Bitcoin

Presented by Jiaxiao Zhou (周佳孝 Mossaka)

Microsoft

What is Bitcoin

Bitcoin: A Peer-to-Peer Electronic Cash System

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

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

A distributed, decentralized digital currency system

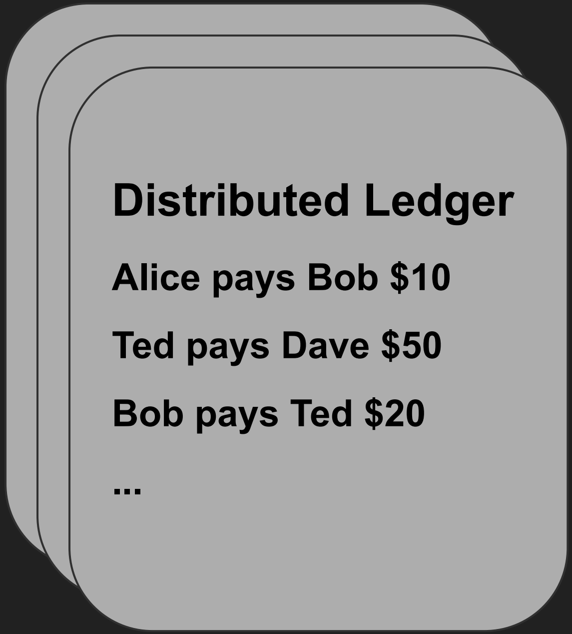
- Like Raft, it’s a state machine, uses Log and consensus algorithm to agree on log content
- Unlike Raft, participants are byzantine.

What are byzantine participants?

- There could be arbitrary number of nodes participating.
- Nodes can be malicious 😈

The core problem: How do you ensure that a peer-to-peer, distributed network with no central authority can make correct decisions, even if some of the nodes tell lie?

Let’s break this down



There are a bunch of coins, each own by someone

Let’s break this down

Distributed Ledger

Alice pays Bob $10

Ted pays Dave $50

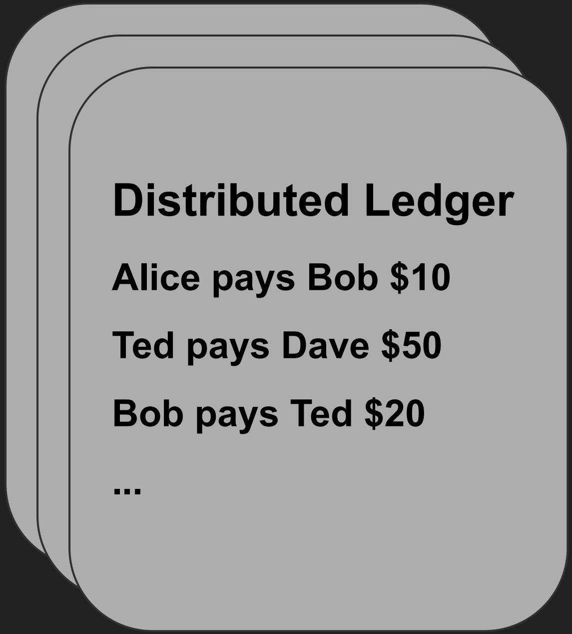
Bob pays Ted $20

...

There are a bunch of coins, each own by someone

Coin = sequence of transaction records

Let’s break this down



There are a bunch of coins, each own by someone

Coin = sequence of transaction records

How can we prevent Bob from saying “Alice pays Bob $1000”?

Solution: Digital Signature

Distributed Ledger

Alice pays Bob $10

Ted pays Dave $50

Bob pays Ted $20

...

Transaction "Alice pays Bob $10"

1. Bob's PK
2. Hash(previous transaction)
3. Sign with Alice's SK
4. Amount
5. ins/outs

Transaction Example

T6: pk(Bob), …

T7: pk(Alice), hash(T6), sig(Bob), 1 BTC

Alice owns a 1 BTC given by Bob

Alice buys a car from Dave and pays with this coin

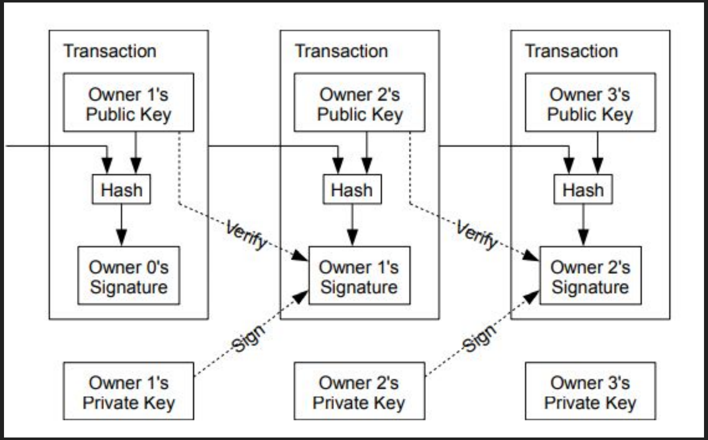
1. Dave sends pk to Alice
2. Alice creates a new transaction and signs it with sk(Alice)

T8: pk(Dave), hash(T7), sig(Alice), 1 BTC

Dave can verify T8 using pk(Alice) and gives the car to Alice

Transaction log = Currency

Some nice properties



1. Anyone in the world can use Alice’s PK to verify this transaction.
2. Invalid transaction will be rejected.
3. The hash of previous transaction is used as a unique identification

What if Dave knows Alice's private key?

