Blockchain, cryptocurrencies and Bitcoin
Printed version

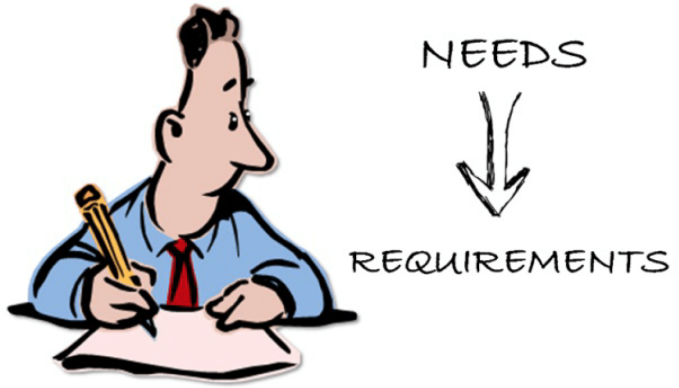
Technology, architecture, implementation

Vasyl Zvarydchuk

Blockchain, cryptocurrencies and Bitcoin

- Blockchain basics
  - Security and integrity by design
  - Blockchain as distributed database
- Blockchain\Cryptocurrency network
  - Users\Software, P2P, nodes, protocols
  - Bitcoin P2P full nodes
- Cryptocurrency wallets
  - Types, public/private keys, ECDSA, addresses
- Blocks and transactions
  - Merkley tree, blocks structure, hashing and validation
  - Money transfer and genesis
- Mining
  - Proof-of-work, hashcash
  - Software and hardware, network difficulty, hash-rate
- Security issues and attacks
  - Anonimity
  - 51% and 34% attacks
  - DoS, Eclipse, ID, MINT and other attacks

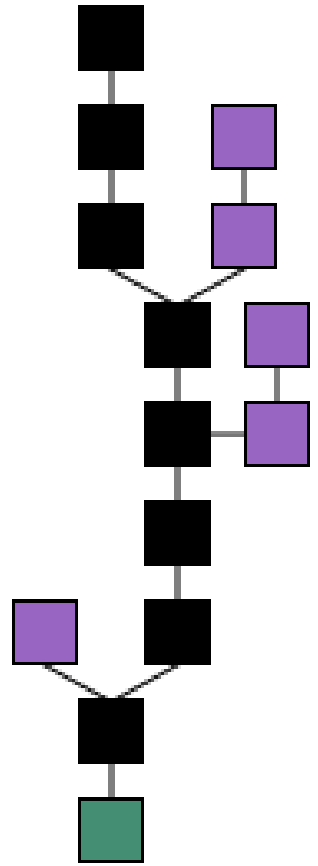
Requirements



- Bare minimum
  - hash, transaction, encryption, decryption, public\private key, TCP\IP basics, data structures (tree, hash table/map), encoding, broadcast
- Good
  - cryptographic hash, SHA 1/2, asymmetric encryption (RSA), digital signature, integrity validation, peer-to-peer (P2P), basic network security, base64 encoding
- Perfect
  - cryptographic protocols, ECDSA, Merkle/hash tree, P2P architecture and vulnerabilities, cryptographic attacks, gossip protocols, distributed hash table

Blockchain basics

- Blockchain
  - Data structure built from blocks with hash pointer to the previous block
  - Data is continuously added
  - Every block has transactions (one or more)
  - Transaction is some data record (money transfer, vote, token, …)
- Security and integrity by design
  - Block change requires changing all next blocks
- Fork (branch)
  - when a few blocks have reference to the same block
  - forks share the same block history up to some point (block)



Blockchain as distributed database

- Every full node has full blockchain local copy
  - Bitcoin has 10-100K copies, the most redundant db (2-20 PB)
- Blockchain database contains:
  - blocks (header + transactions)
    - size limits: Bitcoin– 1MB, Ethereum – no*, BCH – 8MB
  - Indexes, bloom filters
  - Some other data (addresses)
- Blockchain implementations use key/value data storage to store blocks
  - Bitcoin uses Bitcoin – LevelDB
  - Bitcoin used Berkley DB (up to Mar 2013)

