Understanding Blockchain Technology

Introduction for Developers

Minho Shin

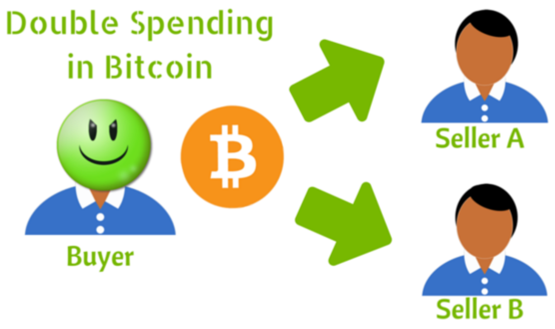
Myongji University

Agenda

- Shortest overview
- Satoshi’s whitepaper
- Deep dive
- Altcoins

Electronic Money Problem

- Electronic money
  - Keep track of who gave how much money to whom (called *Transactions*)
- Easy solution
  - A Trusted Third Party (TTP) can do the job (database)
  - This is what Credit companies do
- Can we trust the financial companies?
  - One organization manages all the cash in the world?
  - *We want a distributed solution*
- Why difficult?
  - Double spending problem is the key challenge
  - Bitcoin uses *consensus algorithm*



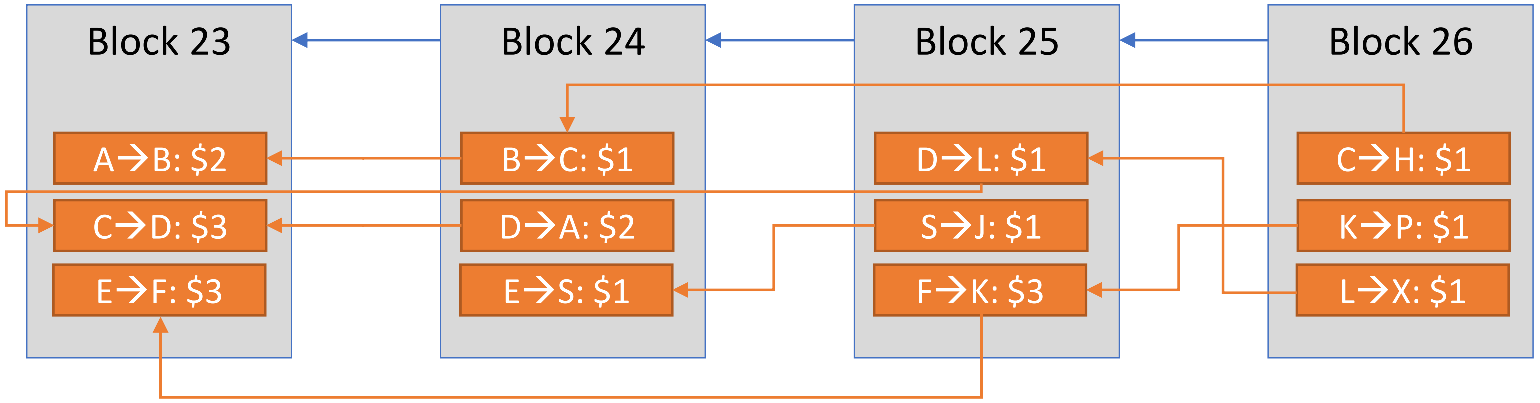
The shortest intro. To Bitcoin (Cryptocurrency)

IN TWO SLIDES

How Bitcoin works (without Why)

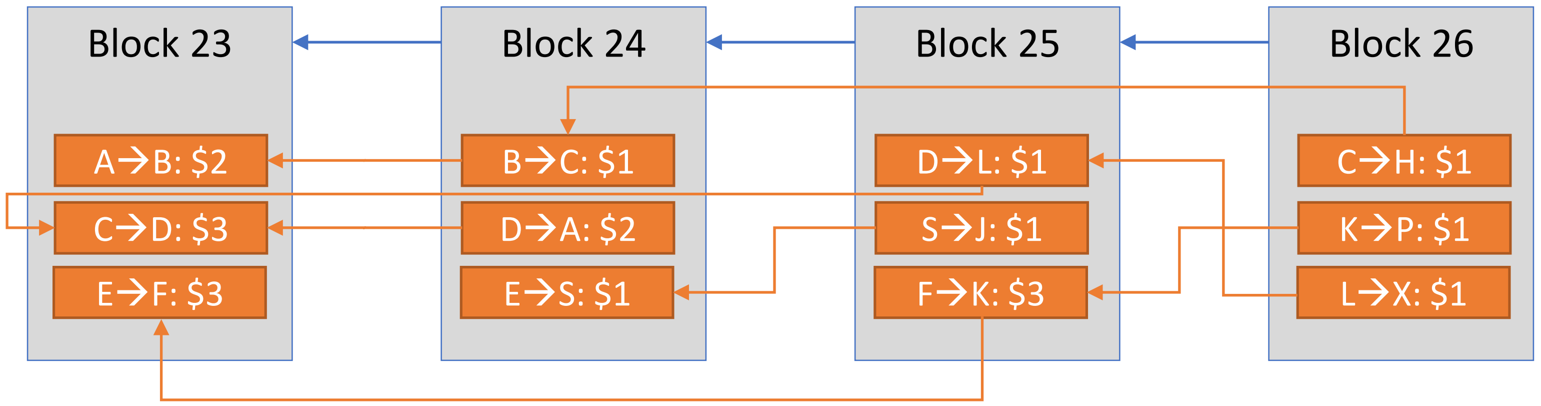
- Store cash flow in (multiple) tx-chains
- Store transactions in chained blocks
  - only one *universally-agreed* chain of blocks
- Hash-point →
  - hash of the previous transaction/block
  - Change in previous XX changes following XX
- Keep blocks in Peer-to-peer fashion

- Transactions are signed by the payer
- Users are identified by public key (or so)
- Blocks are added by miners
  - with great effort
- Miners checks double-spending
- Longest block-chain wins the consensus



Why Bitcoin works

- Public cryptography (ECCDSA)
  - Authenticity, Non-repudiation
- Cryptographic hash (SHA256/ RIPE256)
  - Integrity of transactions and blocks
- Consensus algorithm (Proof of Work)
  - Democratic truthfulness
  - No attacker can make the block chain of its own taste
    - Attackers are outnumbered (outcomputed) by others



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Nakamoto Satoshi's Whitepaper:

Bitcoin: A Peer-to-Peer Electronic Cash System

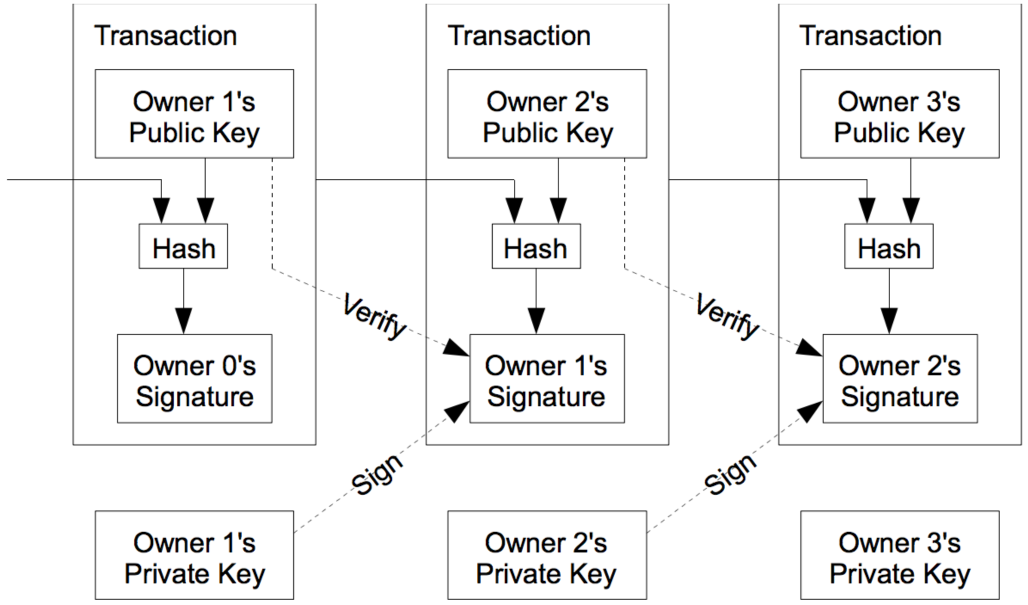
Crux of Bitcoin and Blockchain Technology

https://bitcoin.org/bitcoin.pdf

Introduction

- Limitations of TTP-based financial system
  - non-reversible transaction is not possible
  - mediation cost increases transaction cost
  - no means of payments through communication w/o TTP
- New electronic payment system is needed
  - based on cryptographic proof, not trust
  - computationally-impractically irreversible transactions
- Double-spending problem solved
  - P2P distributed timestamp server computationally proves the chronical order of transactions
  - Secure as long as honest nodes' CPU power collectively surpass attackers'

