Crypto

Principles of Distributed Systems

Tomáš Faltín

KSI MFF

Bitcoin

History

- Can I trust that the money is authentic?
- Can the digital money be spent only once?
- Can no one else claim my money?

- Paper "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto in 2009

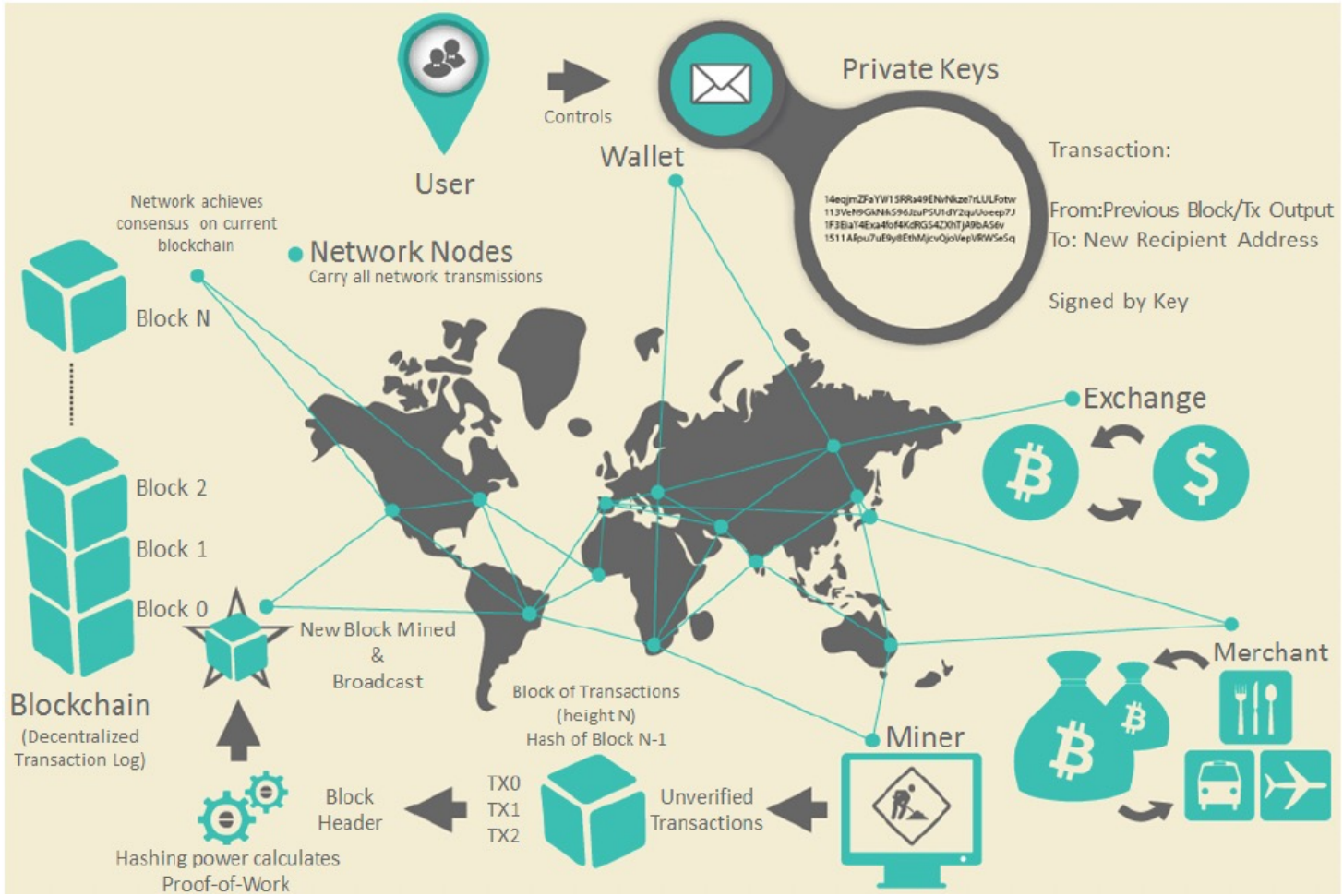
Overview

- Concepts & technologies for digital money ecosystem
- Distributed, P2P system
- bitcoins created through mining every ~10 min
  - Mining = competing to find solution to a mathematical problem while processing transactions
  - Limited to 21 mil. (by 2140)

Bitcoin

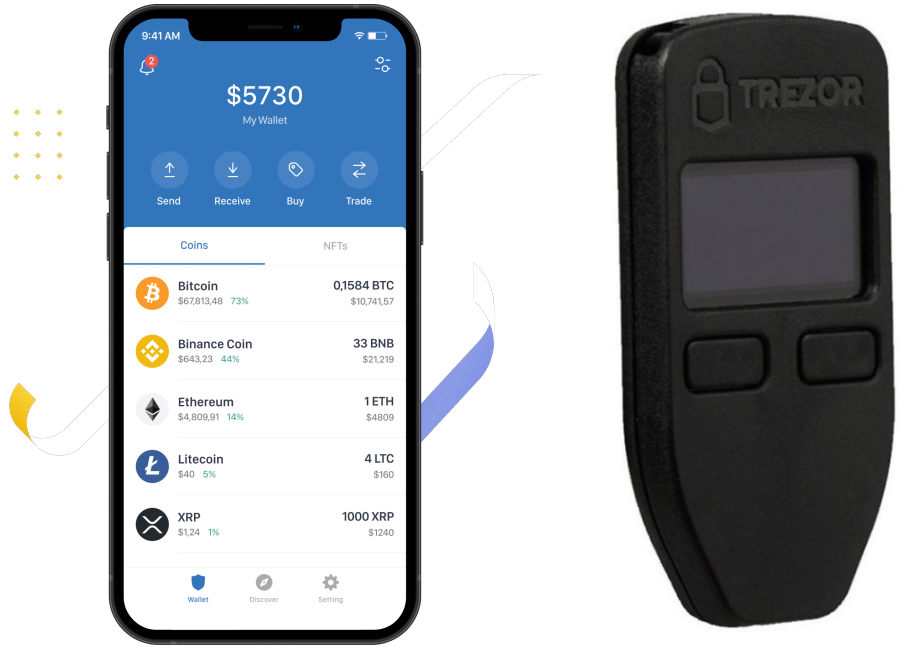
- The Bitcoin protocol - A decentralized P2P network
- Blockchain - A public transaction ledger
- Consensus Rules - A set of rules
- Proof-Of-Work algorithm A mechanism for reaching consensus

High Level


User
Controls
Wallet
Private Keys
Transaction:
From:Previous Block/Tx Output
To: New Recipient Address
Signed by Key
Network achieves consensus on current blockchain
Network Nodes
Carry all network transmissions
Block N
Block 2
Block 1
Block 0
Blockchain
(Decentralized Transaction Log)
New Block Mined & Broadcast
Block of Transactions (height N) Hash of Block N-1
Exchange
Merchant
Miner
Unverified Transactions
TX0
TX1
TX2
Block Header
Hashing power calculates Proof-of-Work

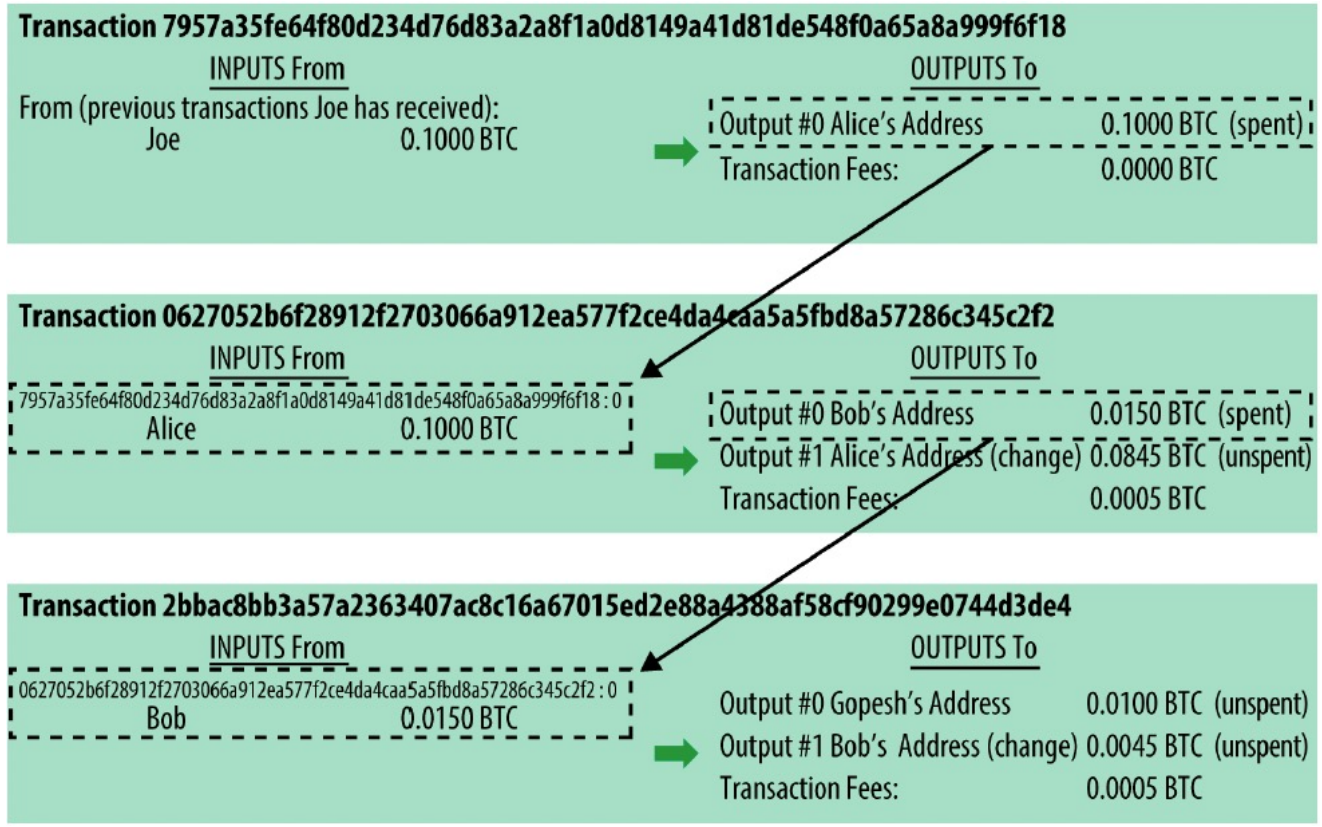
Wallet

- UI to the bitcoin interface
- Desktop, mobile, web, HW, paperwallets, …
- A bitcoin node that stores and manages keys