Bitcoin blockchain DB stat

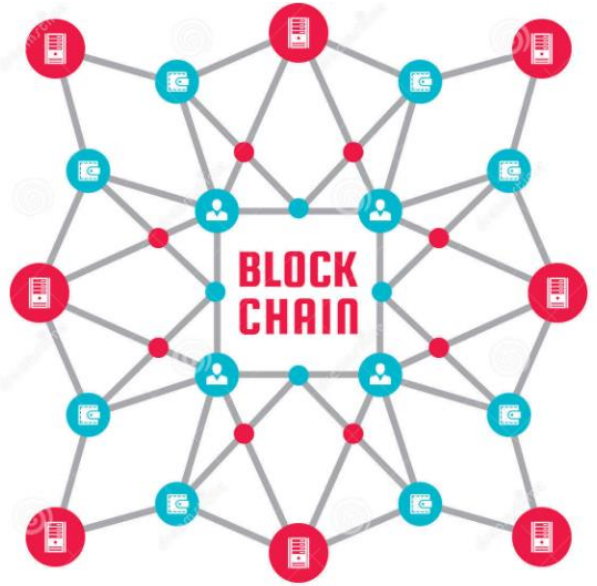
- BD size about 180GB
- 0.5M blocks
- 6 blocks per hour (every 10 min)
- 300M transactions
- 200-300K transactions per day (tpd) or 2-3 tps (per second), peak – 4 tps
  - VISA and Mastercard – 1.8-2K tps
  - PayPal – 100-150 tps
- Transaction size – 0.3-1KB
- Transactions per block – 1.5-2.5K
- Block size limits 1 MB

Blockchain software and users

- Wallets
  - send/receive money, balance, validation
- Miners
  - blockchain builders
  - Blockchain will die without miners
- Services:
  - Exchanges, banks (bitfinex, gdax, coinbase, ..)
  - Info and stats (blockchain.info)
  - ATMs, analytics, calculators, ..

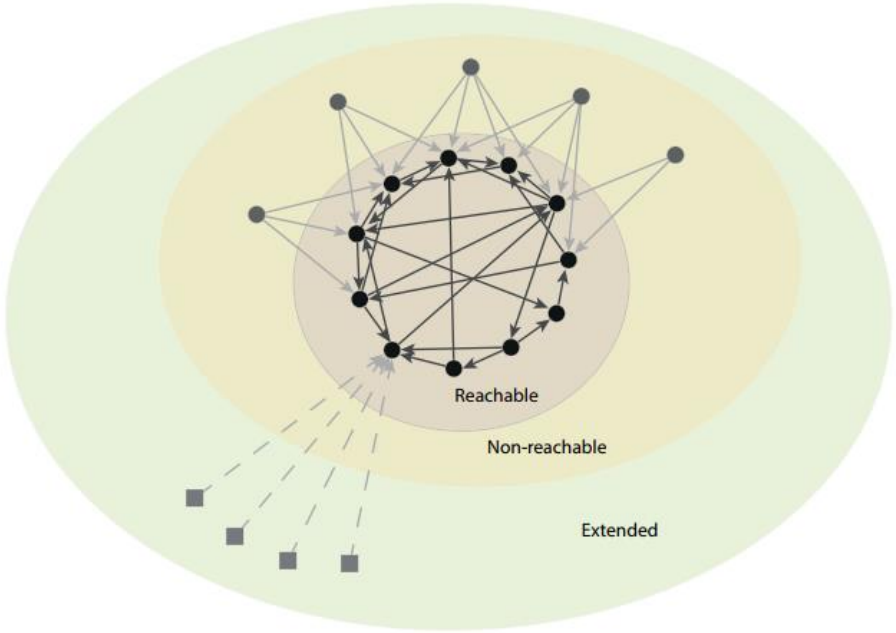
Blockchain\Cryptocurrency network

- Blockchain is basically Peer-to-Peer (P2P) unstructured network
  - There's no network topology
  - Nodes can easy come and leave network
- Blockchain protocols implement network communication rules
  - Bitcoin, Ethereum, … protocols (P2P/TCP)
  - Miner protocols:
    - Stratum (TCP)
    - Getblocktemplate (JSON-RPC)
    - Getwork (Http)