Transactions



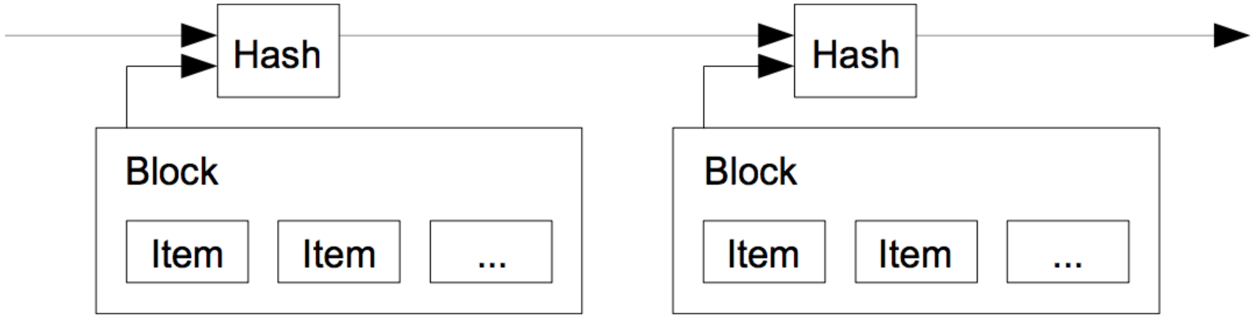
- Coin = a chain of digital signatures
- Tx_i: A transfers coin to B
  - Coin+= Pub_B, $H(Tx_{i-1}|Pub_B)$, $Sign_A(H(Tx_{i-1}|Pub_B))$
- One can
  - check if Tx_i's spender is TX_{i-1}'s recipient
- One cannot
  - check if TX_{i-1}'s recipient spent only once

Double Spending

- TTP (mint) model
  - TTP needs to check all transactions for double-spending
  - TTP issues new coin and only TTP-issued coin is trusted
- Distributed way
  - Payee checks if payer signed the TX for the first time
  - Payee needs to know all transactions, so be announced
  - All participants need to agree on a single ordered history of transactions

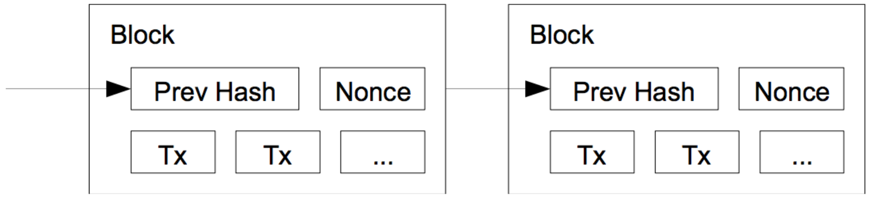
Timestamp Server

- Timestamp server publishes
  - H_t = Hash(H_{t-1}, $Block_t$)
    - $Block_t$=set of items exited at time t
- Publish to newspaper or bulletin boards
  - → Centralized timestamp server = TTP ?



Proof of Work

- Goal: Distributed Timestamp Server
- Proof-of-work
  - Finding a value that hashes to 0-bits-beginning value
    - Exponentially difficult by # of beginning 0-bits
  - Verified by one hashing
  - Hashcash
- Bitcoin PoW
  - Find Nonce s.t.
    - $H(Block_t) = \{0\}^n\{0,1\}^*$
      - where $Block_t := H(Block_{t-1})$ | Tx's | Nonce
- Bitcoin PoW solves
  - Cannot change a $block_t$ w/o redoing PoW for $Block_{>=t}$
  - Implements vote-per-CPU majority decision making
    - Longest chain wins
  - Moving target: n increase to keep avg # of blocks per hour = 6



Hashcash

- Originally proposed by Cynthia Dwork
- Hashcash proposed by Adam Back
- Sending an email costs a PoW
  - Find a <counter> s.t.
    - SHA-1(1:20:<time>:<recipient>:<rand>:<counter>)=$\{0\}^{20}\{0,1\}^{140}$
  - Takes about 1 sec
- Receiver accepts an email only if
  - <time> is recent(2-days), <recipient> is correct
  - and valid PoW
- This prevents a spammer
  - whose business relies on the ability to send many emails quickly

Network

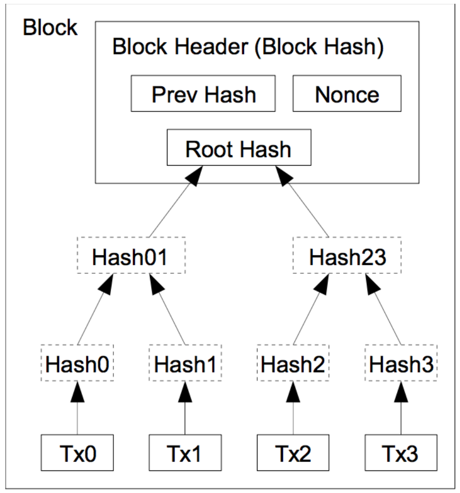
1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Incentive

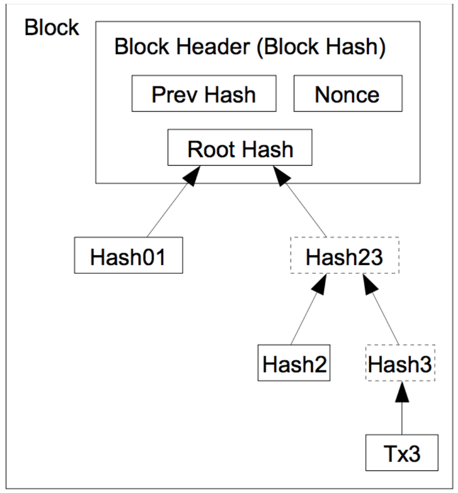
- Node can get incentives by either
  - First transaction in a block starts a new coin of the block creator
  - Transaction fee = input – output of a transaction
  - Once enough coins were created, only transaction fee is incentive
- Incentive encourages attackers (dominating CPU power) to play honest
  - because it is more profitable (generating blocks) than undermining the system

Reclaiming Disk Space

- Growing chains of blocks can use-up disks
- Old transactions buried in enough blocks can be discarded
  - However, it will change the hash of the block, causing a mass
- Use Merkle Tree to hash a set of transactions
  - A set of transactions (i.e., a branch) can be pruned without affecting the root hash of the tree