Well, then Dave can use Alice's private key to sign any transactions.

Meaning, Alice lost all the coins. 😢

This a serious problem, and Bitcoin didn't solve it!

Everyone keeps a copy of the ledger

This open ledger idea solves man-in-the-middle problem as we saw in the last meetup.

Digital Signature solves outright forgery issue

However, this creates a new problem: Double Spending

Double Spending

Alice creates 2 transactions:

“pk(Dave), hash(Tx), sign(Alice), amount=10 BTC”

“pk(Ted), hash(Tx), sign(Alice), amount=10 BTC”

Alice shows different transactions to Dave and Ted

Double Spending

Alice creates 2 transactions:

“pk(Dave), hash(Tx), sign(Alice), amount=10 BTC”

“pk(Ted), hash(Tx), sign(Alice), amount=10 BTC”

Alice shows different transactions to Dave and Ted

Dave and Ted didn’t know complete set and order of transactions.

Double Spending

Alice creates 2 transactions:

“pk(Dave), hash(Tx), sign(Alice), amount=10 BTC”

“pk(Ted), hash(Tx), sign(Alice), amount=10 BTC”

Alice shows different transactions to Dave and Ted

Dave and Ted didn’t know complete set and order of transactions.

Now both Dave and Ted gave cars to Alice. 😁

The Chronologically Ordering is the key

First, ledger should be public, and every sees the same order! This is a consensus algorithm

The Chronologically Ordering is the key

First, ledger should be public, and every sees the same order! This is a consensus algorithm

Raft uses the majority rule.

The Chronologically Ordering is the key

First, ledger should be public, and every sees the same order! This is a consensus algorithm

Raft uses the majority rule.

But it won’t work in byzentine problem

1. Raft works because it knows exactly how many nodes are in the system

The Chronologically Ordering is the key

First, ledger should be public, and every sees the same order! This is a consensus algorithm

Raft uses the majority rule.

But it won’t work in byzentine problem

1. Raft works because it knows exactly how many nodes are in the system
2. Raft knows that all the nodes are honest

The Chronologically Ordering is the key

First, ledger should be public, and every sees the same order! This is a consensus algorithm

Raft uses the majority rule.

But it won’t work in byzentine problem

1. Raft works because it knows exactly how many nodes are in the system
2. Raft knows that all the nodes are honest

Nakamoto Consensus Protocol (PoW)

Proof of Work is a computational puzzle. Winner in PoW decide the next transaction log entry.

Some requirements:

Nakamoto Consensus Protocol (PoW)

Proof of Work is a computational puzzle. Winner in PoW decide the next transaction log entry.

Some requirements:

1. Solving PoW problem is hard. 1 CPU takes 1 month
2. Verify PoW solution is easy.

Crypto Hash Functions

A function that takes any sized message and output a fixed size bit array

a. deterministic,
b. quick to compute,
c. infeasible to reverse (birthday attach is the best solution)
d. infeasible to find a collision,
e. and avalanche effect

SHA1

Alg $\text{SHA1}(M)$ // $|M| < 2^{64}$

$V \leftarrow \text{SHF1}(\texttt{5A827999} \| \texttt{6ED9EBA1} \| \texttt{8F1BBCDC} \| \texttt{CA62C1D6}, M)$

return V

Alg $\text{SHF1}(K, M)$ // $|K| = 128$ and $|M| < 2^{64}$

$y \leftarrow \text{shapad}(M)$

Parse y as $M_1 \| M_2 \| \cdots \| M_n$ where $|M_i| = 512$ $(1 \leq i \leq n)$

$V \leftarrow \texttt{67452301} \| \texttt{EFCDAB89} \| \texttt{98BADCFE} \| \texttt{10325476} \| \texttt{C3D2E1}$

for $i = 1, \ldots, n$ do $V \leftarrow \text{shf1}(K, M_i \| V)$

return V

Alg $\text{shapad}(M)$ // $|M| < 2^{64}$

$d \leftarrow (447 - |M|) \bmod 512$

Let ℓ be the 64-bit binary representation of $|M|$

$y \leftarrow M \| 1 \| 0^d \| \ell$ // $|y|$ is a multiple of 512

return y

shf1

Alg $\text{shf1}(K, B \| V)$ // $|K| = 128$, $|B| = 512$ and $|V| = 160$

Parse B as $W_0 \| W_1 \| \cdots \| W_{15}$ where $|W_i| = 32$ $(0 \leq i \leq 15)$