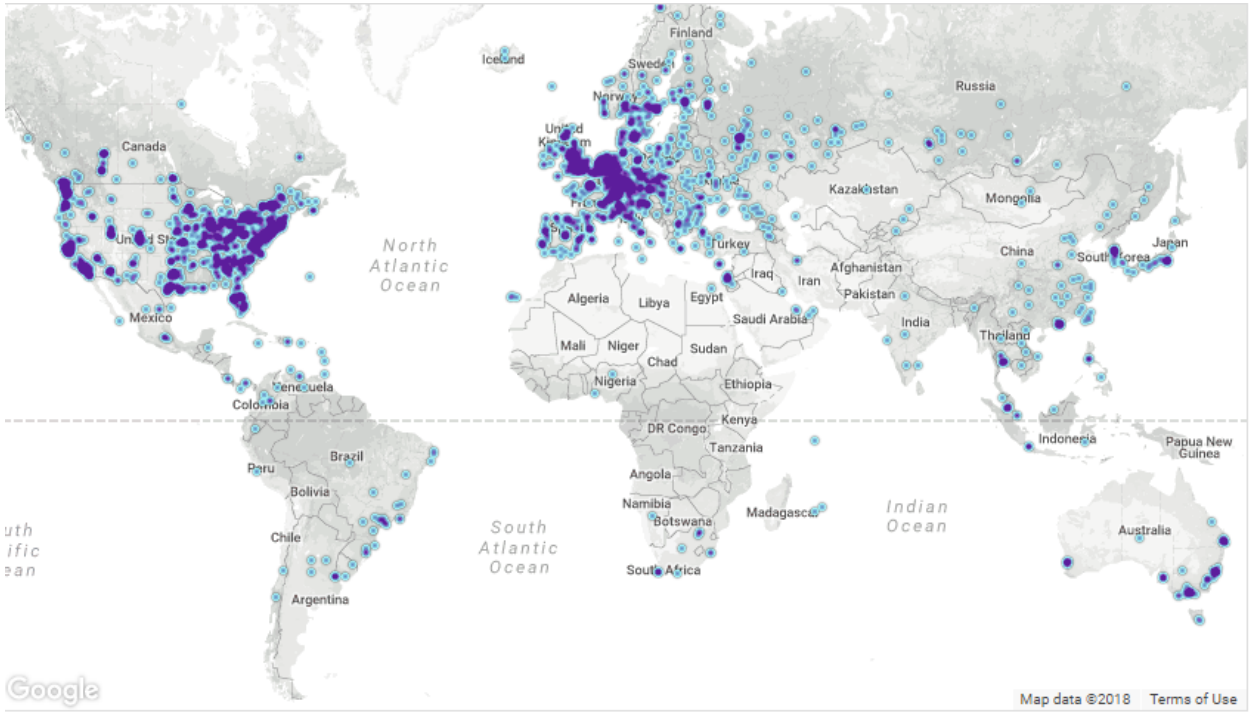


Bitcoin network

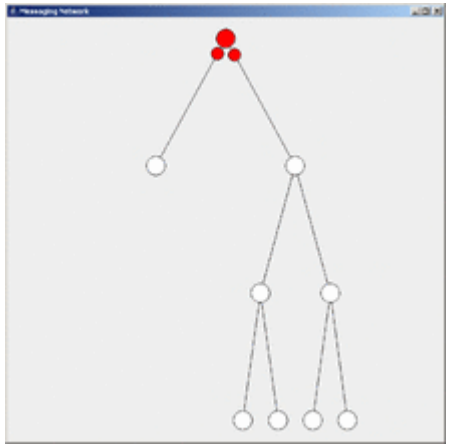
- Full network



Node map (http://bitnotes.earn.com)



Bitcoin full node



- Features:
  - Broadcast transactions and blocks
  - Validation blocks and transactions (wallets, miners)
  - Listening network requests (wallets, miners)
- Network discovery
  - getting own public IP
  - DNS requests (seed.bitcoin.sipa.be, dnsseed.bluematt.me, …)
  - address 24 propagation, getaddr request, connect callback address
  - Hard coded “Seed” addresses (peers.dat)
- Node connections
  - Manage up to 2.5K IPv4\v6 addresses
  - 8 outgoing (send) and up to 117 incoming (receive) bidirectional connections
- Data propagation - gossip (controlled flooding) protocol
  - Nodes are randomly connected
  - Only validated and unknown transaction and blocks are relayed

Bitcoin protocol messages

- **version**, **verack** – connection initialization
- **inv** (inventory vector) – data relaying (transactions, block, block header)
- **getdata** – returns requested data (transactions, block, block header)
- **addr** – broadcast address (version, IP4\6, port)
- **getaddr** – returns known addresses (init state)
- **getblocks** – returns list of blocks (init state)
- **getheaders** - returns list of block headers
- **tx, block** – reply to getdata (transaction, block)
- **ping** – connection is valid