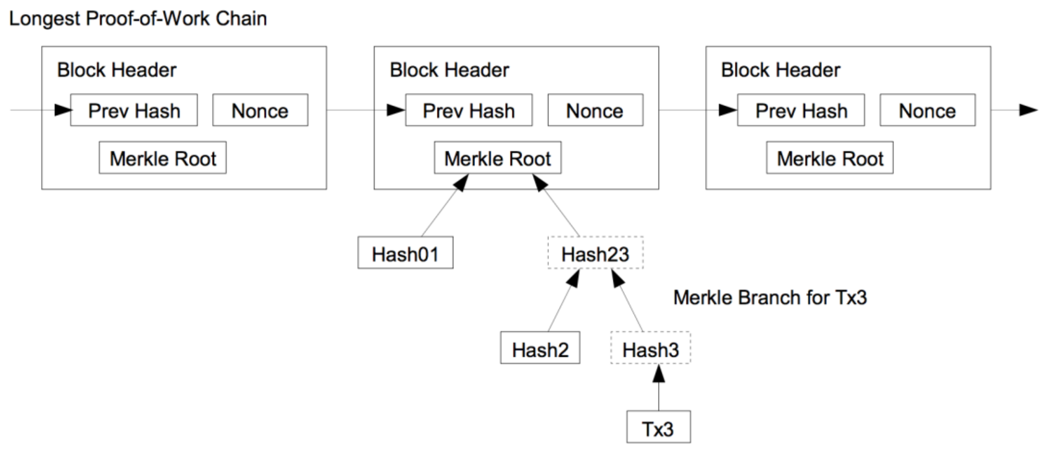
Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

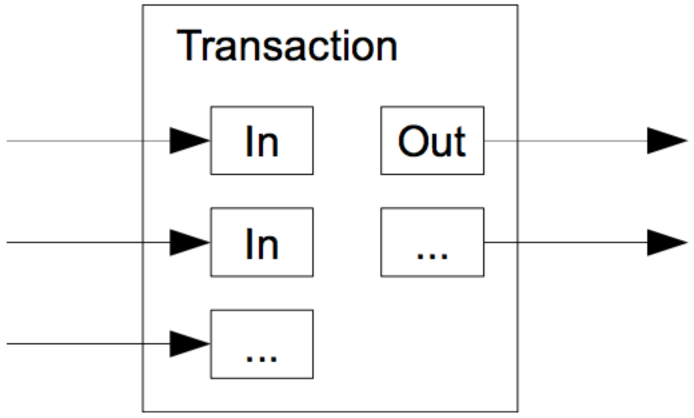
Simplified Payment Verification

- A node (miner) can verify a payment against double-spending
  - as it has all the blocks and its transactions of the longest chain
- A user (not miner) can perform simplified verification by
  - Keep only block headers of the longest chain
  - Get the MTree branch of the transaction to verify
  - Check if the transaction matches with the MTree
  - Meaning that the block creator verified the transaction, and the block was accepted in the longest chain



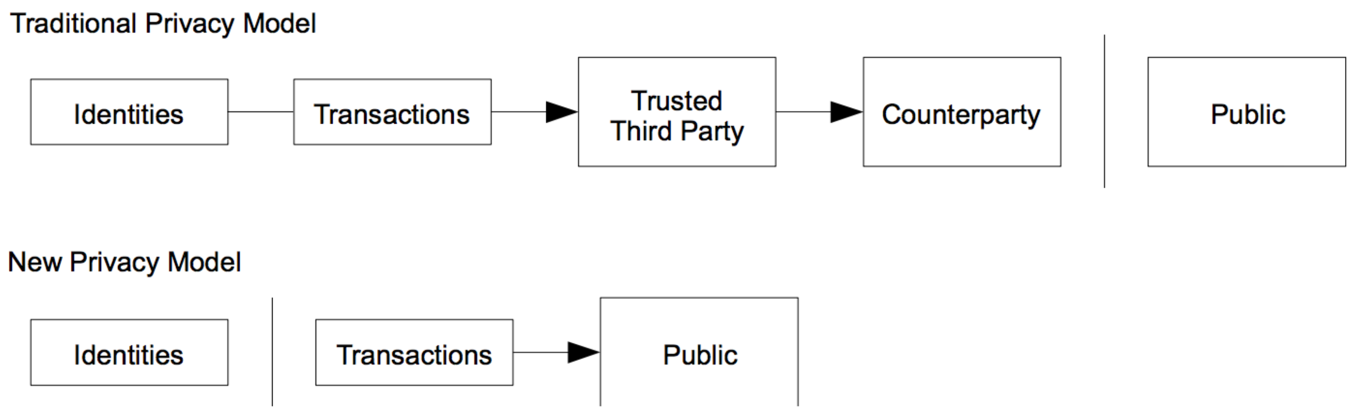
Combining and Splitting Value

- To allow combining/splitting values
  - transaction contains multiple inputs/outputs
- Outputs at most 2
  - One for payment
  - One for back to sender



Privacy

- TTP-model preserves privacy by access control
- Bitcoin privacy is preserved by
  - anonymity of public keys, generated per transaction
  - despite all transactions are announced publicly
- Some linking between transactions are unavoidable
  - when using multiple-inputs per transaction



Calculations

- What if attacker tries to generate an alternative chain faster than honest chain
    - Attacker can't forge other's transactions due to crypto
    - Attacker can only forge its own transactions to get money back
- Let X = Length(honest chain) - Length(alt-chain)
    - X: a binomial random walk
    - X++ if honest nodes find next block (w/ probability p)
    - X-- if attacker finds next block (w/ probability q)
    - p+q = 1
- Qz: probability that X reaches 0 starting from z
    - i.e., attacker catches up from z blocks behind
    - Then

$$q_z = \left\{ \begin{array}{cc} 1 & if\ p \leq q \\ (q/p)^z & if\ p > q \end{array} \right\}$$

Gambler's Ruin

- A gambler will eventually go broke if he bets a fraction of money for each game even if winning probability = 0.5
- Event Q_n
  - Starting with X=n, X reaches 0 (P(X--) = q)
- Event W
  - X decreases by 1 at the first game (attacker wins)
  - P(W) = q, P(~W) = p
- $P(Q_n) = P(Q_n|W)P(W)+P(Q_n|{\sim}W)P({\sim}W)$
  $= P(Q_{n-1})q + P(Q_{n+1})p$
- $P(Q_0) = 1$, $P(Q_{inf}) = 0$
- Solve a Linear Homogeneous Recurrence
  - $A_{n+1} = (1/p)A_n – (q/p)A_{n-1}$

Linear Homogeneous Recurrence

- $A_n = (1/p)A_{n-1} – (q/p)A_{n-2}$
- Characteristic equation
  - $r^2 – (1/p)r + (q/p) = 0$
  - Solution: q/p, 1
- $A_n = X(q/p)^n + Y(1)^n$
- If $p>q$, $A_0 = 1 = X + Y$, $A_{inf} = 0 = Y$, so $X=1$
  - thus $A_n = (q/p)^n$
- If $q>=p$, $1 = X + Y$, $A_{inf} = 0 = X (q/p)^{inf} + Y$
  - so $X=1/(1-(q/p)^{inf}) \rightarrow 0$, $Y \rightarrow 1$
  - thus $A_n = 1$

Double spending attack

- After a transaction that attacker spends money, attacker tries to branch the chain so that it can send the money back to himself
- What's the success probability of the attack when the recipient waits for z blocks before accepting the payment?
- During z blocks of honest chain, average # of blocks the attacker could generate (Poisson process)
  - (z/p) * q = (time for generating z honest blocks) * q
- Probability that attacker can still catch up after z blocks

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{Bmatrix} (q/p)^{(z-k)} & if\ k \le z \\ 1 & if\ k > z \end{Bmatrix}$$

Double Spending Attack

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{Bmatrix} (q/p)^{(z-k)} & if\, k \le z \\ 1 & if\, k > z \end{Bmatrix}$$

- T=z/p: average time for generating z honest blocks
- ƛ=zq/p: average # of attacker’s blocks generated during T
- k: # of attacker’s blocks generated during T
- event A: attacker’s block chain wins consensus
- event A_k: attacker generates k blocks during T
- $P(A) = P(A|A_0)P(A_0)+P(A|A_1)P(A_1)+ P(A|A_2)P(A_2)+ \ldots$
- $P(A_k) = ƛ^k e^{-ƛ}/k!$ (Poisson distribution)
- if k > z, $P(A|A_k) = 1$ (attacker wins during T)
- if k ≦z, attacker has to catch up z-k blocks after T and until forever
- done

