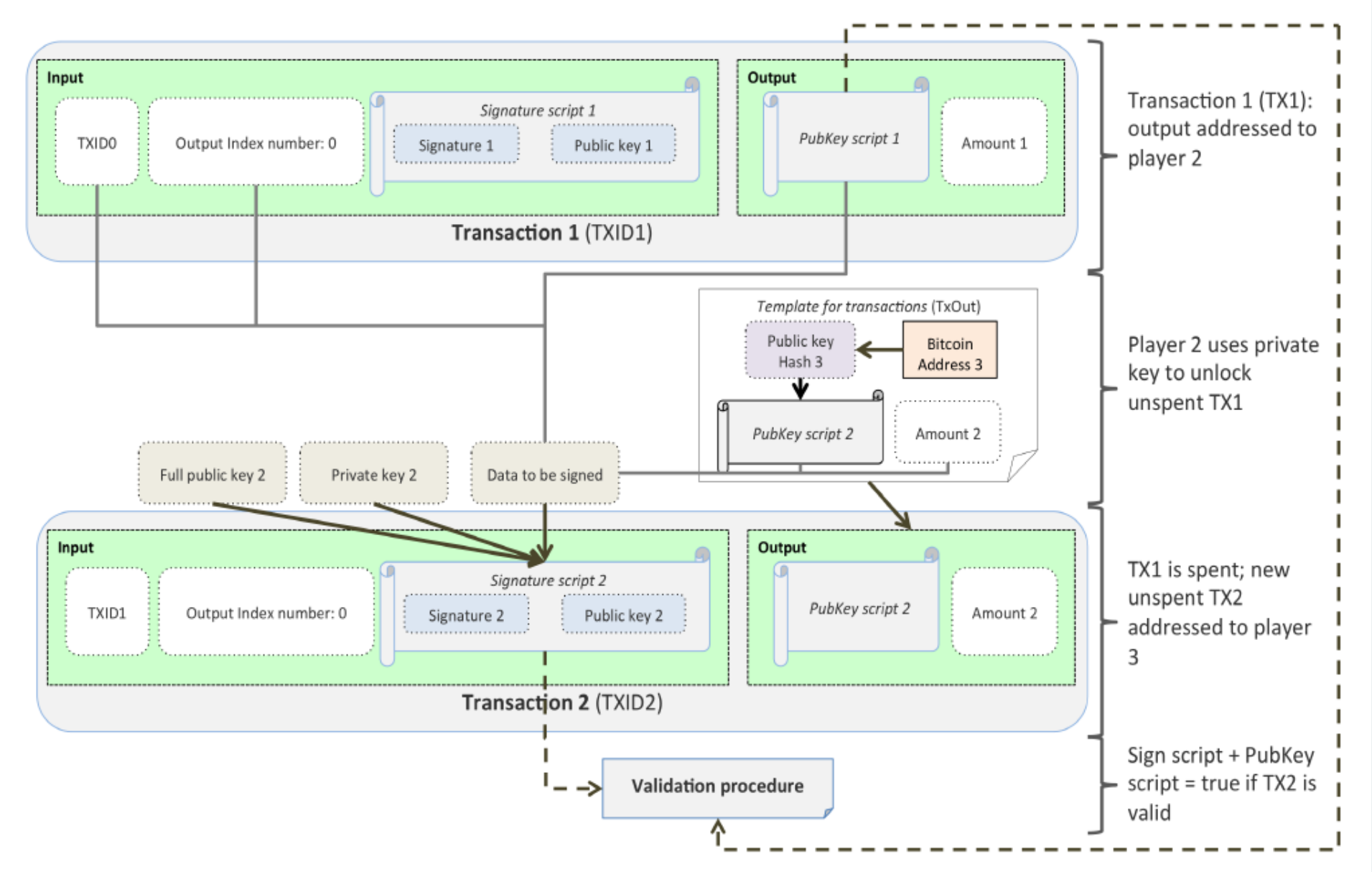
# Bitcoin Blockchain (block header)

| Field | Updated when | Size (bytes) |
|---|---|---|
| Version of software used | When software is upgraded | 4 |
| Hash of previous block | When new block comes in | 32 |
| Hash of Merkle root | When new transaction is accepted | 32 |
| Timestamp | Every few seconds | 4 |
| Current target in compact format (Bits) | When difficulty is adjusted | 4 |
| Nonce | When a block header hash is tried | 4 |