Cryptocurrency wallets

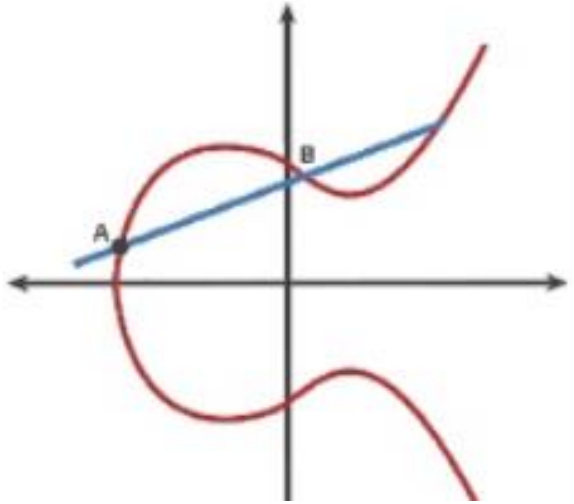
- Full node wallet (Bitcoin Core/Qt, Bitcoin Knots)
  - Stores full local blockchain
  - P2P/TCP, validation/listening/relaying
- Simplified Payment Validation (SPV) wallet
  - stores locally keys, block headers and some blocks
  - provides Simplified Payment Validation (SPV), P2P/TCP
- Lightweight (Jaxx)
  - stores locally keys but rely on validation outside, P2P/TCP or HTTP
- Web or online (Coinbase.com)
  - store keys and validation in server, HTTPS
- Hardware or cold (Ledger, Trezor)
  - Stores keys in encrypted offline storage, USB based
- Paper
  - Printed keys

Bitcoin transaction signing

- Bitcoin uses ECDSA to sign transactions
  - small keys size comparing to RSA
  - slow validation comparing to RSA
- Loosing private key means loosing money
- 256 bits ECDSA
  - private key – 32 bytes random number
  - public key – generated from private key 65 bytes (4,x,y) number
  - transaction signature – 71-73 bytes two numbers (r,s)

ECDSA

Elliptic Curve Digital Signature Algorithm



Bitcoin payment address\keys

18BgGGrBms3eGCRj2Vvrxso4XzWjJEEMBv

- Bitcoin supports a few payment methods:
  - P2PK – pay to public key – first payment approach
  - P2PKH – pay two public key hash (starts from 1):

```
hash = version + ripemd160(sha256(public_key)) # 21 bytes, version – network
dhash = sha256(sha256(hash)) # double sha256
sum = check_sum(dhash) # first 4 bytes of dhash
address = base58_encode_padded(hash + sum) # 25 bytes, 26-35 alphanumeric characters
```

  - P2SH – pay to script hash (starts from 3), new format
    - Script – a simple, stack-based, left->right language
- Mnemonic sentence or single key root can be used to generate\backup all key pairs

Bitcoin transaction structure

- Inputs (0+) – reference to the previous transaction(s)
  - previous tx – doubled sha256 transaction hash
  - scriptSig – ECDSA signature
  - index – input index in transaction
- Output (1+) – money destination(s)
  - value – number of Satoshi (1 BTC = 100M Satoshi)
  - scriptPubKey – destination address (P2PK, P2PKH, P2SH)
- Time (lock_time)