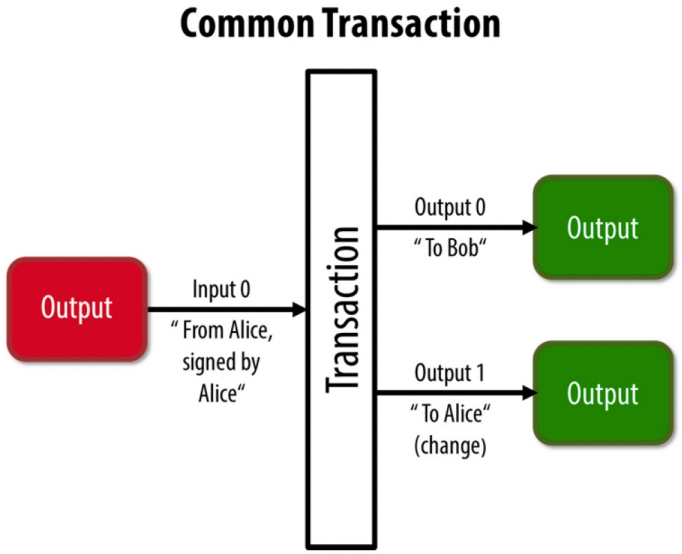


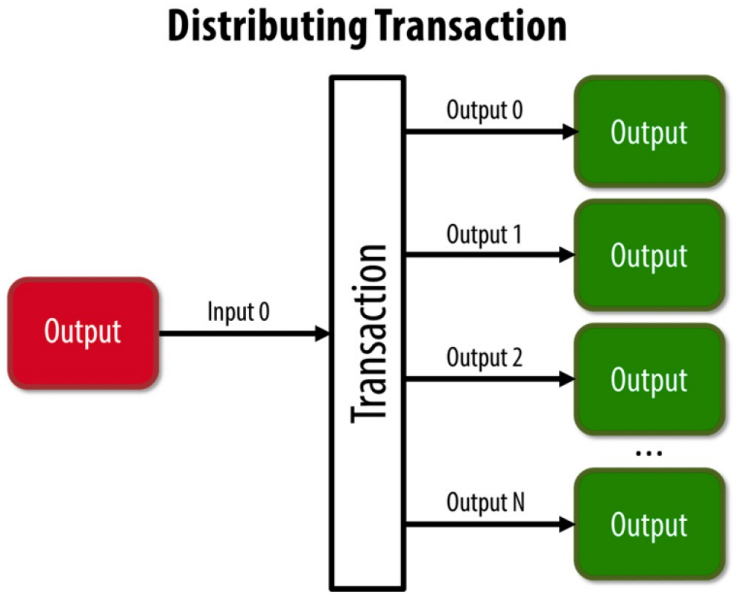
Transactions

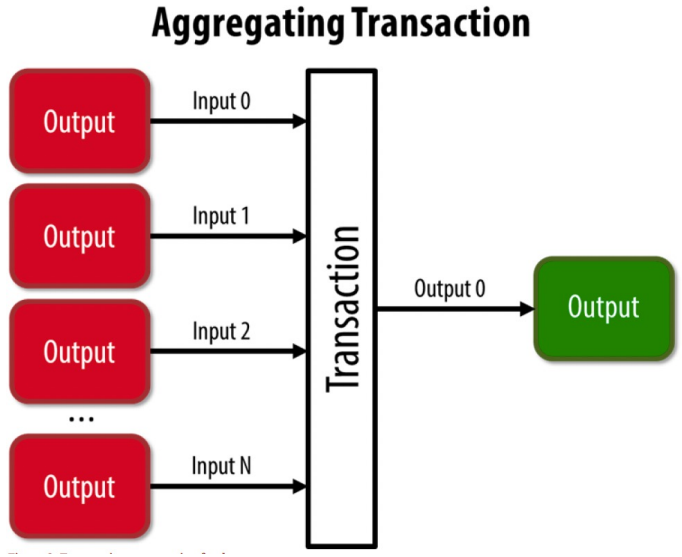
- Bitcoin transfer from the owner to another user
- Propagation via flooding mechanism



Transactions

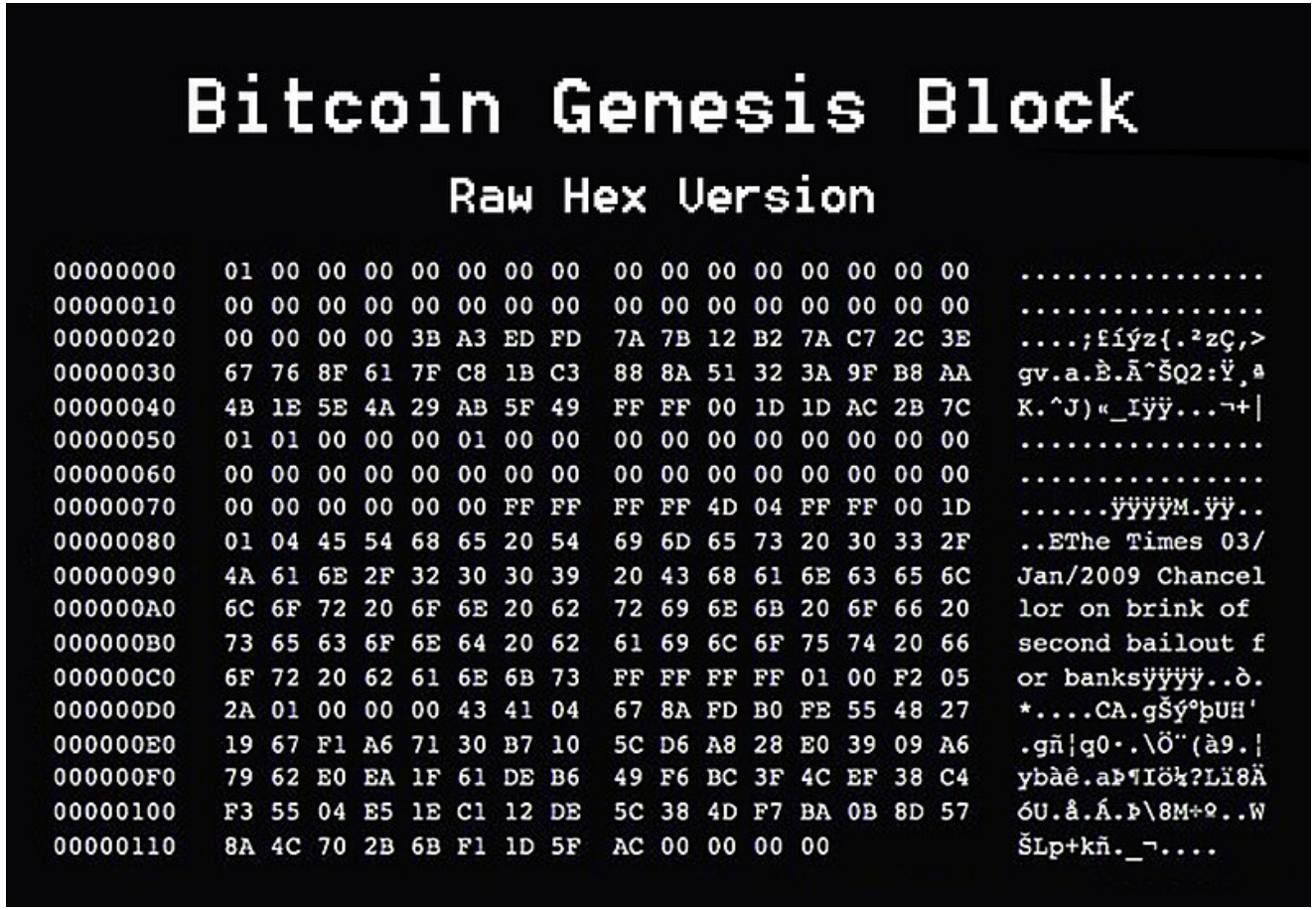






Mining

- Process of including transactions into the blockchain
- Validation of transactions by solving a puzzle (Proof-of-Work)
- Creation of new bitcoins (reward, currently 6.25 BTC)