Puzzles for Proof of Work

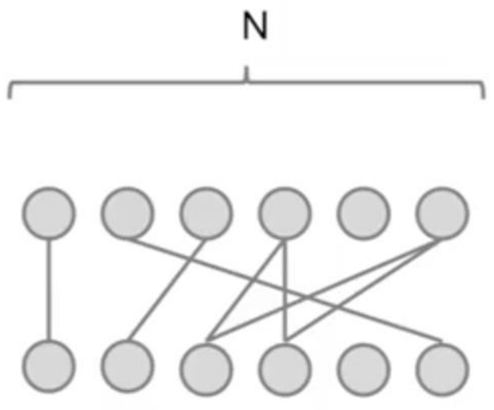
- Easy to verify
- Difficulty can be adjusted
- Chance of winning proportional to hash power
  - Sequential puzzle → Bad
  - Weighted sampling → Good
- Concerns about ASICs

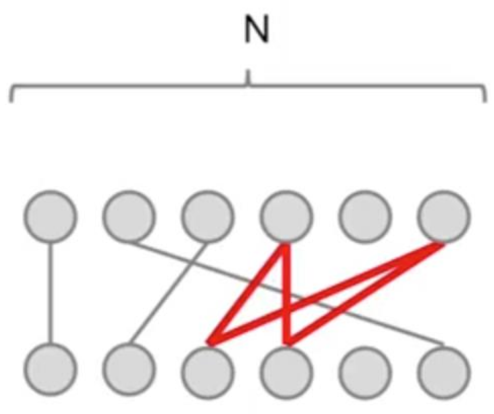
Scrypt

- Memory hard hash function
  - Constant time/memory tradeoff
- Used widely
  - Litecoin
- How it works
  - Fill memory with random values
  - Read from memory in random order
- Disadvantage
  - Verification is expensive
  - Not ASIC resistant

Cuckoo Hash Cycles

- Memory hard puzzle with cheap verification
- How it works
  - Compute a random bipartite graph based on X
  - Find a cycle of K size
  - Output X and K edges





Virtual Mining

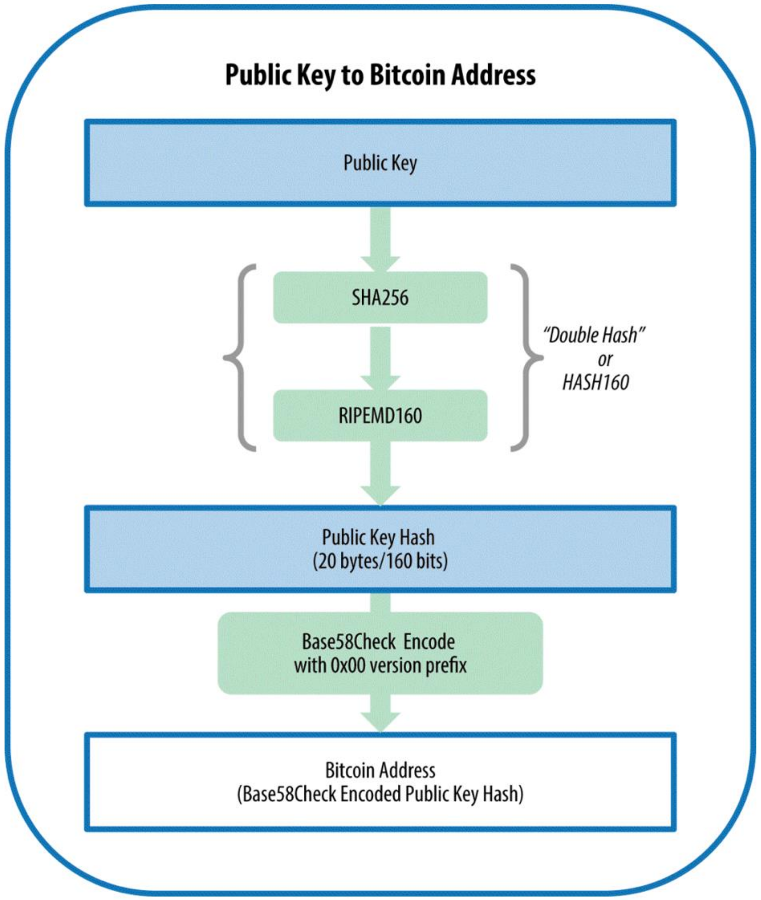
- No real resources are consumed for mining
  - Electric power, computing equipments
- Virtual Mining
  - Miners are chosen based on the contributed Bitcoins
  - Brining wealth outside Bitcoin into the system
  - 51% Attack is even harder

- 210,000 블록마다 화폐 발행량이 50% 줄어든다.

- Approaches
  - Proof-of-Stake: older coin gets higher stake
  - Proof-by-Burn: coins used for mining gets destroyed
  - Proof-by-Deposit: mining coins can be reclaimed later
  - Proof-of-Activity: any coin (if online) can win
- Any security gain by consuming real resources?
  - Unanswered yet

Keys and Addresses

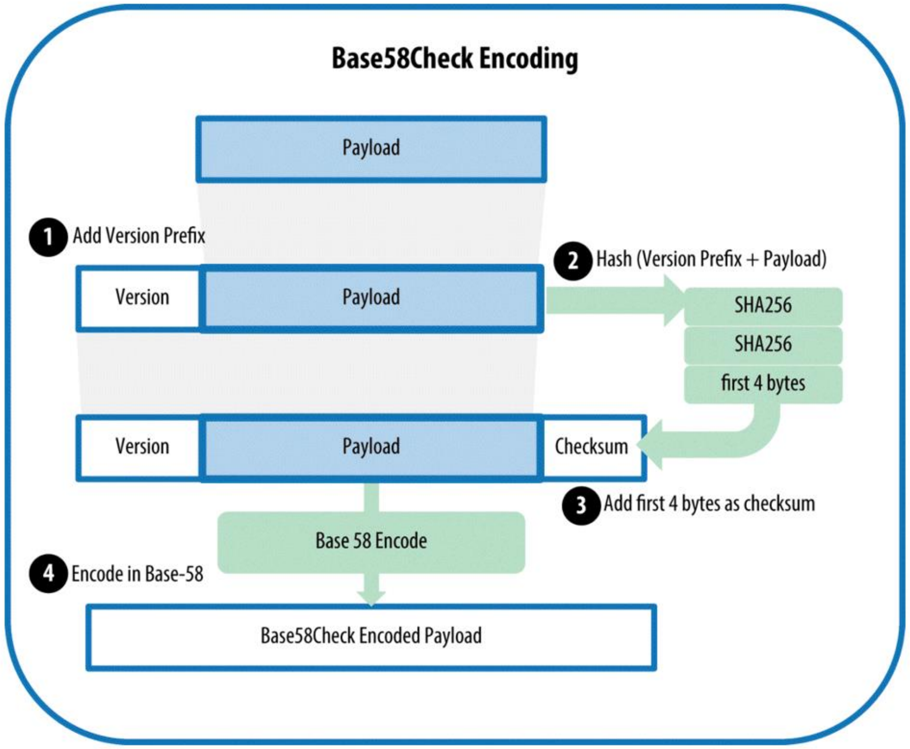
- Generate Public/Private key pair of Elliptic Curve cryptography
- Generate Bitcoin address from Public key

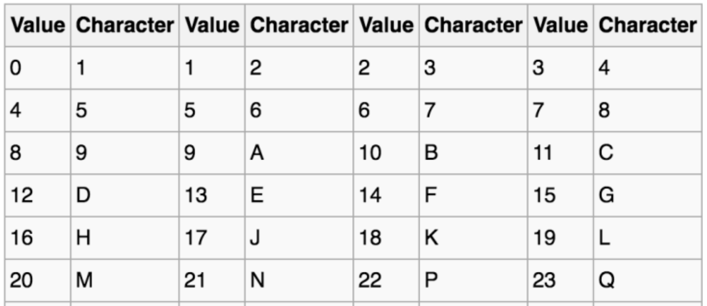


Public Key to Bitcoin Address
Public Key
SHA256
RIPEMD160
"Double Hash" or HASH160
Public Key Hash (20 bytes/160 bits)
Base58Check Encode with 0x00 version prefix
Bitcoin Address (Base58Check Encoded Public Key Hash)