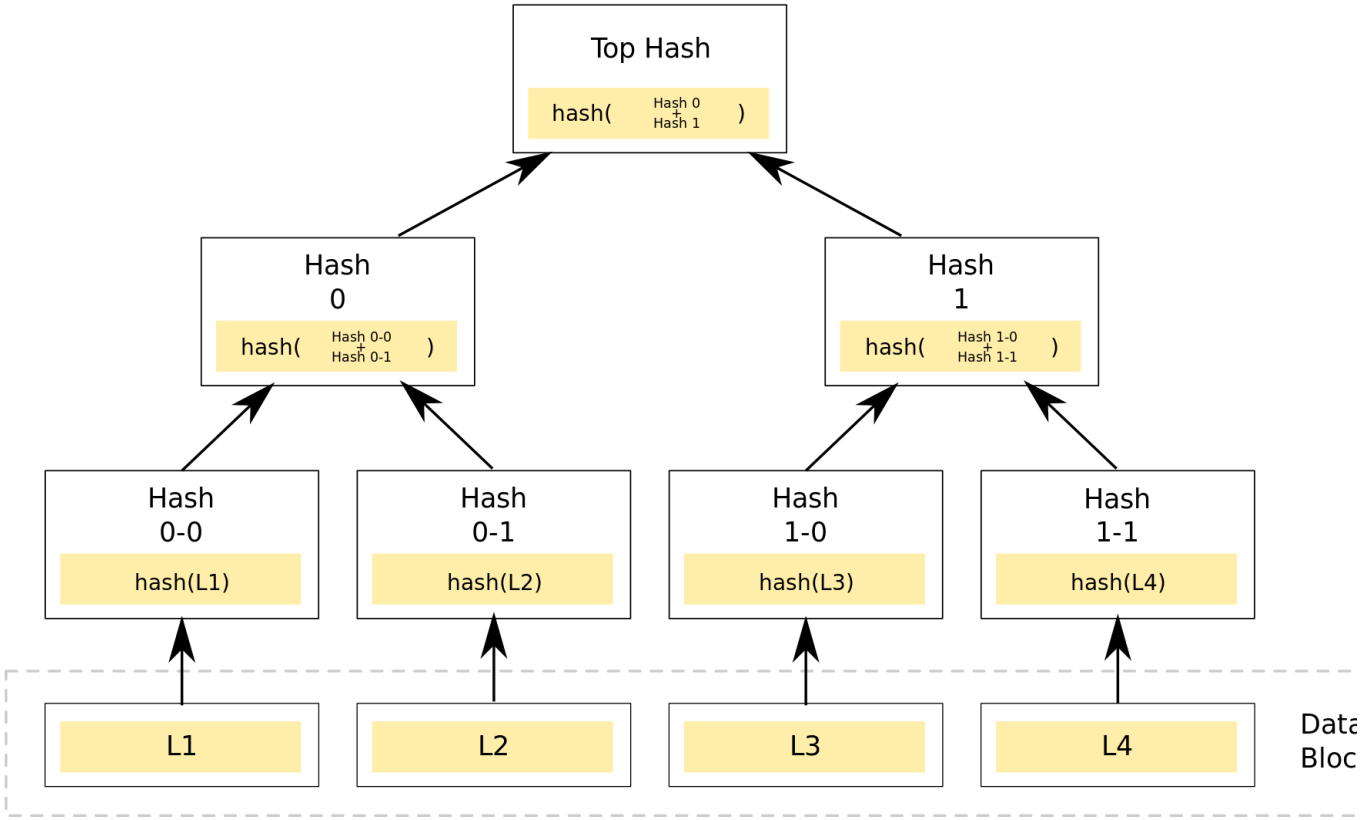
```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

Money transfer

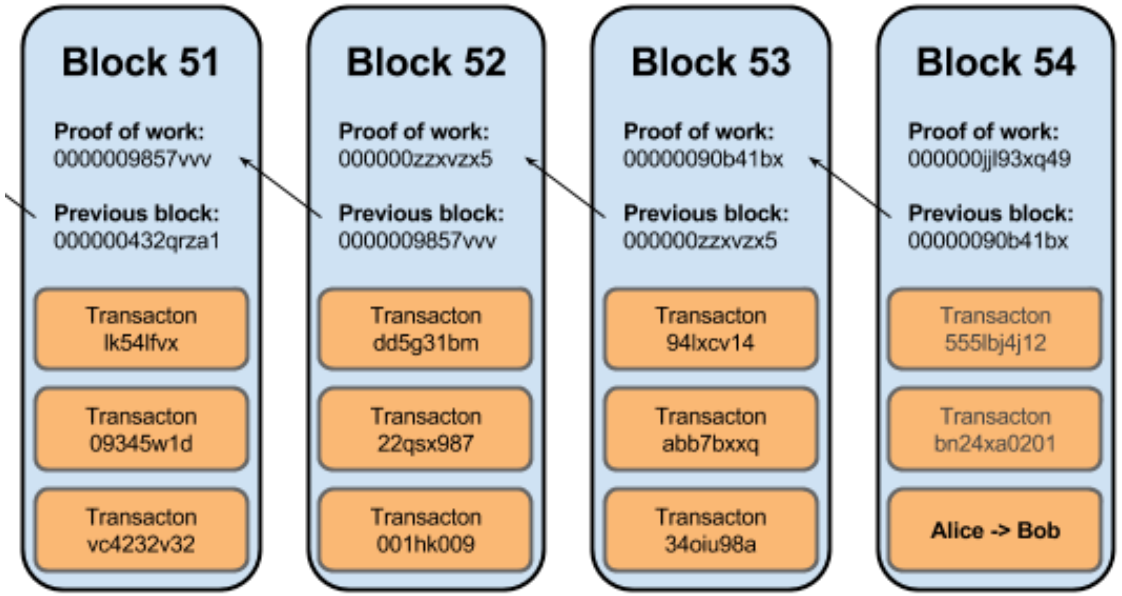
- Private key is required to send (sign) money
- Inputs are combined in one transaction if not enough money
- If sum inputs > requested outputs - output is a sender’s address
- Sum inputs >= Sum outputs
- Transaction fee = Sum inputs - Sum outputs
  - fee defines according to priority and network loading

Merkley\hash tree



- Bitcoin uses Merkley\hash tree to store\validate transactions integrity
- Doubled sha256 hash is used
- Labeling:
  - Leaf node – dhash(tran_data)
  - Non-leaf – dhahs(hash1 + hash2)
- Changing any transaction will cause top hash change
- $O(n^2)$ hashes – to create\validate tree

Bitcoin block structure



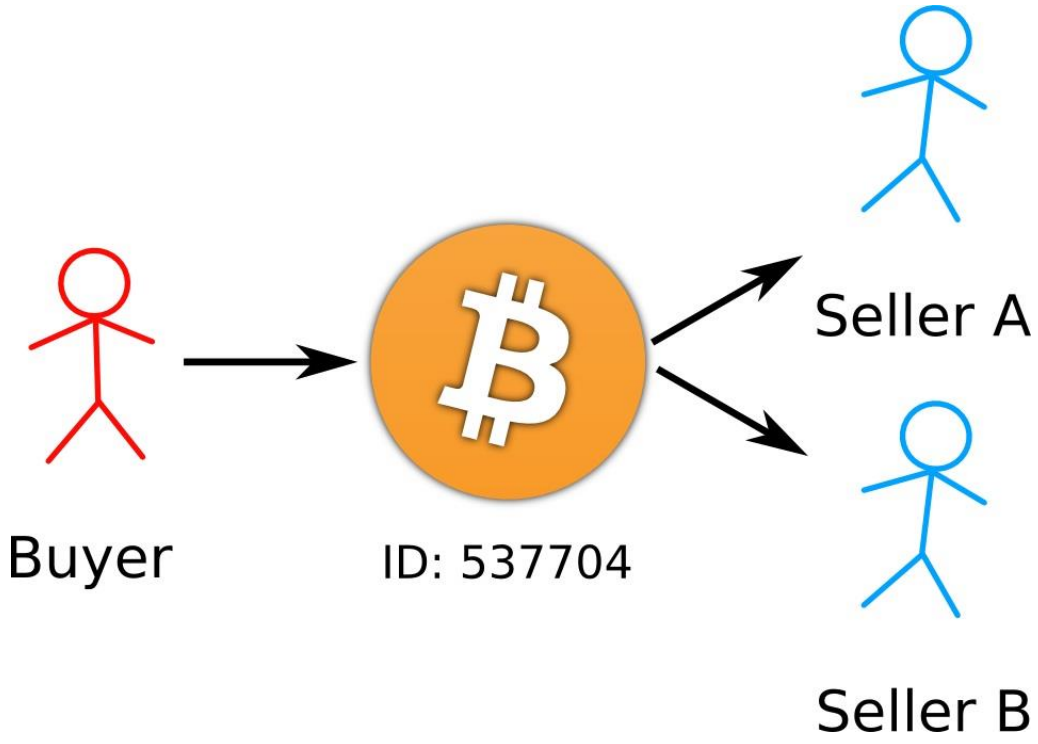
- Magic no – 4 bytes
- Block size – 4 bytes
- Block header – 80 bytes
  - version – 4 bytes
  - **hashPrevBlock** – doubled sha256 previous block hash
  - **hashMerkleRoot** - doubled sha256 hash of transactions
  - time – 4 bytes
  - bits – block target/difficulty, 4 bytes
  - **nonce** – 4 bytes number
- Transaction counter (1-9 bytes)
- Transactions

Bitcoin money genesis and fees

- First block (genesis) – one 50 BTC coinbase transacti
  - 50 BTC – initial miner incentive
- Every new block emits new money (50 BTC)
  - 1+ transaction(s) in block has no inputs (coinbase)
- Every 210K blocks (4 years) miner incentive is halved
  - Now the 2th half-life (12.5 BTC)
- Total limit – 21M BTC
- Now 16M BTC (80%) in network
- Miner incentive – genesis + all transaction fees
  - On average miner has 15-16 BTC per block