Going Into Details

Bitcoin Ownership

Digital keys

Private secret key

Public key

Bitcoin address

Generated from the public key

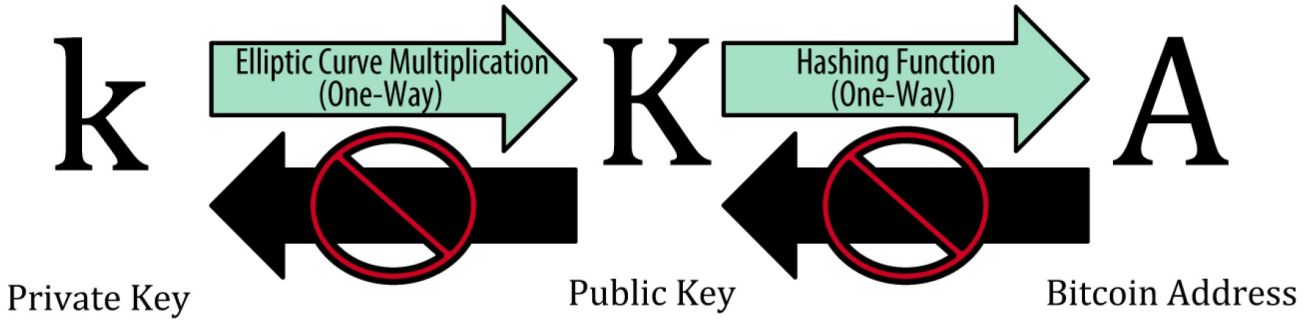
Digital signature

(Most) transactions require to be included in blockchain

Generated from the secret key

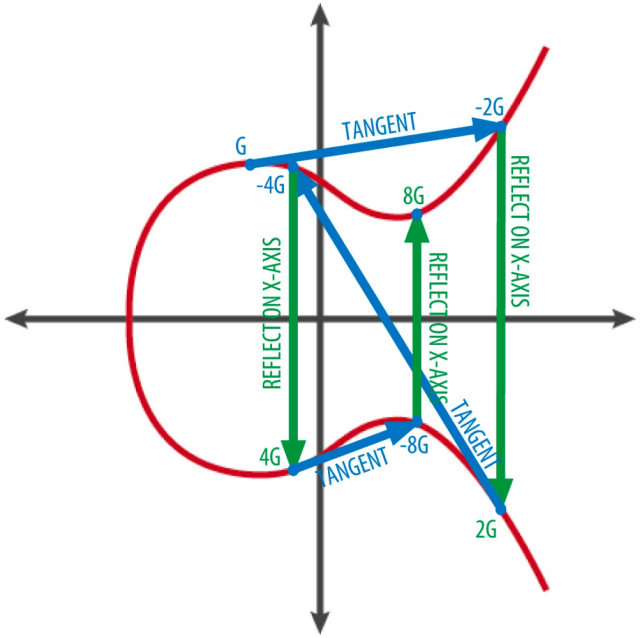
Public And Private Keys

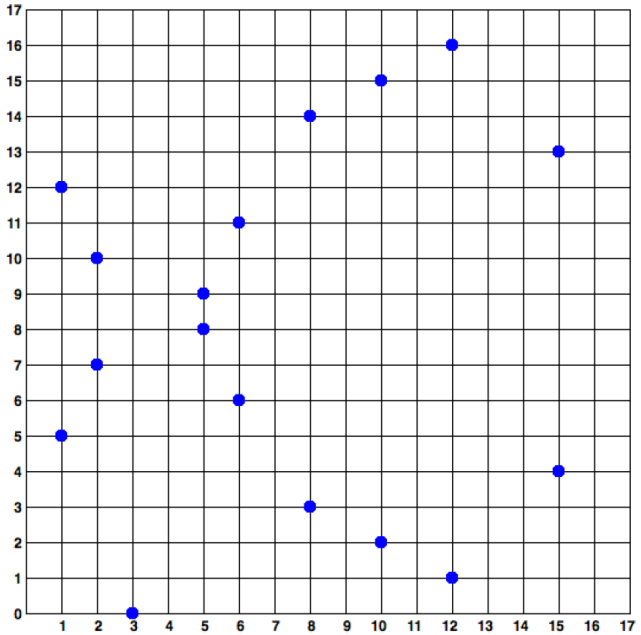
- Need an irreversible function (**elliptic curve multiplication**, prime number exponentiations, …)
- Digital signatures through asymmetric cryptography
- Private key = a number between <1,2^{256})



Public Keys

- Calculated from private key using elliptic curve multiplication
  - K (public key) = k (private key) * G
  - f(x, y): $y^2 \bmod p = (x^3 + 7) \bmod p$
  - $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
  - defined by a secp256k1 standard
- Reverse operation: finding the discrete logarithm

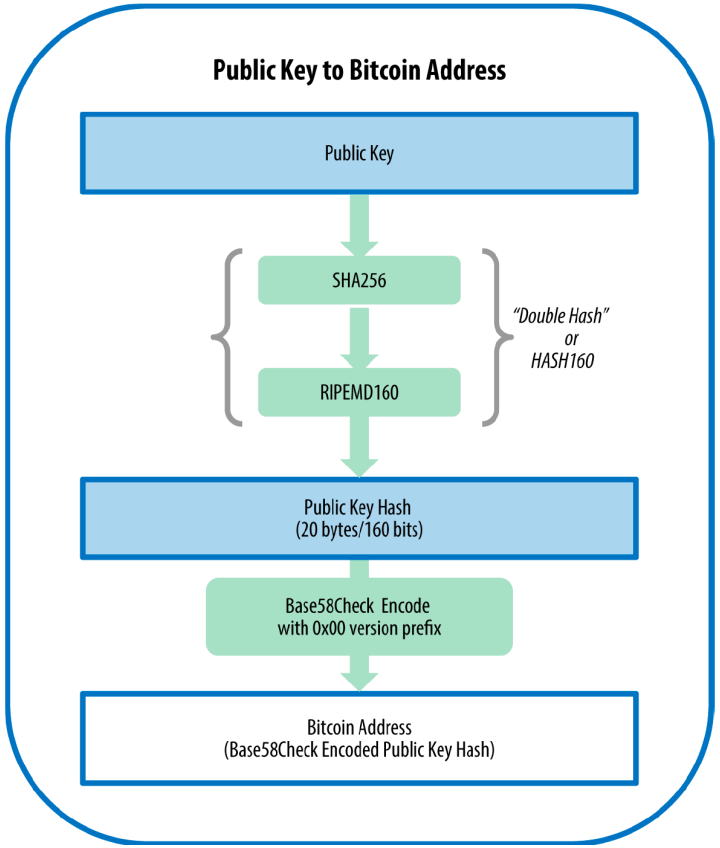




Addresses

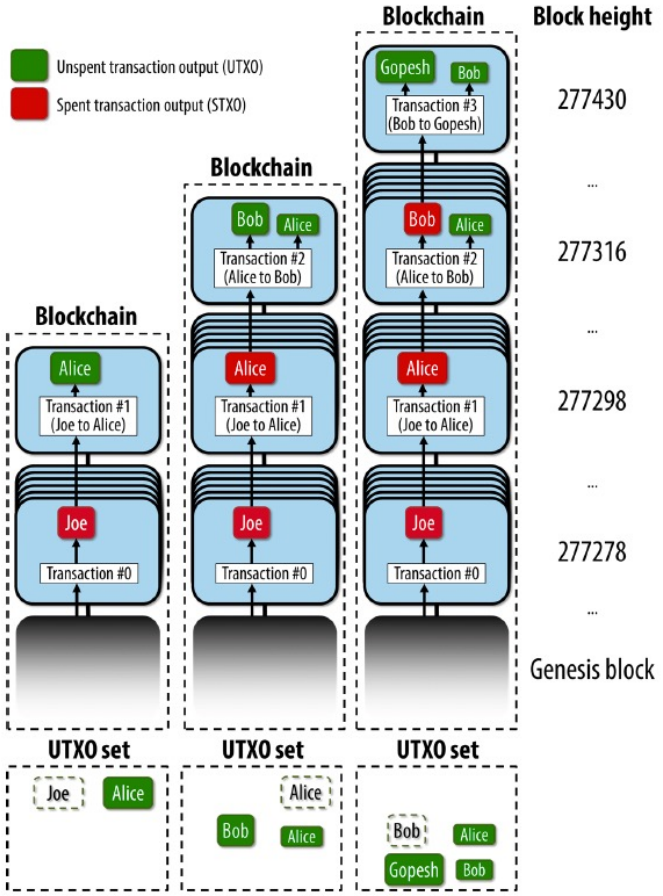
- Produced from a public key and starts with 1
  - SHA256, RIPEMD160
  - A = RIPEMD160(SHA256(K))
- Encoded using Base58Check
- Use only x coordinate (y can be computed)

- Pay-to-Script addresses
  - Address created from a script
- Multisig Addresses
  - Requires M-of-N signatures



Transactions

- Transfer of value between participants
- UTXO = unspent transaction outputs
  - Full nodes track UTXO set
- Received bitcoin ~ blockchain contains an UTXO
- Change returned as another transaction output
- Coinbase transactions = reward for the miners
  - Creates new bitcoins



Transaction Outputs

- An amount of bitcoin in satoshi (0.00000001 BTC)
- scriptPubKey (locking script/witness script)
    - A cryptographic puzzle determining the requirements

```JSON
"vout": [
  {
    "value": 0.01500000,
    "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY
    OP_CHECKSIG"
  },
  {
    "value": 0.08450000,
    "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
  }
]
```

Transaction Inputs

- Indentifier of UTXO
- Output index
- scriptSig - unlocking script
- sequence

JSON

```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig" :
"3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298c
ad530a863ea8f53982c09db8f6e3813[ALL]
0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b
4a10fa336a8d752adf",
    "sequence": 4294967295
  }
]
```