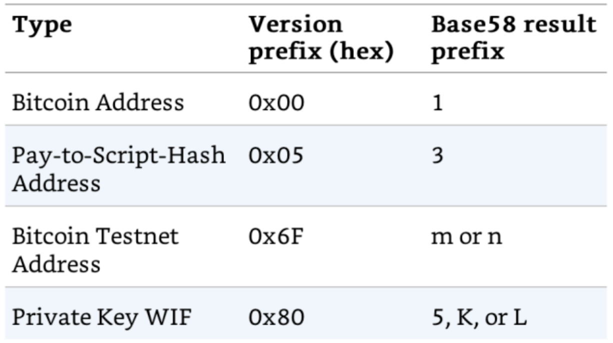
Base58Check Encoding

- Payload
  - Public key hash (20bytes)
- Base58
  - Encoding by {a..z, A..Z, 0..9} – {0, O, l, I}
- Base58Check
  - Add version & checksum
- Bitcoin address: 25 bytes
- Encoded address: 34 chars



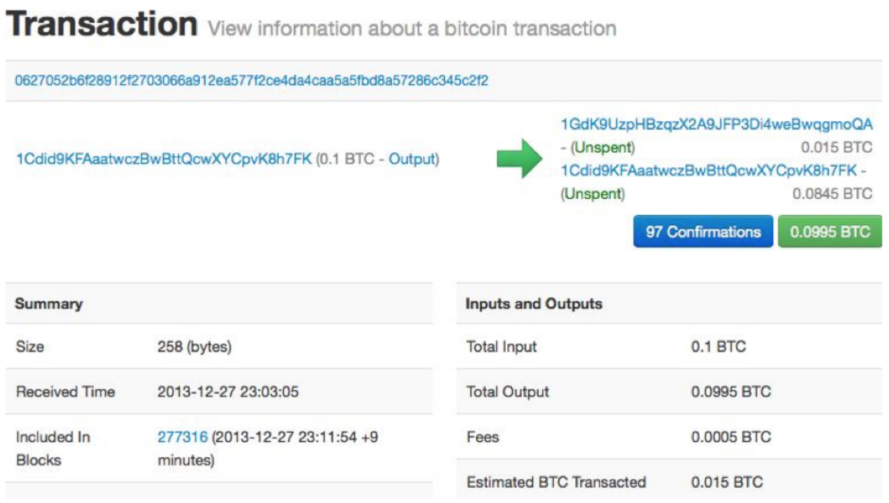


Value	Character	Value	Character	Value	Character	Value	Character
0	1	1	2	2	3	3	4
4	5	5	6	6	7	7	8
8	9	9	A	10	B	11	C
12	D	13	E	14	F	15	G
16	H	17	J	18	K	19	L
20	M	21	N	22	P	23	Q



Type	Version prefix (hex)	Base58 result prefix
Bitcoin Address	0x00	1
Pay-to-Script-Hash Address	0x05	3
Bitcoin Testnet Address	0x6F	m or n
Private Key WIF	0x80	5, K, or L

Transaction



Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA - (Unspent) 0.015 BTC
1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK - (Unspent) 0.0845 BTC

97 Confirmations 0.0995 BTC

Summary	
Size	258 (bytes)
Received Time	2013-12-27 23:03:05
Included In Blocks	277316 (2013-12-27 23:11:54 +9 minutes)

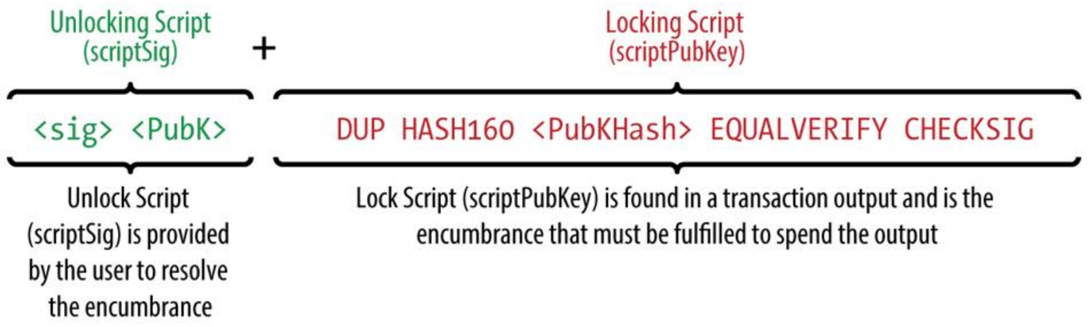
Inputs and Outputs	
Total Input	0.1 BTC
Total Output	0.0995 BTC
Fees	0.0005 BTC
Estimated BTC Transacted	0.015 BTC

- Fee = Input – Output
- Receiver’s pub key → script Pub key

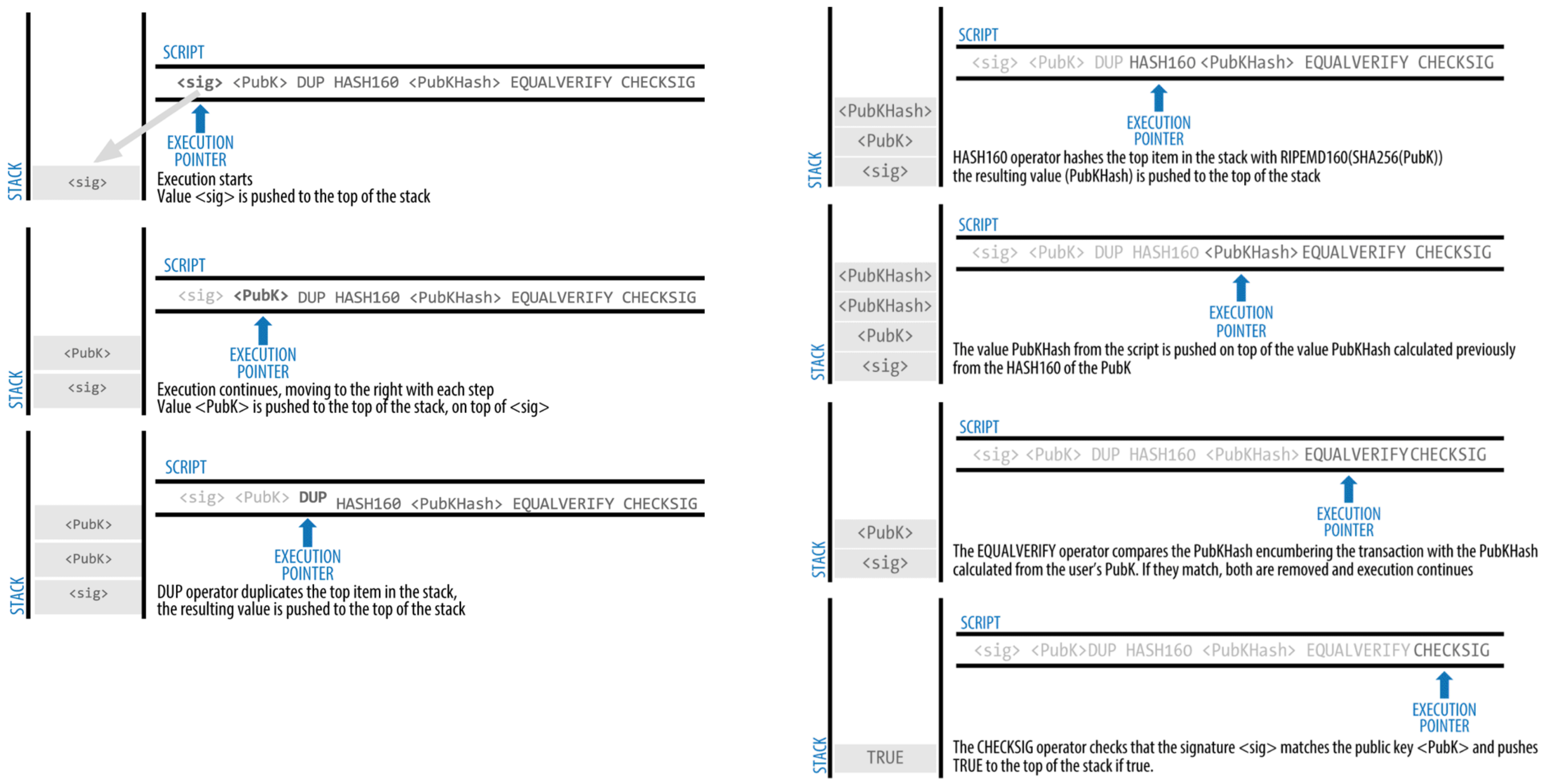
```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid":
"7957a35fe64f80d234d76d83a2a8f1a0d814
9a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" :
 "3045022100884d142d86652a3f47ba4746ec
 719bbfbd040a570b1deccbb6498c75c4ae24c
 b02204b9f039ff08df09cbe9f6addac960298
 cad530a863ea8f53982c09db8f6e3813[ALL]
 0484ecc0d46f1918b30928fa0e4ed99f16a0f
 b4fde0735e7ade8416ab9fe423cc541233637
 6789d172787ec3457eee41c04f4938de5cc17
 b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.01500000,
      "scriptPubKey": "OP_DUP
 OP_HASH160
 ab68025513c3dbd2f7b92a94e0581f5d50f65
 4e7 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": 0.08450000,
      "scriptPubKey": "OP_DUP
 OP_HASH160
 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc02
 5a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
  ]
}
```