Double-spending problem

- One transaction is spent (sent) twice
- General solution – mint (central bank)
  - Money emission
- Bitcoin uses prof-of-work to prevent double-spending
- First transaction in blockchain is valid
- The right of building next block is given to miner who solved some crypto-puzzle



Prof-of-work and Hashcash

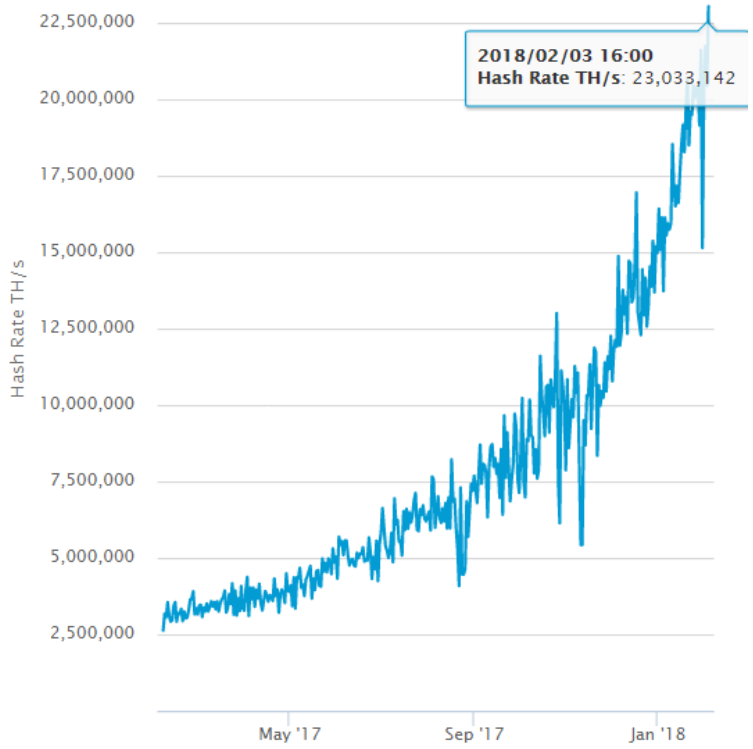
- Prof-of-work - to protect email span and DoS attacks
- Idea
  - Sender - CPU expensive task to send email/request
  - Receiver - CPU cheap task to validate email/request
- Implementation:
  - Email header: email-time-**counter** (nonce)
  - Calculate sha160 hash
  - If first 20 bits are zero – send email\request
    - hash <= target (2^{20}-1)
  - Else – counter++ and check hash again
  - 2^{20} hashes on average to get valid header (nonce)
  - Validate – hash(header) has 20 leading zeros
    - hash(header) <= target

Bitcoin mining

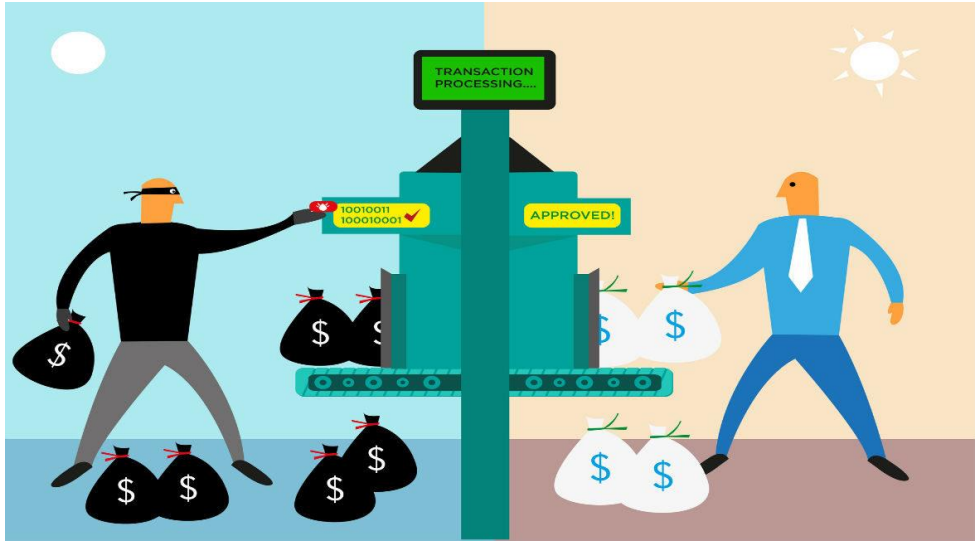
- Every miner collects transactions in queue
- Competition - find block hash <= target
  - min target = 2^{32}-1 (first 32 zeros)
  - now target about 2^{80}-1
- Winner - found first, gain incentive (genesis + fees)
- Collisions – lower hash wins
- Target is adjusted by DAA every 2016 blocks (2 weeks)
  - Goal - to build block every 10 mins

Miner software\hardware

- Software:
  - Block candidate preparation (pools)
  - Hash calculations
- Mining hardware (hash machines)
  - CPU
  - GPU
  - FPGA – Field-Programmable Gate Arrays
  - ASIC – Application Specific Integrated Circuits
    - Price - $2-15K, 4-15 TH/s
- Bitcoin network hash rate – 15-25 EH/s (exa hashes per sec)
  - 1 exa = 1M tera = 10^{18}