Transaction Fees

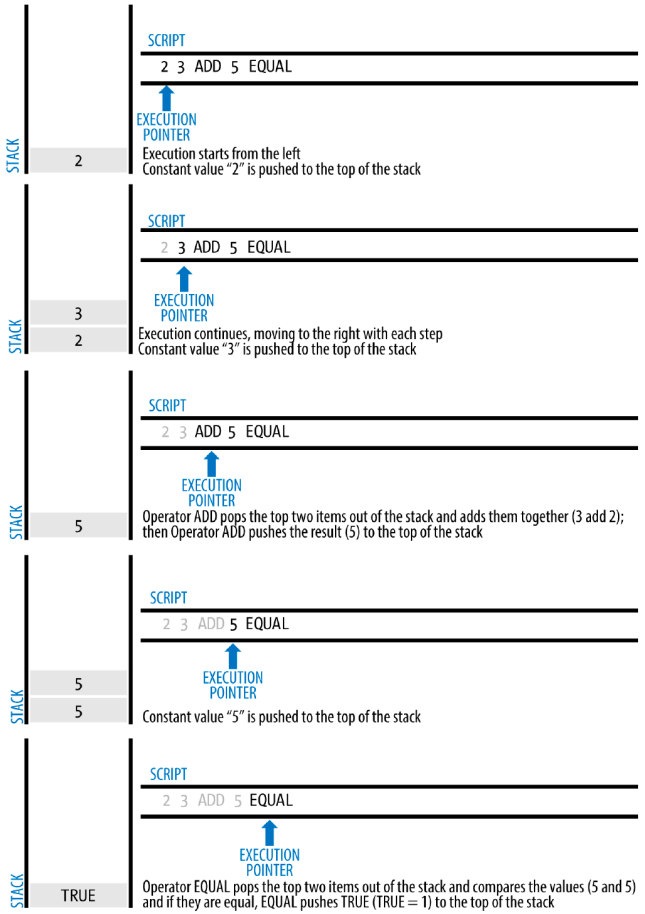
AFFECT THE PROCESSING PRIORITY

CALCULATED BASED ON THE SIZE OF TRANSACTION (IN KB) OR DYNAMICALLY

IMPLIED FROM THE DIFFERENCE BETWEEN INPUTS AND OUTPUTS

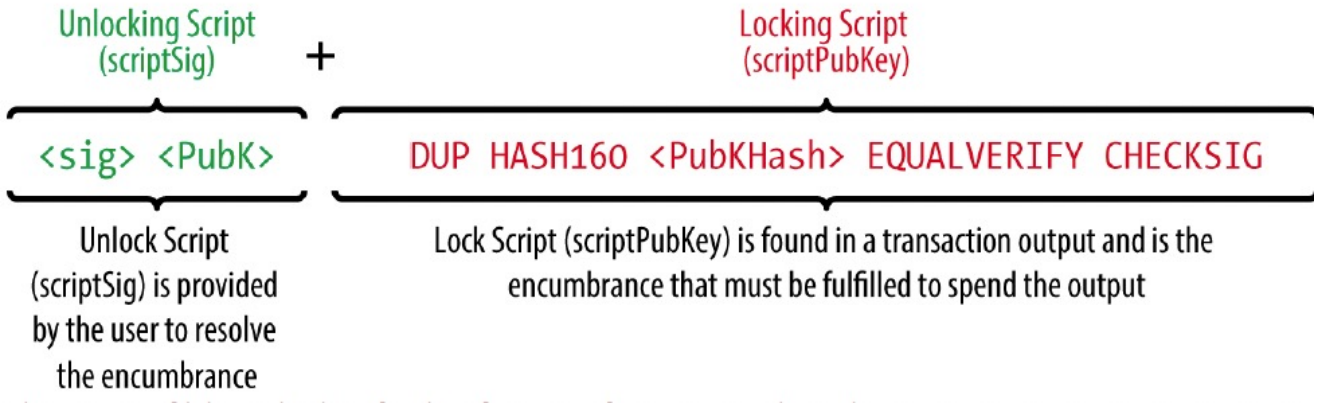
Script Language

- Script, Forth-like stack-based execution language
- Stateless, Turing incomplete



Pay-To-Public-Key-Hash (P2PKH)

- Major transaction's script



Digital Signatures

- Undeniable proof for authorization
- Transaction cannot be modified after signing
- Each transaction signed independently

- Using ECDSA
- Functions: CHECK(MULTI)SIG, CHECK(MULTI)SIGVERIFY
- SIGHASH (Signature Hash Type): ALL, NONE, SINGLE, ANYONECANPAY

```
3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298ca
d530a863ea8f53982c09db8f6e381301
```

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Creates a temporal private + public key pair
- Signing: $S = k^{-1} (Hash(m) + dA * R) \bmod n$
  - k - a temporal private key
  - R - x coordinate of the temporal public key
  - dA - the signing private key
  - m - the transaction
- Verification: $P = S^{-1} * Hash(m) * G + S^{-1} * R * Qa$
  - R, S - signature values
  - Qa - public key
  - M - the transaction
  - G - generator point

```
3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e381301
```

Multisignature

- X-of-3 multisignature
- X-of-15 in P2SH

```
<Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
```

Pay-to-Script-Hash (P2SH)

- Script replaced with a hash fingerprint (redeem script)
- Script must be presented on spending
- Not recursive
- Advantages
  - Makes the transaction smaller
  - Shifts burden from output (UTXO) into input (blockchain)

```
<2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG> HASH160 <redeem scriptHash> EQUAL
```

```
<Sig1> <Sig2> 2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG
```

Data Recording Output (RETURN)

- Create transaction outputs to record data on the blockchain
  - E.g., proof-of-existence of a file (https://proofofexistence.com/)
- 80B data portion
- Create a provably unspendable output which does not need to be stored in UTXO set

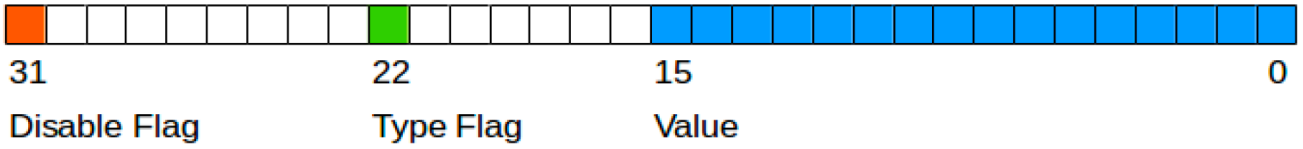
Absolute Timelocks

- nLockTime – Per-transaction timelock
  - < 500mil. – not valid before the given block height
  - >= 500mil. – not valid before the given UNIX epoch
  - Double spending problem
- Check Lock Time Verify (CLTV) – Per-output timelock
  - UTXO can be spent with nLockTime set to >=

```
<now + 3 months> CHECKLOCKTIMEVERIFY DROP DUP HASH160 <Bob's Public Key Hash> EQUALVERIFY CHECKSIG
```

Relative Timelocks

- nSequence ($< 2^{31}$)
  - Behavior based on type flag
    - 0: Not valid before given number of blocks, or
    - 1: Not valid before given number of seconds (multiples of 512s)
- CHECKSEQUENCEVERIFY (CSV)



Median Time Passed

- Timestamps set by miners
  - Potential motivation to lie
- Median-Time-Passed is a median timestamp of last 11 blocks (~2h)
  - Consensus time for timelock operations

Fee Snipping Attack

- to snipe higher-fee transaction from future to rewrite past blocks
- Fixed by limiting nLockTime to the next block

Flow Control (if-else)

- IF-ELSE-ENDIF
  - Can be nested

```
condition
IF
  code to run when condition is true
ELSE
  code to run when condition is false
ENDIF
code to run in either case
```

- VERIFY guard clause – terminates if FALSE

```
HASH160 <expected hash> EQUALVERIFY <Bob's Pubkey> CHECKSIG
```

A Complex Example

```
01  IF
02    IF
03      2
04    ELSE
05      <30 days> CHECKSEQUENCEVERIFY DROP
06      <Abdul the Lawyer's Pubkey> CHECKSIGVERIFY
07      1
08    ENDIF
09    <Mohammed's Pubkey> <Saeed's Pubkey> <Zaira's Pubkey> 3 CHECKMULTISIG
10  ELSE
11    <90 days> CHECKSEQUENCEVERIFY DROP
12    <Abdul the Lawyer's Pubkey> CHECKSIG
13  ENDIF
```

- Unlocking script A: `0 <Mohammed's Sig> <Zaira's Sig> TRUE TRUE`
- Unlocking script B: `0 <Abdul the Lawyer's Sig> <Saeed's Sig> FALSE TRUE`
- Unlocking script C: `<Abdul the Lawyer's Sig> FALSE`

Segregated Witness (segwit)

- Architectural change of separating unlocking script/scriptSig
- Data moved into a separated witness data
- Soft Fork

Pay-to-Public-Key-Hash (P2PKH)

- Old P2PKH

```
[...]
“Vin” : [
"txid": "0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2",
"vout": 0,
         "scriptSig": “<Bob’s scriptSig>”,
]
[...]
```

```
DUP HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 EQUALVERIFY CHECKSIG
```

Pay-to-Witness-Public-Key-Hash (P2WPKH)

- P2WPKH
  - A witness version
  - A witness program
- Similar for P2WSH

```
[...]
“Vin” : [
"txid": "0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2",
"vout": 0,
         "scriptSig": “”,
]
[...]
“witness”: “<Bob’s witness data>”
[...]
```

```
0 ab68025513c3dbd2f7b92a94e0581f5d50f654e7
```

Segregated Witness Transaction Identifiers

- Transaction signatures could have been modified
  - Potential for DoS
- 2 new fields:
  - Txid – double SHA256 hash of transaction **without** the witness data
  - Wtxid – double SHA256 hash of transaction **with** the witness data