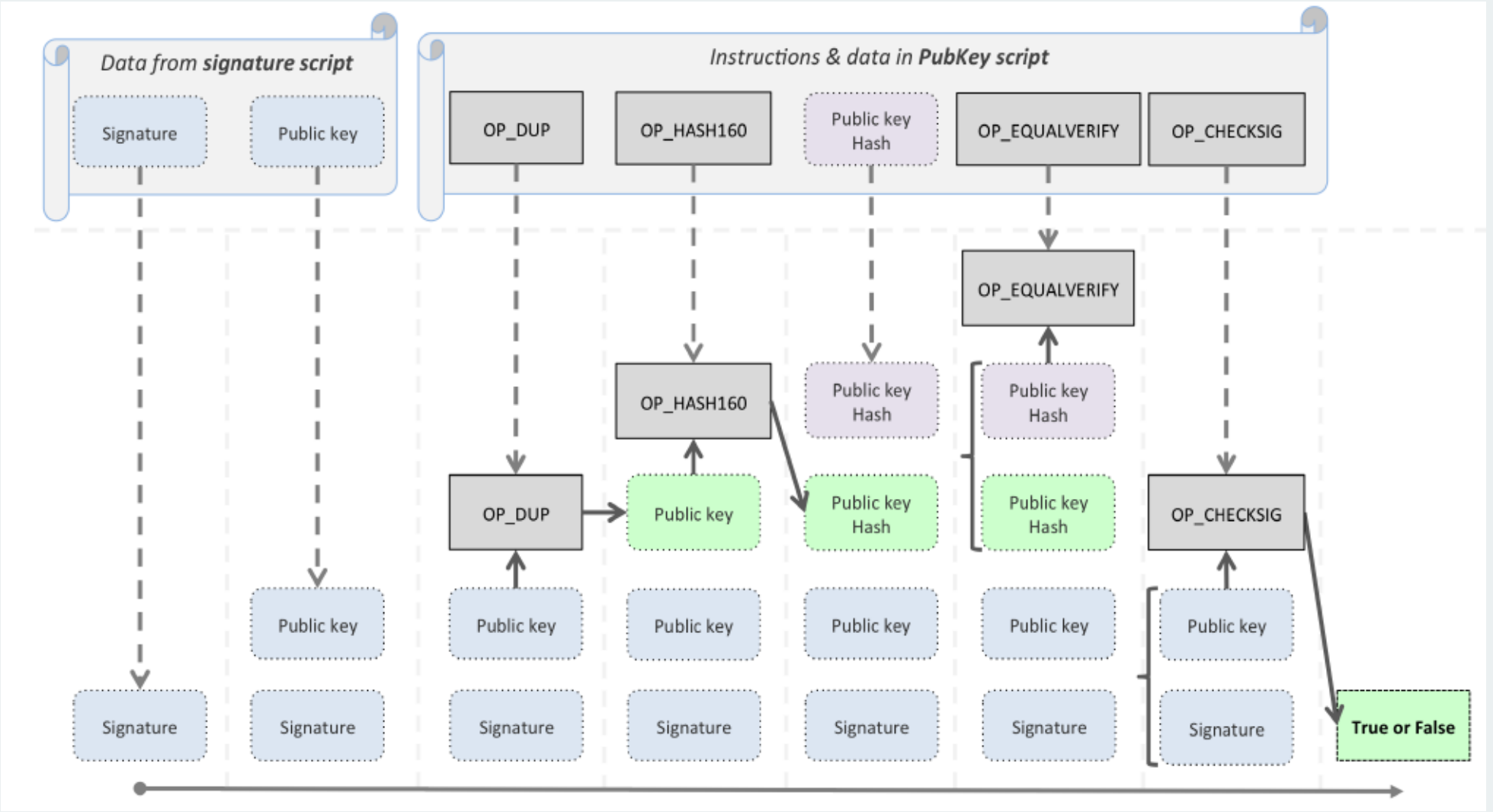
# Bitcoin mining(searching for nonce)