Parse V as $V_0 \| V_1 \| \cdots \| V_4$ where $|V_i| = 32$ $(0 \leq i \leq 4)$

Parse K as $K_0 \| K_1 \| K_2 \| K_3$ where $|K_i| = 32$ $(0 \leq i \leq 3)$

for $t = 16$ to 79 do $W_t \leftarrow \text{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$

$A \leftarrow V_0$; $B \leftarrow V_1$; $C \leftarrow V_2$; $D \leftarrow V_3$; $E \leftarrow V_4$

for $t = 0$ to 19 do $L_t \leftarrow K_0$; $L_{t+20} \leftarrow K_1$; $L_{t+40} \leftarrow K_2$; $L_{t+60} \leftarrow K_3$

for $t = 0$ to 79 do

  if $(0 \leq t \leq 19)$ then $f \leftarrow (B \wedge C) \vee ((\neg B) \wedge D)$

  if $(20 \leq t \leq 39 \text{ OR } 60 \leq t \leq 79)$ then $f \leftarrow B \oplus C \oplus D$

  if $(40 \leq t \leq 59)$ then $f \leftarrow (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$

  $temp \leftarrow \text{ROTL}^5(A) + f + E + W_t + L_t$

  $E \leftarrow D$; $D \leftarrow C$; $C \leftarrow \text{ROTL}^{30}(B)$; $B \leftarrow A$; $A \leftarrow temp$

$V_0 \leftarrow V_0 + A$; $V_1 \leftarrow V_1 + B$; $V_2 \leftarrow V_2 + C$; $V_3 \leftarrow V_3 + D$; $V_4 \leftarrow V_4 + E$

$V \leftarrow V_0 \| V_1 \| V_2 \| V_3 \| V_4$; return V

The Block chain

We want every server to agree on a ordering of transaction log to prevent double-spending.

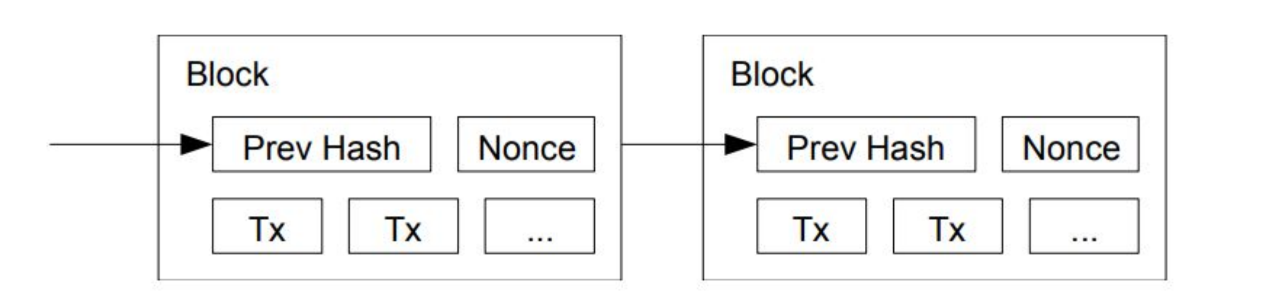
We break the ledger to blocks.

Each block has

1. Hash of previous block
2. Set of transactions
3. Nonce
4. Timestamp

The Block chain

- A transaction is included in a block means that someone has done enough work to prove the ordering of this transaction.
- New block generated every 10 minutes.
- Payee doesn't accept transaction until it's in the block chain.



Bitcoin Network

- There are some servers (miners) trying to solve the PoW puzzle: hash(block) has N leading zeroes.

Bitcoin Network

- There are some servers (miners) trying to solve the PoW puzzle: hash(block) has N leading zeroes.
- As we know, the best way to revert SHA 256 is to guess and check… so miners try all possible nonce to get solve the puzzle.

Bitcoin Network

- There are some servers (miners) trying to solve the PoW puzzle: hash(block) has N leading zeroes.
- As we know, the best way to revert SHA 256 is to guess and check… so miners try all possible nonce to get solve the puzzle.
- The winner broadcast the new block and to all peers.