Segregated Witness' New Signing Algorithms

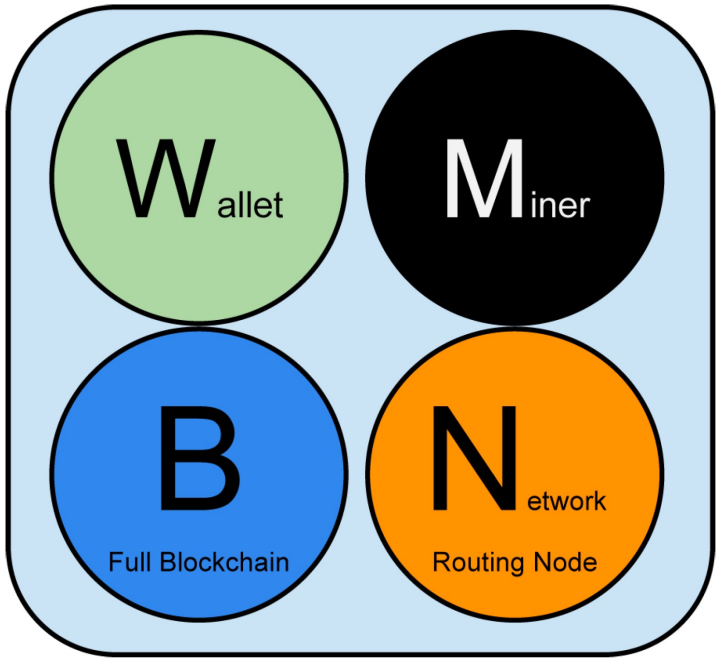
- CHECK(MULTI)SIG(VERIFY)
- Signatures applied on commitment hash
  - SIGHASH_ALL uses all inputs and outputs
- Hash operations increases in $O(n^2)$ with number of operations in Tx
  - Very large transactions can cause performance problem (DoS) of the whole network

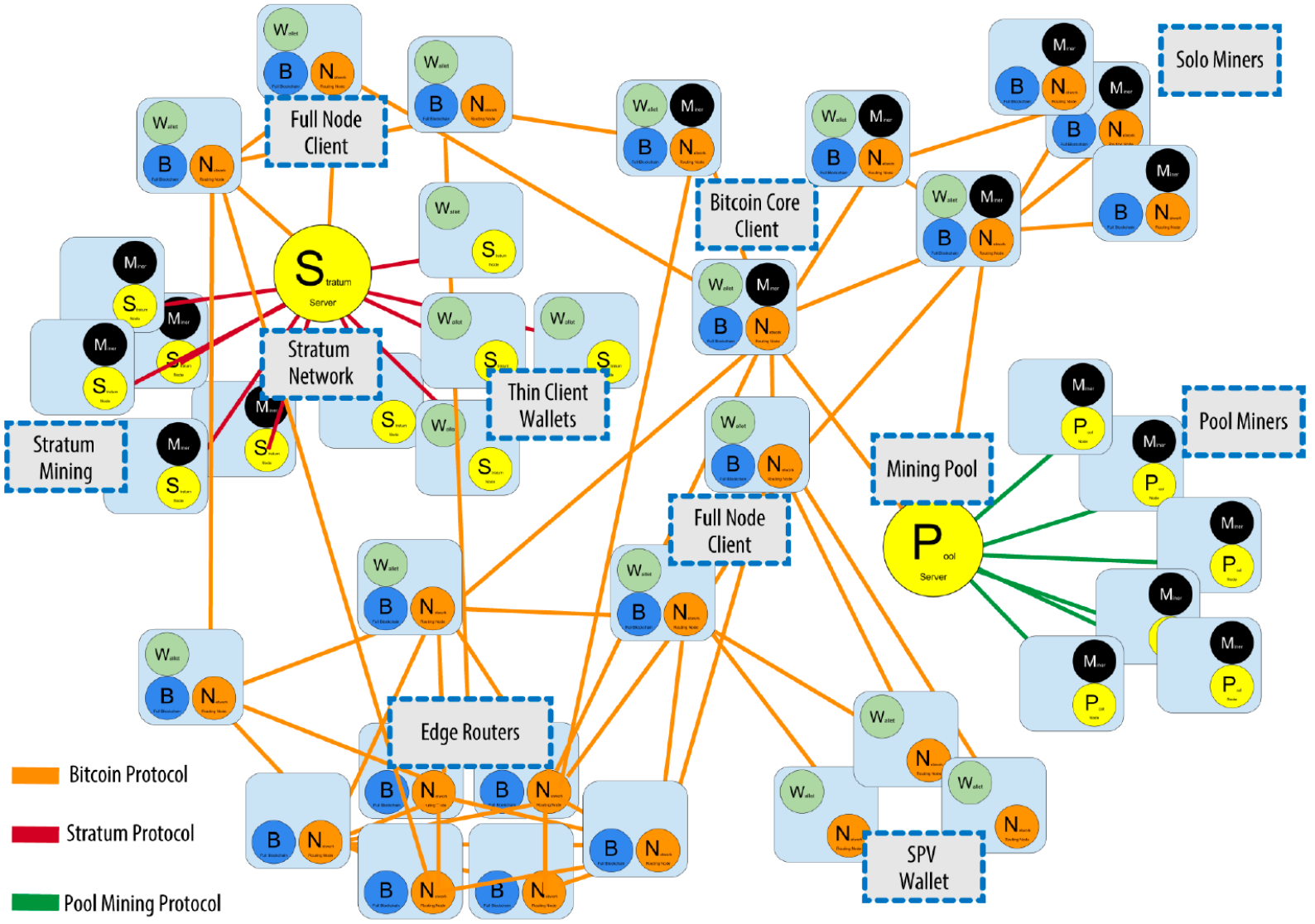
- New algorithm increases only in O(n)
- Commitment hash includes the value of each input

The Bitcoin Network

Network Architecture

- Bitcoin protocol = P2P protocol on top of the internet
- Stratum protocol = protocol for lighweight devices
- Node’s functions: wallet, mining, blockchain, routing




Full Node Client
Solo Miners
Bitcoin Core Client
Stratum Server
Stratum Network
Thin Client Wallets
Stratum Mining
Pool Miners
Mining Pool
Pool Server
Full Node Client
Edge Routers
SPV Wallet
Bitcoin Protocol
Stratum Protocol
Pool Mining Protocol

Bitcoin Relay Networks

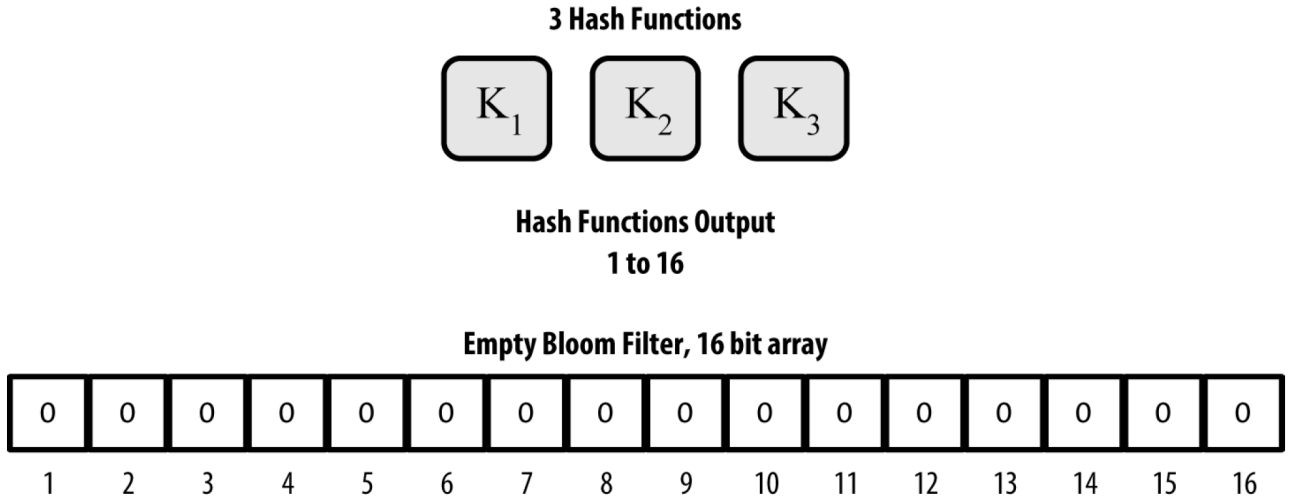
- A network minimizing latency between miners
- V1: Bitcoin Relay Network
- V2: FIBRE – UDP-based