Transaction Script

- Script
  - reverse-polish notation stack-based execution language
- P2PKH (Pay to Public Key Hash)


Unlocking Script (scriptSig)
+
Locking Script (scriptPubKey)
<sig> <PubK>
DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
Unlock Script (scriptSig) is provided by the user to resolve the encumbrance
Lock Script (scriptPubKey) is found in a transaction output and is the encumbrance that must be fulfilled to spend the output

Execution of {Unlock lock} script



Execution starts
Value <sig> is pushed to the top of the stack

Execution continues, moving to the right with each step
Value <PubK> is pushed to the top of the stack, on top of <sig>

DUP operator duplicates the top item in the stack,
the resulting value is pushed to the top of the stack

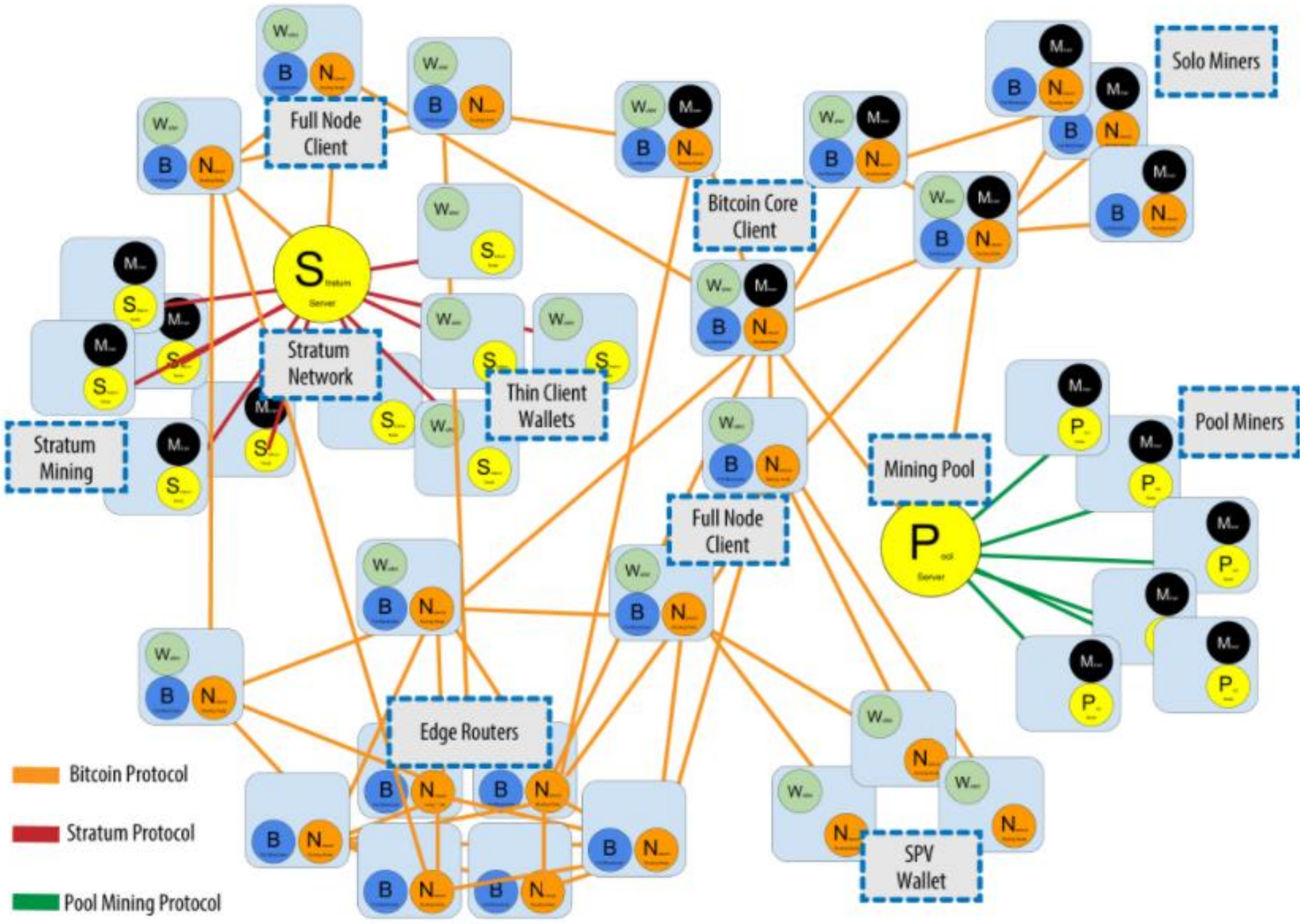
HASH160 operator hashes the top item in the stack with RIPEMD160(SHA256(PubK))
the resulting value (PubKHash) is pushed to the top of the stack

The value PubKHash from the script is pushed on top of the value PubKHash calculated previously
from the HASH160 of the PubK

The EQUALVERIFY operator compares the PubKHash encumbering the transaction with the PubKHash
calculated from the user's PubK. If they match, both are removed and execution continues

The CHECKSIG operator checks that the signature <sig> matches the public key <PubK> and pushes
TRUE to the top of the stack if true.