Transaction\block validation



- Transaction validation
  - Signature validation
  - Double-spent transaction (old clients)
- Block validation:
  - Block header hash is <= current target (DAA)
  - Size, incentive and fees validation
  - Merkley root hash ($O(n^2)$)
  - Transaction signatures (O(n))
  - Double-spent transactions
    - Every transaction input shouldn’t be already spent (O(n*k))

Anonymity protection

- To stay anonymous
  - address should be not mapped to person identity
  - one address per one transaction
  - empty addresses\keys should be deleted
  - wallets on Tor net or/and anonymous VMs
- Giving wallet keys means transferring money (physical keys exchange)
- Some web-wallets manage many addresses per one real address
  - A few virtual addresses mapped to one Bitcoin address
  - Internal transactions are virtual
- CoinJoin – some kind of “transaction cleaning”
- Exchanges between cryptocurrencies “hide” addresses
  - Local exchange history is evidence
- Printed Bitcoin banknote
  - Only note service knows purchase

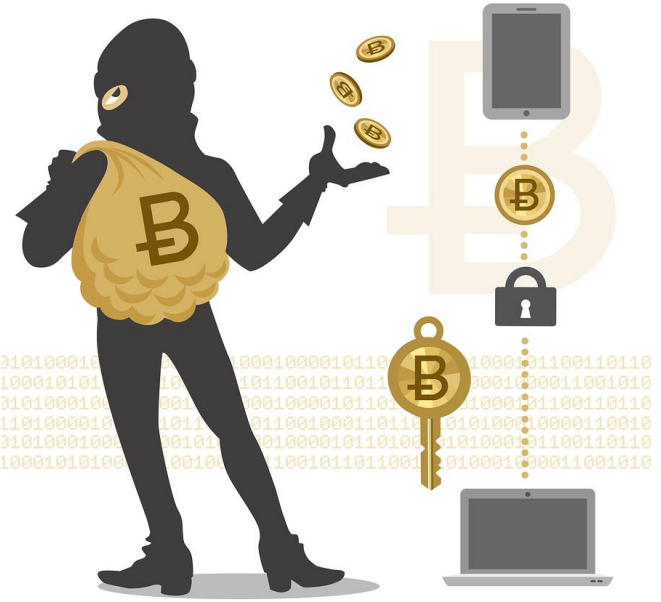
51% or majority attack

- 50%+ network power is controlled by one\group attacker(s)
  - Private branch with reverted attacker transactions
- Incidents:
  - July 2014 **gash.io** pool had 50%+ Bitcoin power hash
  - August 2016 – 51% attacks on Krypton and Shift (Ethereum based)
- Preventing 51% attack:
  - Mining pool is not “strong structure”
  - 6 blocks confirmation
    - attacker has to build 7+ blocks
  - Hardcoded block checkpoint to prevent reverting big branches
    - 100% hash power – 180 and 80 days to rebuild Bitcoin and Bitcoin Cash blockchains