Simple Payment Verification (SPV) Nodes

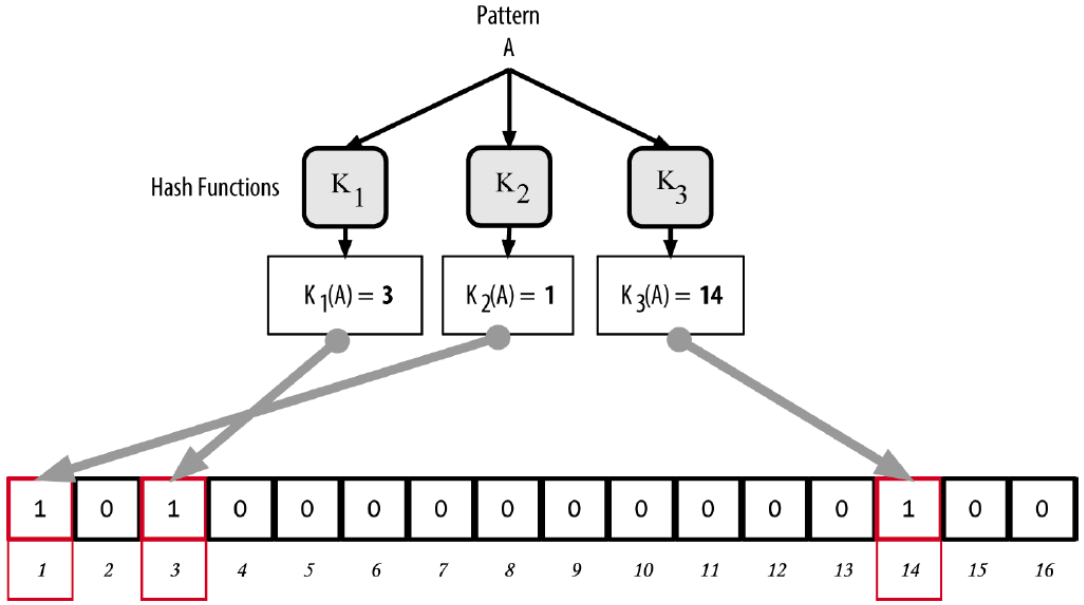
- Not storing the whole blockchain
- Download only block headers
- Verify transactions using views on demand from peers
  - SPV node establishes link between the transaction and the block containing it using merkle path
  - +6 additional blocks atop of the block with the transaction
- Cannot verify, e.g., double spending of transactions
  - Double spending, DoS, network partition or Sybil attacks
    - Connect randomly to several nodes to decrease probability

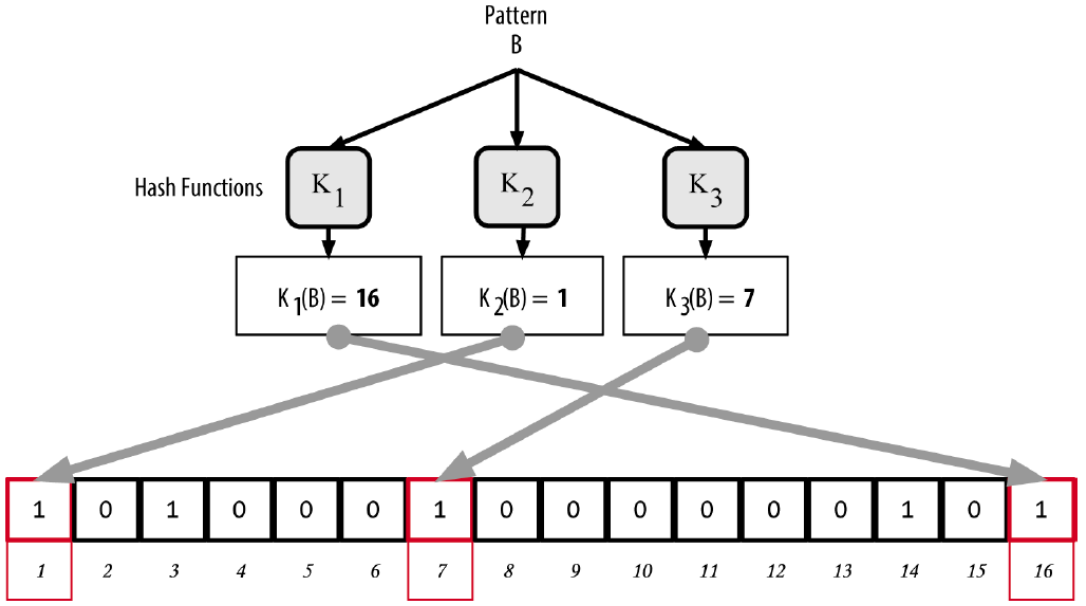
Bloom Filters

- A probabilistic filter protecting privacy
- A bitset of size N and variable number of M hash functions

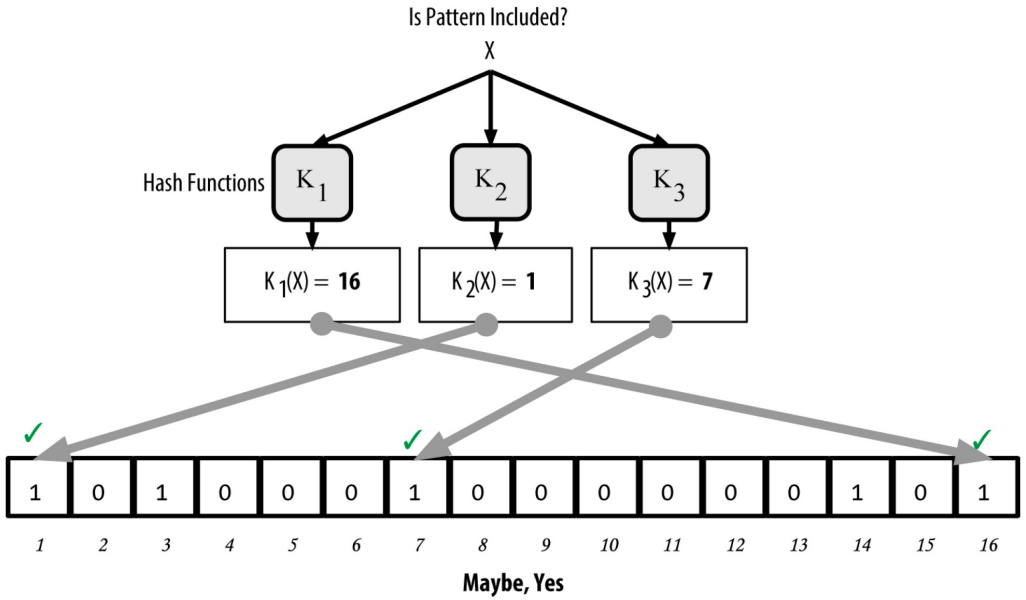


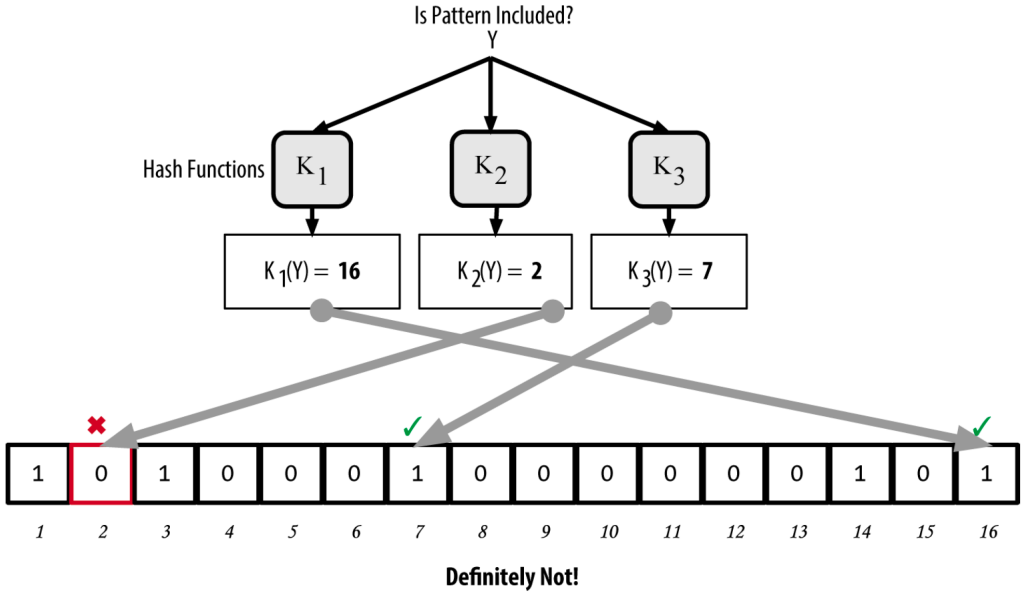
Bloom Filters - Insert





Bloom Filters - Search





Bloom Filters In SPV

- To filter interesting transaction received from peers
- Extract public key hash, script hash and transaction ID from UTXO controlled by the wallet into the bloom filter
- Send filter load to a peer along with the filter
- A peer pre-filters transaction for SPV

Encrypted And Authenticated Connections

- Problem only for SPV nodes
- Tor Transport (The Onion Routing network)
  - Network offering encryption and encapsulation of data through randomized paths
- P2P Authentication & Encryption
  - Allow nodes to authenticate using ECDSA

Transaction Pool (Memory Pool/Mempool)

- A temporal list of unconfirmed transactions
- Sometimes a separated pool for orphaned transactions
  - Orphan = transaction with a missing parent
- Sometimes UTXO pool
  - Set of unspent outputs

The Blockchain

The Blockchain

- An ordered back-linked list of blocks of transactions
- Each block identified by SHA256 of its header
  - The header contains a parent header and thereby affect the hash
  - (Can be referenced by the height as well)
- Each block references a parent block
  - The genesis block = the first block ever

Structure Of The Block

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1–9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