Bitcoin Network



Block header

Table 1. The structure of a block

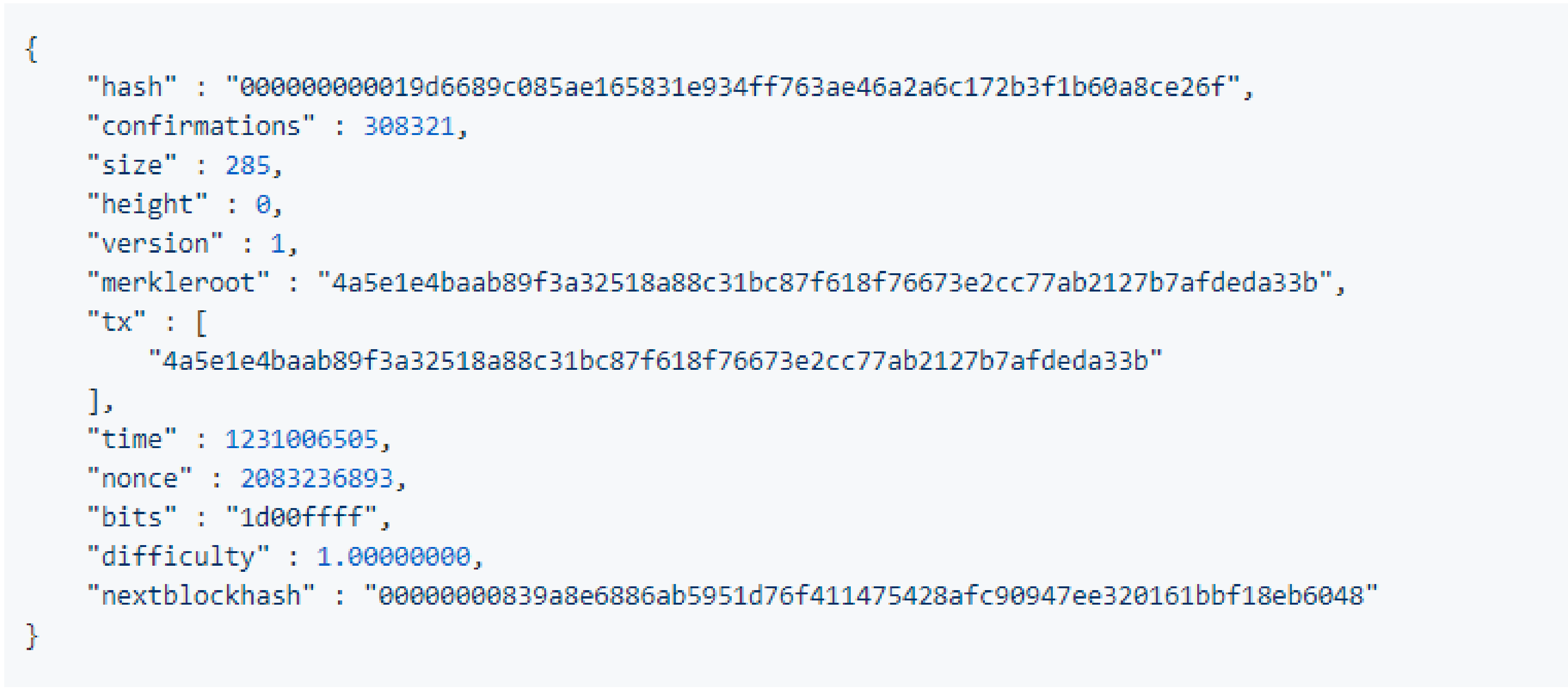
Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1–9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

Table 2. The structure of the block 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 (seconds from 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

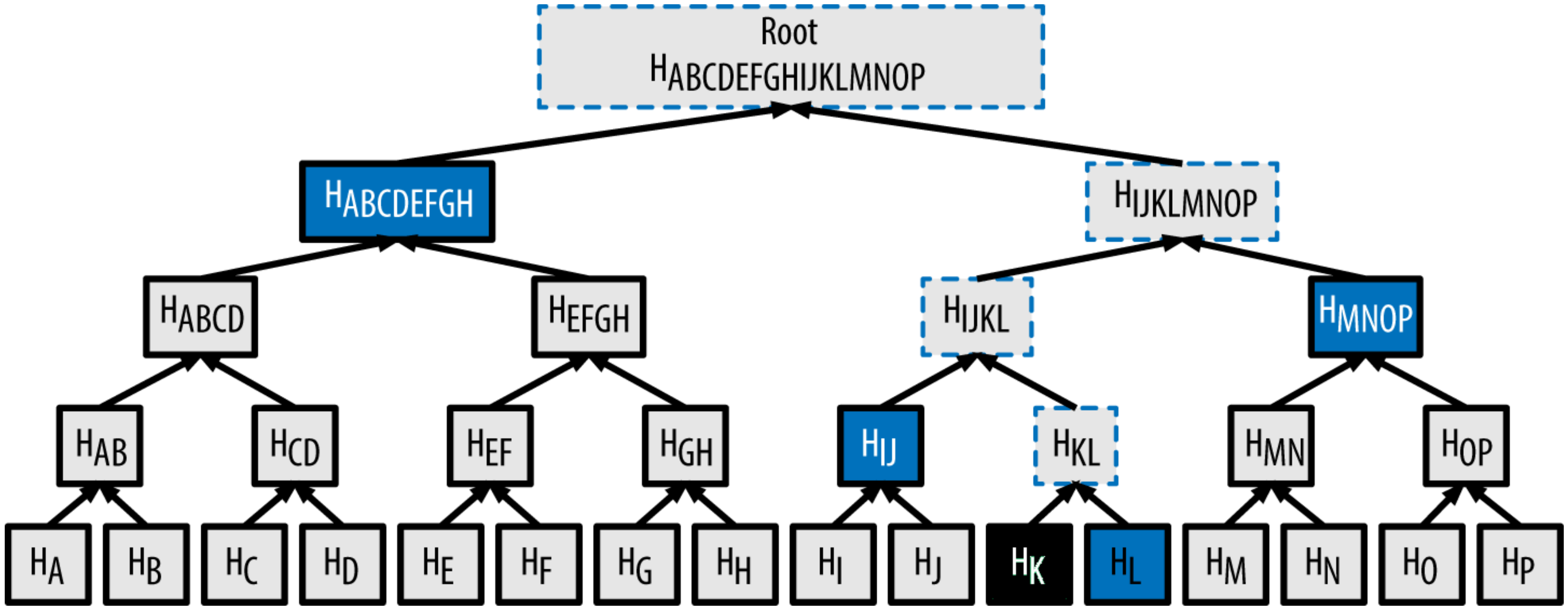
Genesis Block

- Transaction contains the text:
  - The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.



```
{
    "hash" : "000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
    "confirmations" : 308321,
    "size" : 285,
    "height" : 0,
    "version" : 1,
    "merkleroot" : "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
    "tx" : [
        "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
    ],
    "time" : 1231006505,
    "nonce" : 2083236893,
    "bits" : "1d00ffff",
    "difficulty" : 1.00000000,
    "nextblockhash" : "00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048"
}
```

Merkle Tree

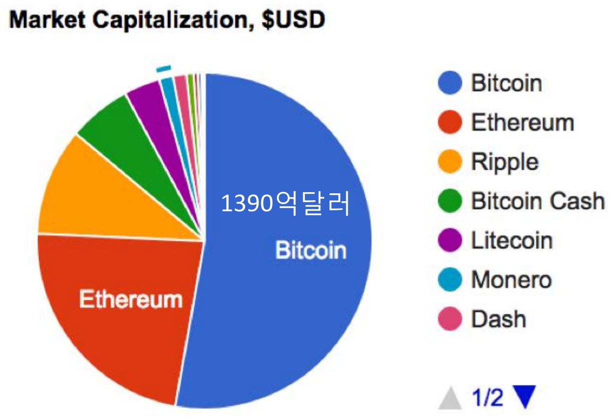

Root
$H_{ABCDEFGHIJKLMNOP}$
$H_{ABCDEFGH}$
$H_{IJKLMNOP}$
H_{ABCD}
H_{EFGH}
H_{IJKL}
H_{MNOP}
H_{AB}
H_{CD}
H_{EF}
H_{GH}
H_{IJ}
H_{KL}
H_{MN}
H_{OP}
H_A
H_B
H_C
H_D
H_E
H_F
H_G
H_H
H_I
H_J
H_K
H_L
H_M
H_N
H_O
H_P