DoS, Eclipse and ID attacks

- DoS – flooding invalid transactions and blocks
  - Easy to validate transactions and blocks
    - only valid transactions are relayed
  - Ban protocol (IP restriction)
- Eclipse attack – one node communicates only with malicious nodes (40% of nodes\IPs)
  - Random node selection in different network zones
  - Tried (known) and new address tables
- User profiling – getting user identity
  - Hard to do – attacker should be connected to all nodes
  - Bitcoin node can run upon Tor network (proxy)

Other attacks



- ID mapping and collision attack – trying to create the same private key (public key and address)
- Sybil attack – attacker creates a lot of identities (IP)
- Fake bootstrapping – new nodes can get malicious addresses
- Unauthorize resource access – steal private keys
  - Encryption and offline storage
- Man-in-the Middle (MITM) – routing P2P attack
  - Onion routing
- Many other attacks – P2P and crypto attacks

Sources

- Bitcoin Core source code https://github.com/bitcoin/bitcoin
- Bitcoin foundation https://bitcoin.org/en/
- Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto https://bitcoin.org/bitcoin.pdf
- Cryptocurrency networks: a new P2P paradigm http://downloads.hindawi.com/journals/misy/aip/2159082.pdf
- Bitcoin wiki https://en.bitcoin.it/wiki/Main_Page
- Bitcoin blockchain explorer https://blockexplorer.com/
- Bitcoin stats https://blockchain.info/
- Bitcoin node map http://bitnotes.earn.com
- Majority is not Enough: Bitcoin Mining is Vulnerable, Ittay Eyal and Emin G¨un Sirer, https://arxiv.org/pdf/1311.0243v2.pdf
- Bitcoin StackEchange https://bitcoin.stackexchange.com/

Thank you!