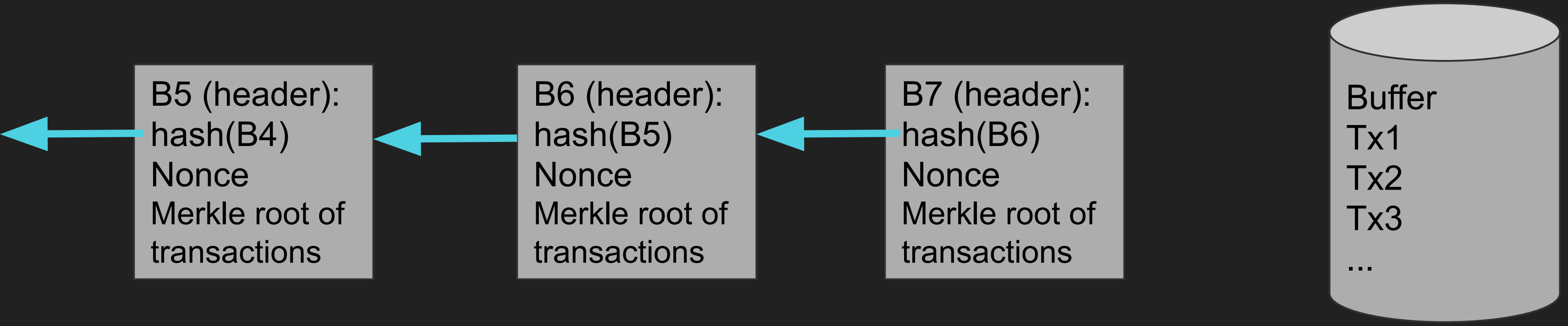
Bitcoin Network

- There are some servers (miners) trying to solve the PoW puzzle: hash(block) has N leading zeroes.
- As we know, the best way to revert SHA 256 is to guess and check… so miners try all possible nonce to get solve the puzzle.
- The winner broadcast the new block and to all peers.
- Nodes accept the block only if all transactions in it are valid and not already spent.

Bitcoin Network

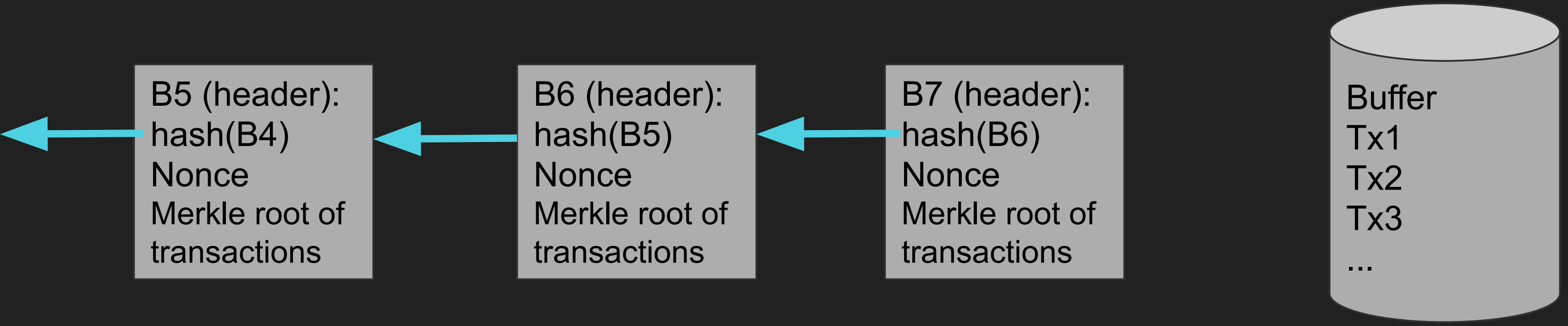
- There are some servers (miners) trying to solve the PoW puzzle: hash(block) has N leading zeroes.
- As we know, the best way to revert SHA 256 is to guess and check… so miners try all possible nonce to get solve the puzzle.
- The winner broadcast the new block and to all peers.
- Nodes accept the block only if all transactions in it are valid and not already spent.
- Nodes express their acceptance by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Example (miner)



The chain is shared by all peers. They are all mining block B6

Example (miner)



The chain is shared by all peers. They are all mining block B6

Now "Alice pays Dave 1 BTC, with hash(prev T)" broadcast to all miners. Miner saves the transaction in the buffer. Until B8 is computed, then miner include this transaction in B9.

So B5 - B6 - B7 - B8 - B9, eventually has this transaction

What could go wrong?

1. Two peers solves the puzzle at the same time
2. Network is slow, 2nd block found before 1st is known

What could go wrong?