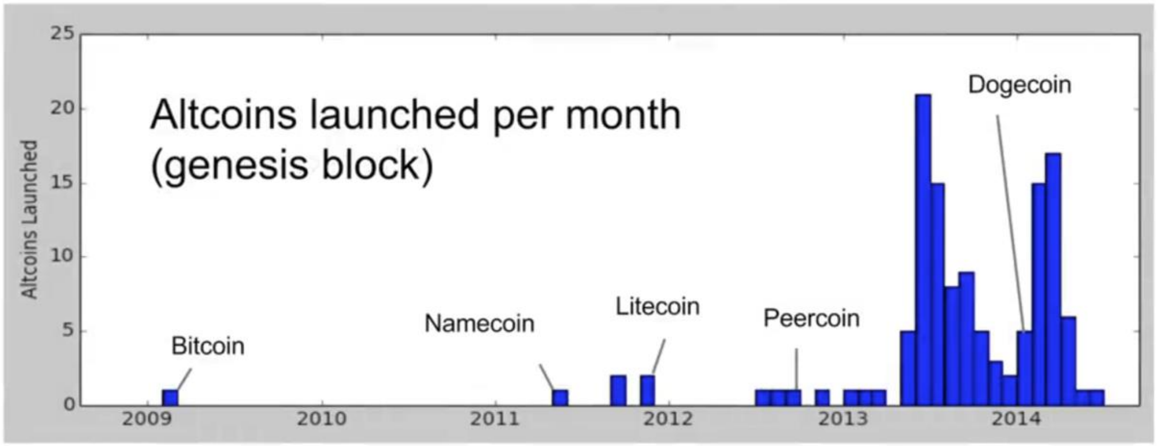
Miner’s incentives

- Mining reward는 4년마다 절반으로 감소되고 있다.
  - 210,000 블록마다 화폐 발행량이 50% 줄어든다.
  - 50(2009 / 2) -> 25(2012 / 11) -> 12.5(2016 / 7)
  - 2140년까지 기하급수적으로 줄어든다.
  - 2140년 이후에는 새로운 비트코인은 발행되지 않는다.
  - 2140년에는 2,099,999,997,690,000 사토시, 2100만 비트코인이 모두 발행됨.
  - 2140년 이후에는 모든 수익이 Transaction fee에서 발생한다.
- Fee는 비트코인 소득의 0.5% 또는 그 이하로 나타난다

Altcoin

- Bitcoin Jan, 2009
- One altcoin launched every month (now ~500)





Altcoins

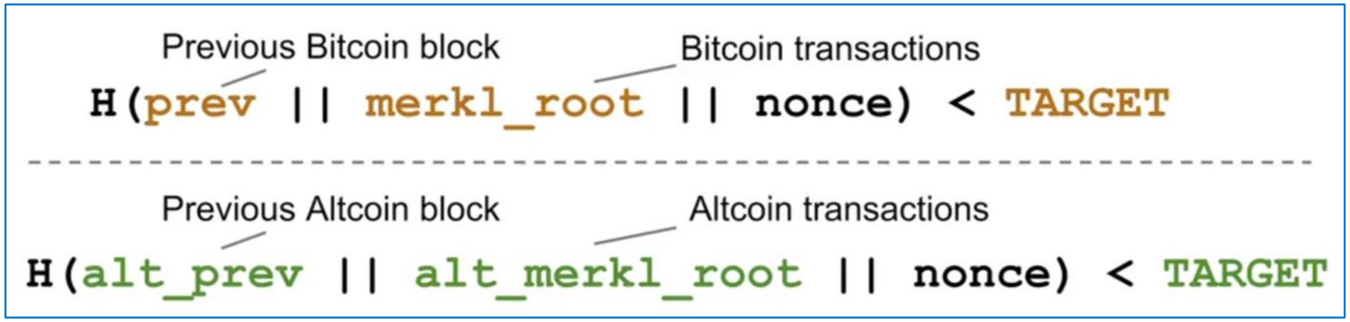
- Features
  - Better/Different security
    - Mining puzzles
  - Contract/platform features
  - Different parameters
  - Community supports
- Namecoin (2011)
  - replaces Domain Name Registration
- Litecoin (2011)
  - Memory-hard puzzle
  - Block-rate 2x faster
- Peercoin (2012)
  - First PoS

Mining Attacks

- Miner (Mining pool) on a large network can demolish small altcoin
  - Jan 2012, CoiledCoin by Eligius pool
  - Jul 2013, TerraCoin
  - Nov 2013, WorldCoin

Merge Mining

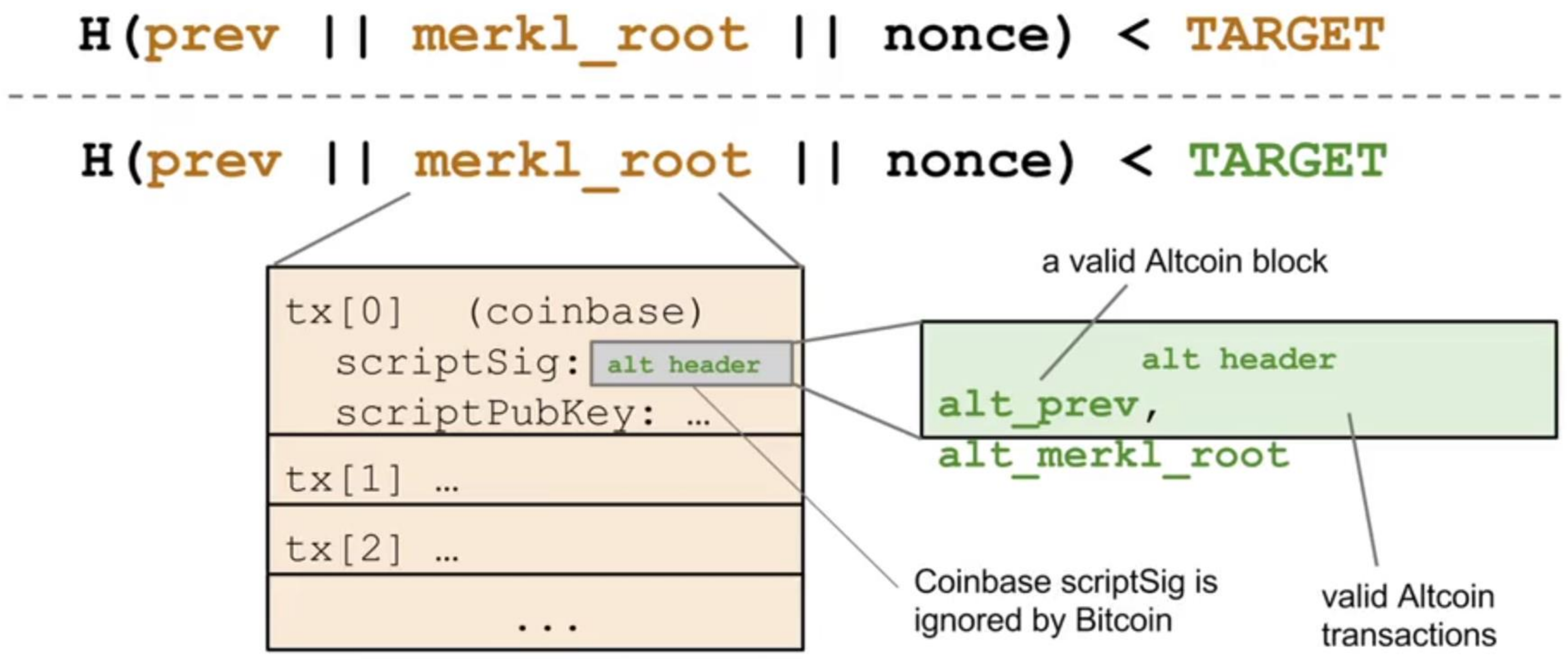
- Mining is exclusive between coins



```
H(prev || merkl_root || nonce) < TARGET
-----------------------------------------------------------
H(alt_prev || alt_merkl_root || nonce) < TARGET
```

- Merge Mining
  - combine mining efforts of two coins
  - Mining in an altcoin is a valid mining attempt in Bitcoin
  - Embed the block header of altcoin into the block header of Bitcoin

Merge Mining: How it works


H(prev || merkl_root || nonce) < TARGET
H(prev || merkl_root || nonce) < TARGET
tx[0] (coinbase)
scriptSig: alt header
scriptPubKey: …
tx[1] …
tx[2] …
. . .
a valid Altcoin block
alt header
alt_prev,
alt_merkl_root
Coinbase scriptSig is ignored by Bitcoin
valid Altcoin transactions