# Bitcoin scripting (P2SH)

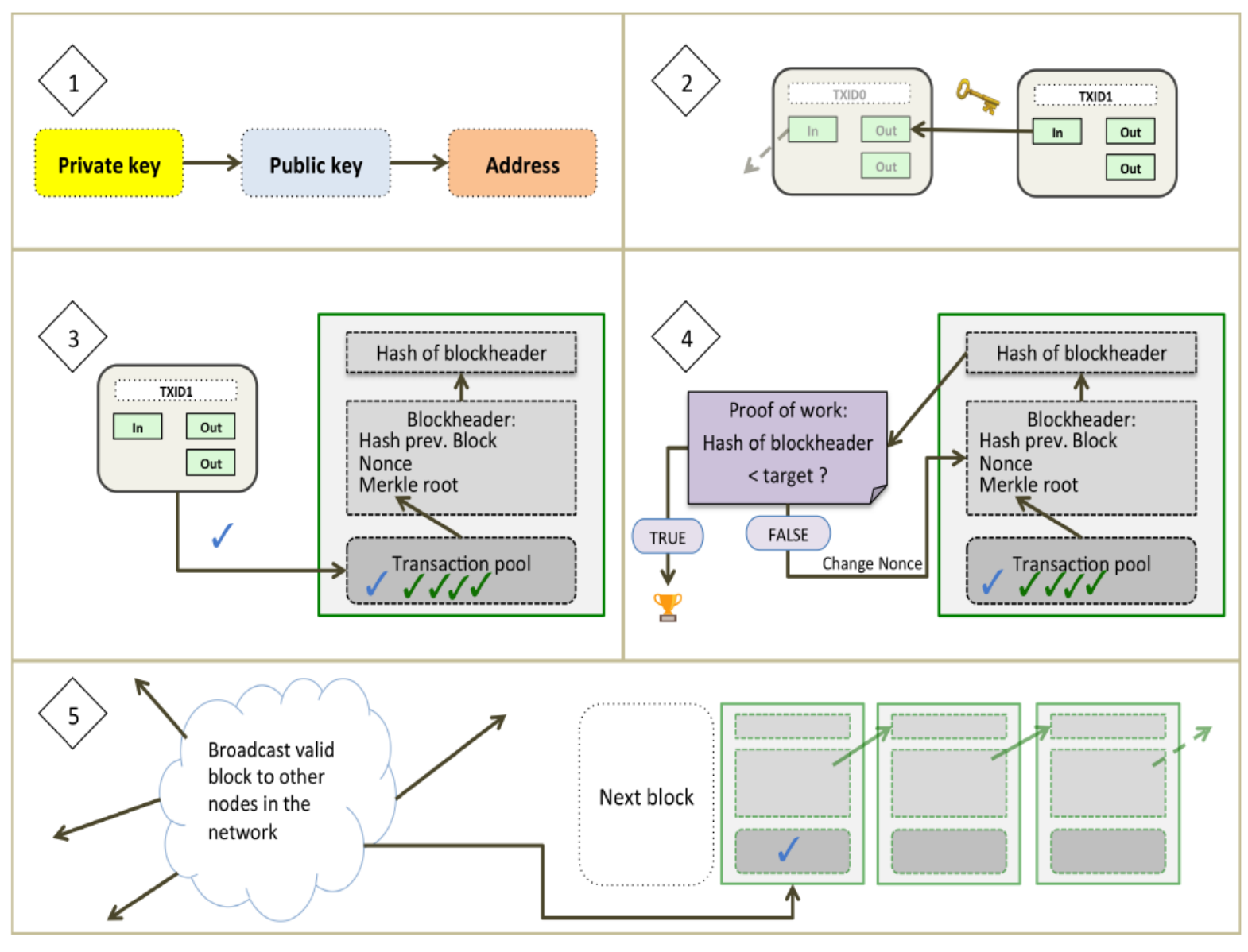



sofitto




| Opcode | Function |
|---|---|
| OP_DUP | duplicates top stack item |
| OP_HASH160 | computes public key hash from public key |
| OP_EQUALVERIFY | compares top two items; returns true or false |
| OP_CHECKSIG | verifies signature; returns true or false |

# Different steps involved to transfer bitcoins




sofitto

# Links

Libraries:

https://github.com/bitpay/bitcore-lib - Bitcore (JavaScript)

https://github.com/bcoin-org/bcoin - Bcoin (JavaScript)

https://github.com/bitcoinj/bitcoinj - BitcoinJ (Java)

Wallets:

https://github.com/mycelium-com/wallet-android

https://github.com/greenaddress/GreenBits

# Practical Exercise