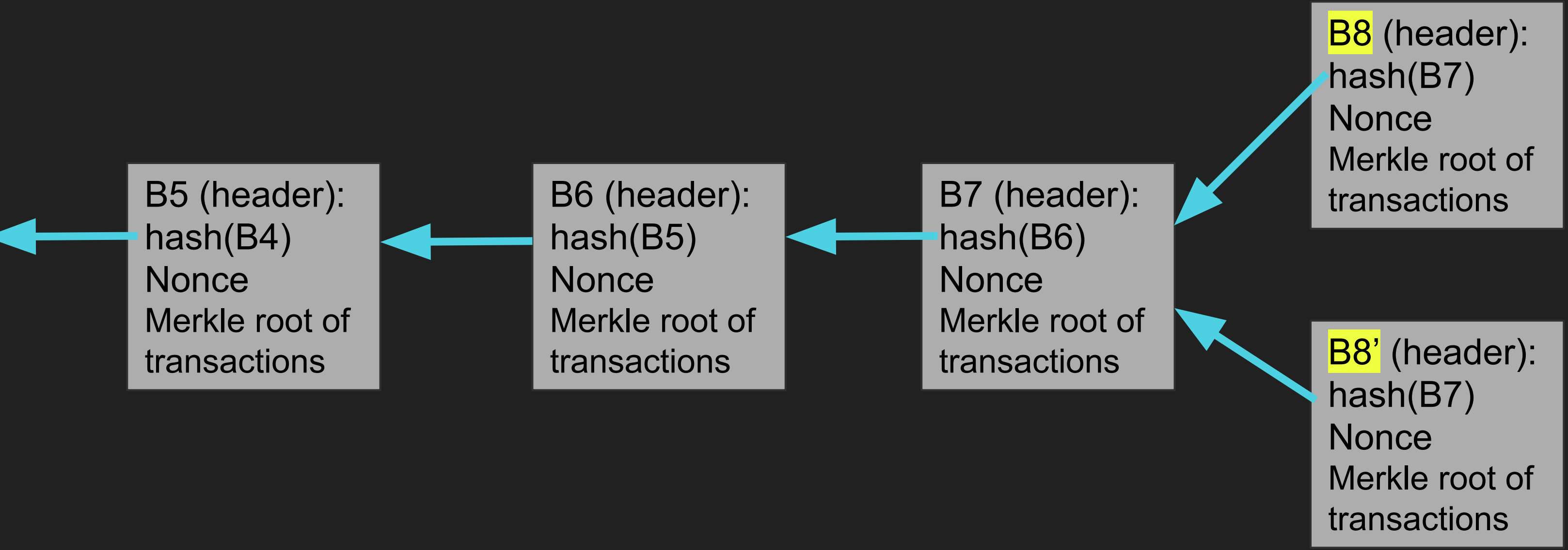
1. Two peers solves the puzzle at the same time
2. Network is slow, 2nd block found before 1st is known

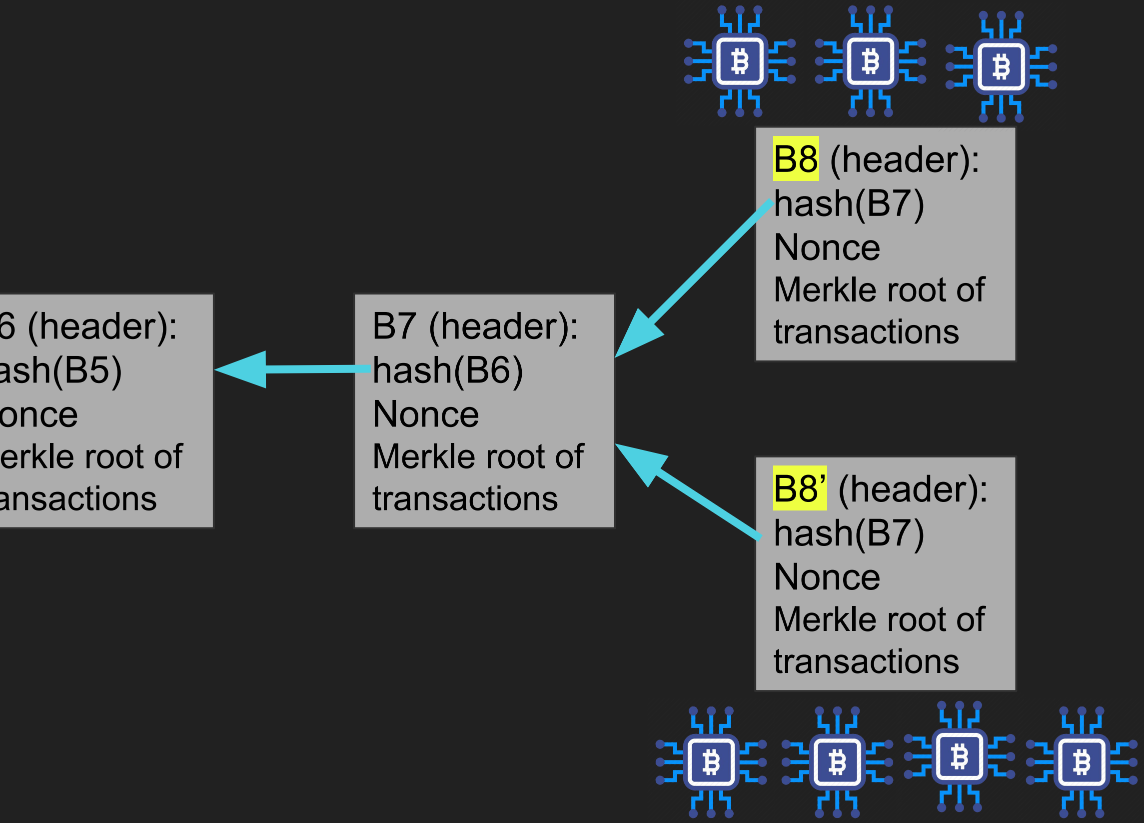
Solution, define a simple rule:

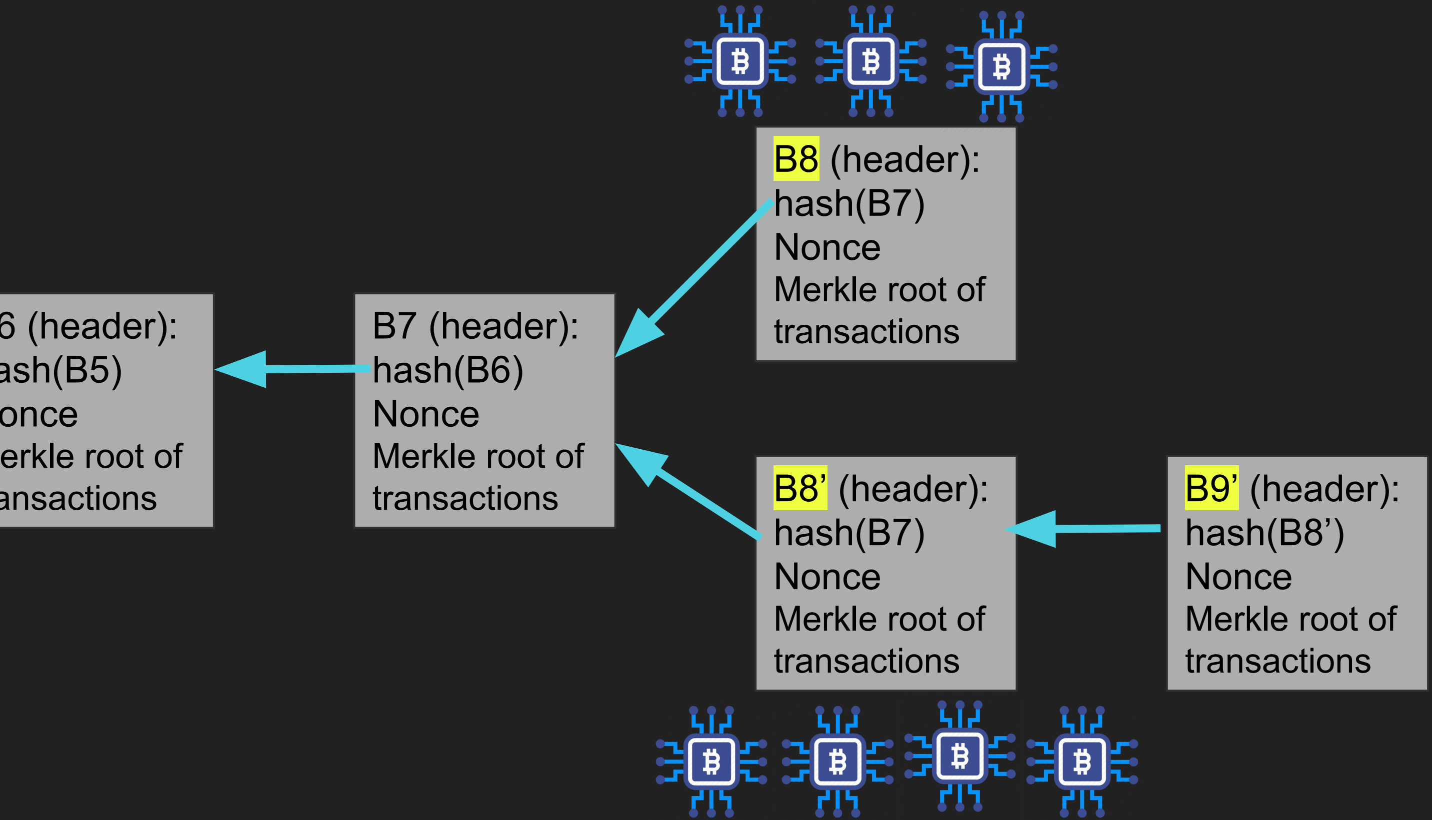
If the chain forks, peers work on whichever block they heard first and save the forked chain, but switch to longer chain if they become aware of one.