Structure Of The Header

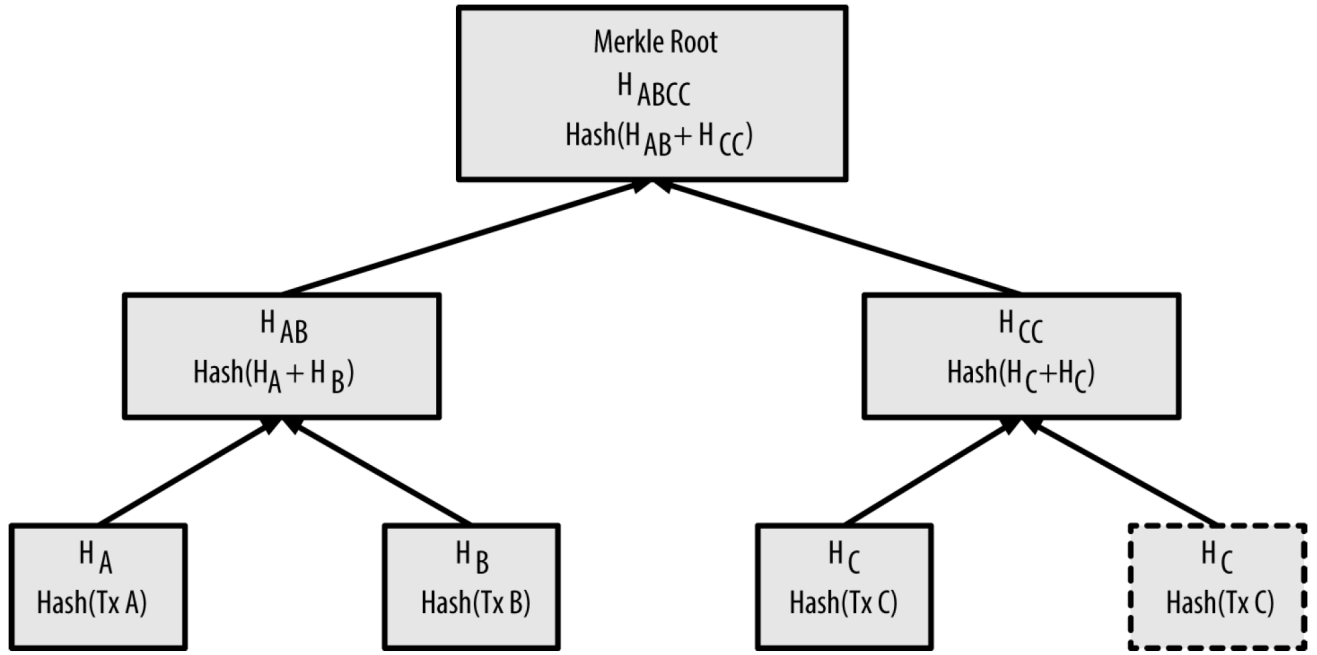
Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (in seconds elapsed since Unix Epoch)
4 bytes	Difficulty Target	The Proof-of-Work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

Merkle Trees (Binary Hash Trees)

- A data structure used to efficiently summarizing and verifying the integrity of large dataset
- A binary tree containing cryptographic hashes
- Need 2 * $\log_2 N$ calculations to check for element existantance

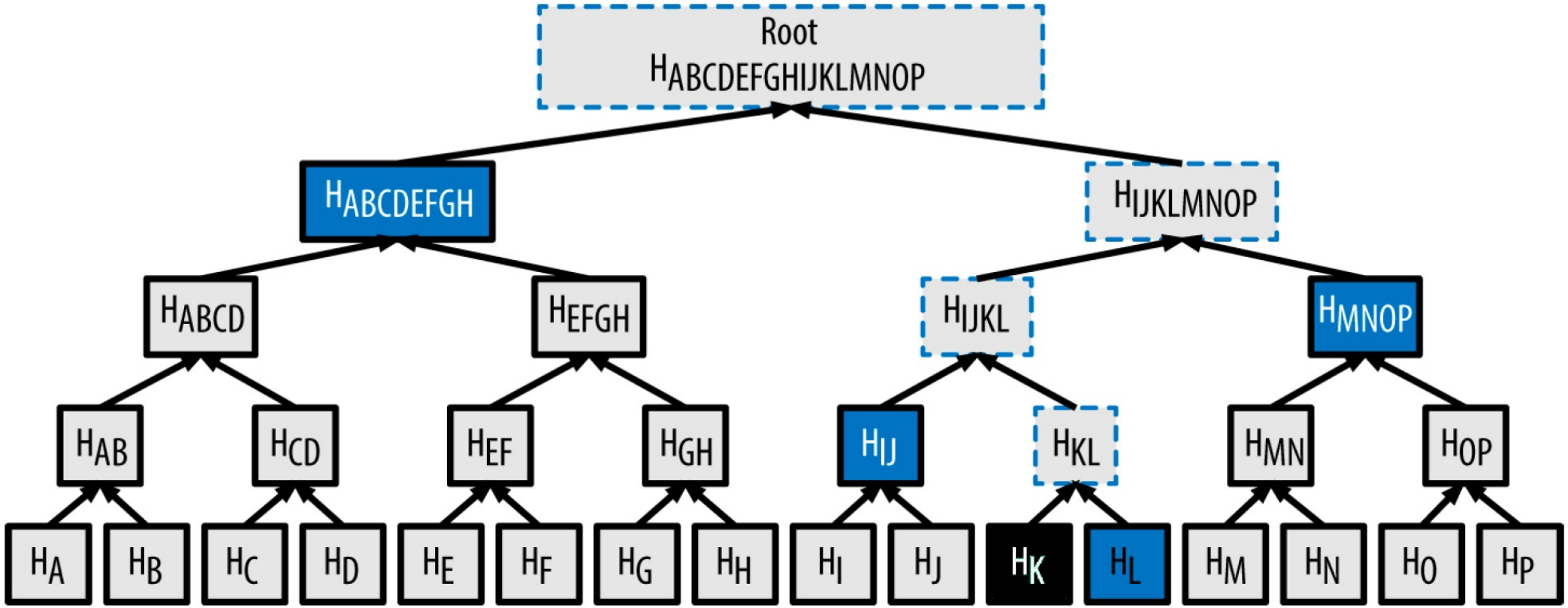
Merkle Trees - Building

- Constructed bottom-up
- H_X = SHA256(SHA256(X))
- H_{X+Y} = SHA256(SHA256(H_X+H_Y))



Merkle Trees - Search

- Merkle path



Merkle Trees in SPV

- Merkle path used for verification of a transaction inside a block

Mining & Consensus

Mining

- A decentralized mechanism of clearing and validation of transactions
- Enables consensus without central authority
- Miners validate and record transactions on the global ledger
- Rewards for miners: A coinbase/block reward and transaction fee
- Miners compete in solving a difficult math problem based on hash
  - Proof-of-Work = solution to the problem
- The maximum amount of newly created BTCs halves every 210k blocks (~ 4 years)

Decentralized Consensus

- Bitcoin has no central authority
- The consensus is an emergent artifact of the asynchronous interaction of independent nodes all following simple rules

Independent Transaction Verification

- Each node validates forwarded transactions agains many criterias
  - Syntax, non-empty inputs/outputs, values in ranges, …

Aggregating Transactions Into Blocks

- After validation, transactions added into memory pool
- Transaction aggregated into a candidate block
  - Adding the coinbase transaction
    - Does not consume UTXO inputs

Size	Field	Description
32 bytes	Transaction Hash	All bits are zero: Not a transaction hash reference
4 bytes	Output Index	All bits are ones: 0xFFFFFFFF
1–9 bytes (VarInt)	Coinbase Data Size	Length of the coinbase data, from 2 to 100 bytes
Variable	Coinbase Data	Arbitrary data used for extra nonce and mining tags. In v2 blocks; must begin with block height
4 bytes	Sequence Number	Set to 0xFFFFFFFF

Proof-of-Work Algorithm

- Hashing the block header repeatedly, changing one parameter until the resulting hash matches a specific target
  - Nonce = A number used to vary outputs of the hash function

Adjusting Difficulty

- The target determines the difficulty of the Proof-of-Work algorithm
- The target is set so the mining power result in 10-minutes intervals
- Each node retargets the Proof-of-Work algorithm every 2,016 blocks
  - Compares with expected time and adjust accordingly
  - To avoid extreme volatility, change is not more than factor of 4

Validating A New Block

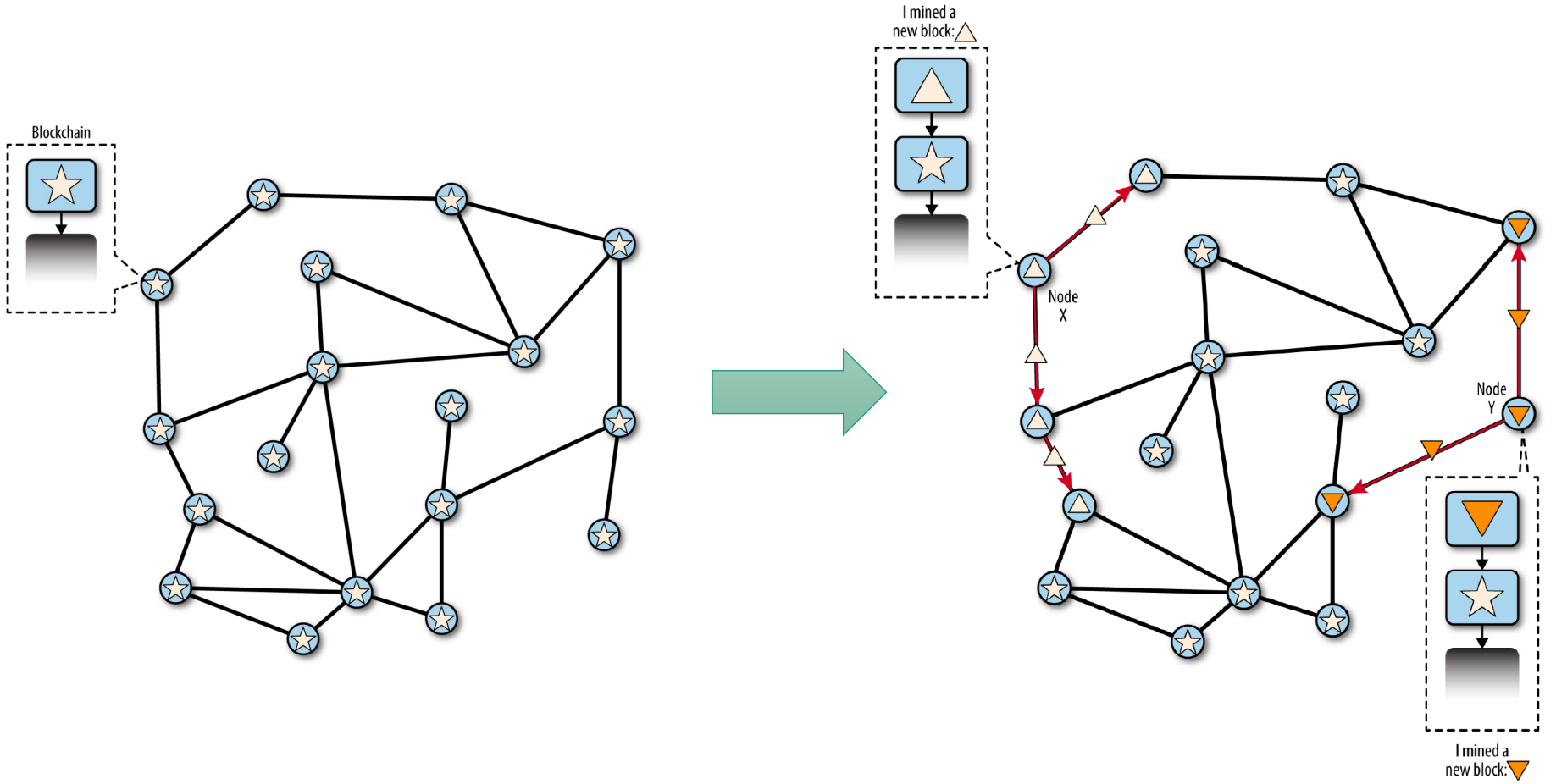
- An independent validation by every node of the network
  - Miners cannot cheat
- A lot of validation criterias
  - Syntax, block header hash, block timestamp, block size, coinbase transaction, all transactions are valid