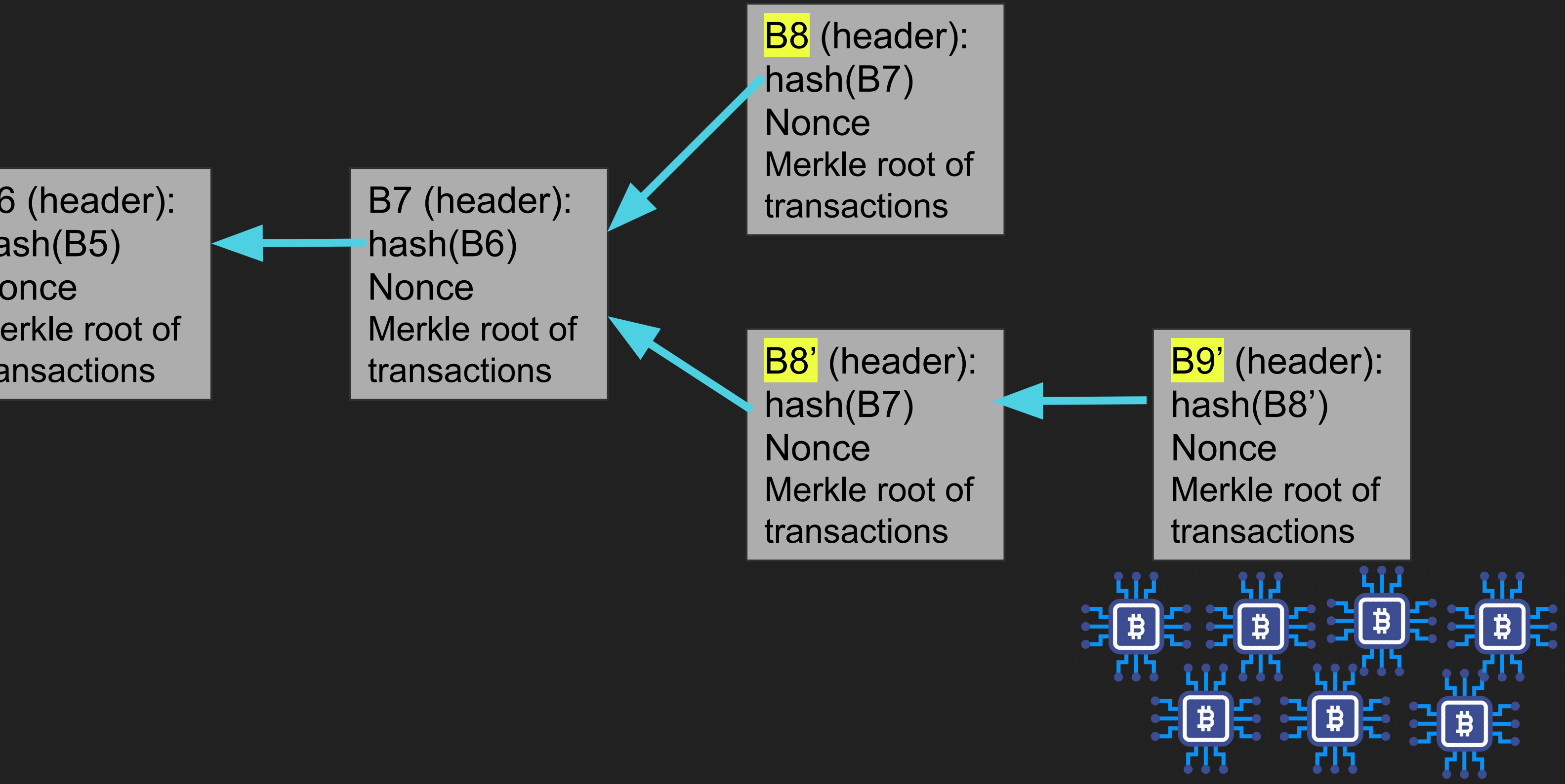
How is double-spending solved?