Assembling And Selecting Chains Of Blocks

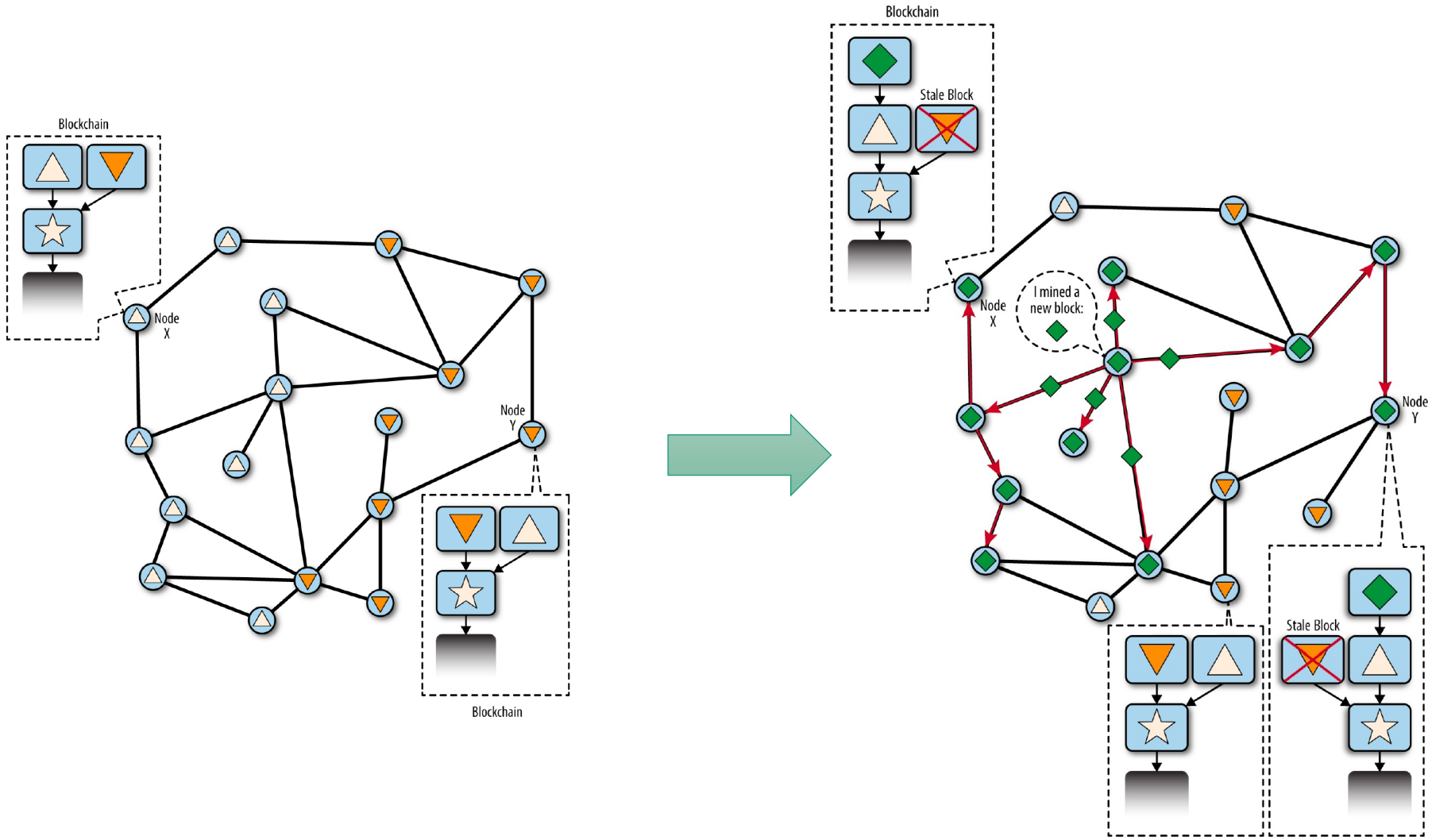
- Once block validated, attempt to assemble chain with most PoW
- Nodes maintain main blockchain, secondary blockchains and orphans
- Received blocks connected to the main/secondary chain
  - Recorverging to secondary chain if more cumulative work
- Consensus by selecting the greatest-comulative-work valid chain
  - “Voting” which chain is the main – extending the chain with a new block is the vote

Forks

- Different copies of blockchain not always consistent, e.g., delays

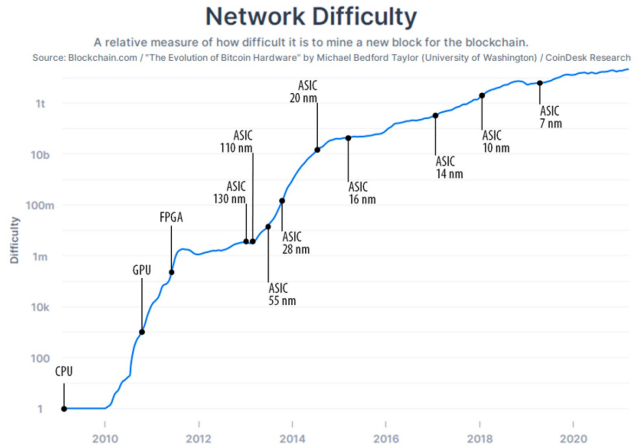


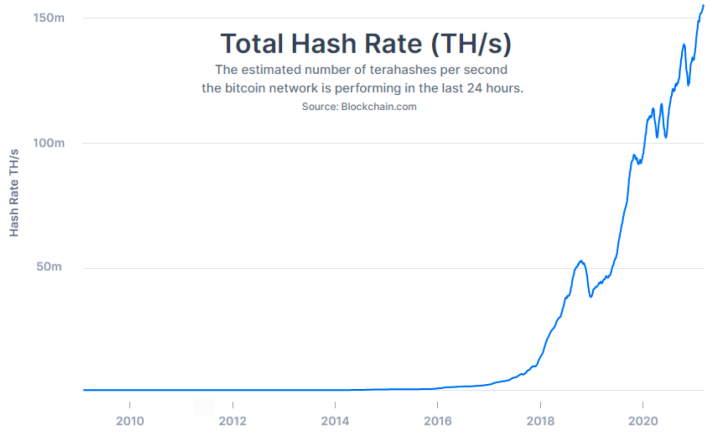
Fork Resolution



Mining Race

- Mining related to the cost of electricity
- The hashing power increased exponentionaly (Moore's Law)
  - CPU → GPU → FPGA → ASIC
  - The extra nounce solution





Mining Pools

- Solo miners have no chance any longer
  - E.g., $P = (14 * 10^{12} / 3 * 10^{18}) * 210{,}240 = 0.98$
    - HW with 14TH/s ~ $2,500 + $1.5/day for electricity
    - 3EH/s – network wide hashing rate
    - 210,240 – number of blocks in 4 years
- Managed pools
  - Pool operator
- P2Pool
  - No central operator
  - Blockchain like system, e.g., sharechain

Consensus Attacks

- Attack future or recent past (~ tens of blocks)
  - Large computation power needed for very deep forks
- Causes DoS, cannot steal bitcoins

51% Attack

- Majority of miners (51%) can cause deliberate forks
  - Theoretically ~30% of hashing power should be enough
- Double spend on attackers transactions
  - Wait >= 6 confirmations, or
  - Multisignature account with few confirmations
- DoS of addresses
  - Ignore specific addresses

Changing Consensus Rules

- Hard forks
  - Not forward compatible changes
  - Change of consensus rules (a bug, modification)
    - Software forks (Bitcoin XT, Bitcoin Classic, Bitcoin Unlimited)
  - Forcing minority to upgrade or stay on a minority chain
    - Potentially creating two competing systems
- Soft forks
  - Forward compatible change (non-upgraded clients continue to operates)
  - E.g., redefining NOP opcodes