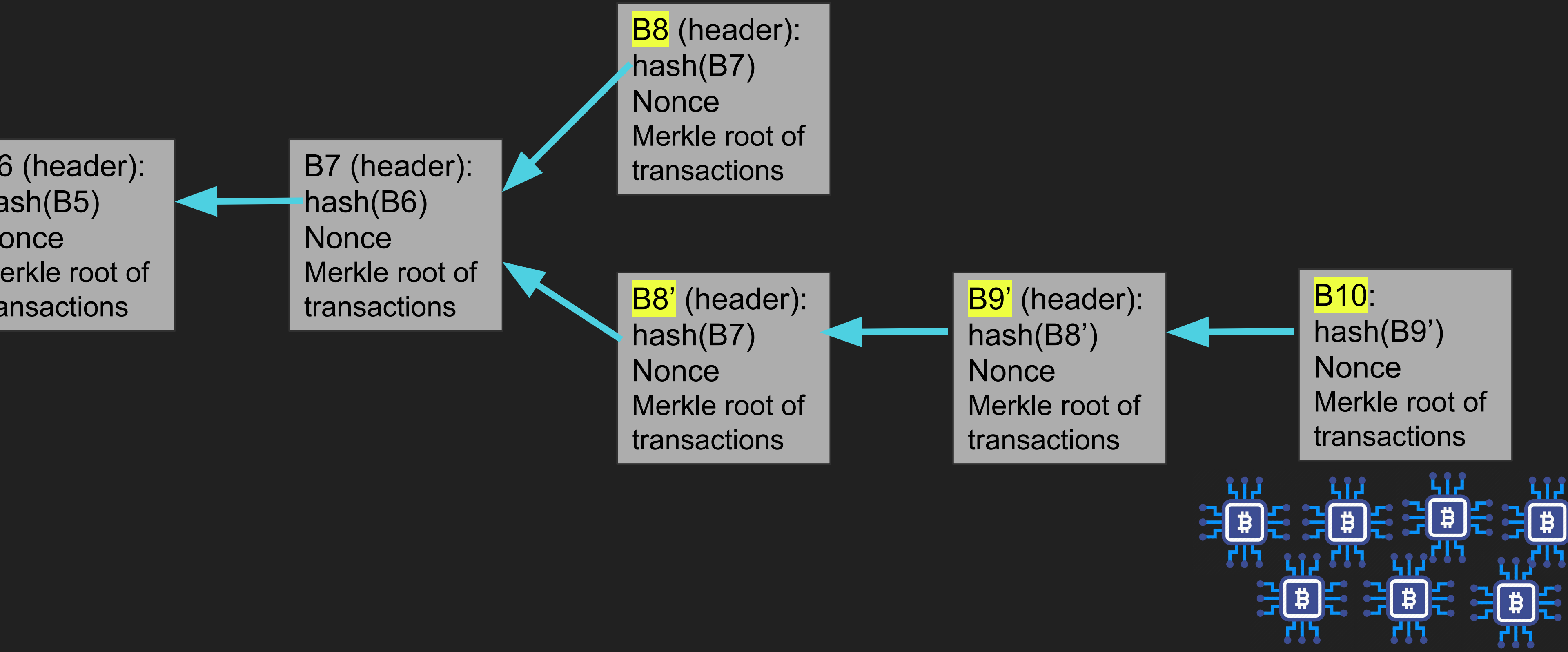
Suppose Alice tells some peers that Alice pays Dave 1 BTC, and tell other peers that Alice pays Ted 1 BTC




6 (header):
ash(B5)
once
erkle root of
ansactions
B7 (header):
hash(B6)
Nonce
Merkle root of
transactions
B8 (header):
hash(B7)
Nonce
Merkle root of
transactions
B8’ (header):
hash(B7)
Nonce
Merkle root of
transactions


6 (header):
ash(B5)
once
erkle root of
ansactions
B7 (header):
hash(B6)
Nonce
Merkle root of
transactions
B8 (header):
hash(B7)
Nonce
Merkle root of
transactions
B8’ (header):
hash(B7)
Nonce
Merkle root of
transactions
B9’ (header):
hash(B8’)
Nonce
Merkle root of
transactions


6 (header):
ash(B5)
once
erkle root of
ansactions
B7 (header):
hash(B6)
Nonce
Merkle root of
transactions
B8 (header):
hash(B7)
Nonce
Merkle root of
transactions
B8’ (header):
hash(B7)
Nonce
Merkle root of
transactions
B9’ (header):
hash(B8’)
Nonce
Merkle root of
transactions


6 (header):
ash(B5)
once
erkle root of
ansactions
B7 (header):
hash(B6)
Nonce
Merkle root of
transactions
B8 (header):
hash(B7)
Nonce
Merkle root of
transactions
B8’ (header):
hash(B7)
Nonce
Merkle root of
transactions
B9’ (header):
hash(B8’)
Nonce
Merkle root of
transactions
B10:
hash(B9’)
Nonce
Merkle root of
transactions

Can a attacker modify an existing block in the middle of the chain?

Answer: no

Since the “hash(previous block) in the next block will be wrong, and the effectively attacker creates a fork

Assumption: good miners outnumber bad miners. So the longest PoW chain represents the truth.

Why would miner want to solve the puzzle?

There are some incentives

Why would miner want to solve the puzzle?

There are some incentives

1. Each time a peer mines a block, it gets 6.25 bitcoin (currently)

Why would miner want to solve the puzzle?

There are some incentives

1. Each time a peer mines a block, it gets 6.25 bitcoin (currently)
2. It puts its public key in a special transaction in the block

Why would miner want to solve the puzzle?

There are some incentives

1. Each time a peer mines a block, it gets 6.25 bitcoin (currently)
2. It puts its public key in a special transaction in the block
3. Every transaction also can include a fee to incentivize the miner to include their transactions

Why would miner want to solve the puzzle?

There are some incentives

1. Each time a peer mines a block, it gets 6.25 bitcoin (currently)
2. It puts its public key in a special transaction in the block
3. Every transaction also can include a fee to incentivize the miner to include their transactions
4. Transactions with no fees are hardly picked up by miners.

How does Bitcoin makes sure each transaction is created in 10 minutes

The puzzle, finding hash(block) has N leading zeros, is actually tuneable. N could vary depends on how fast people mine bitcoin.

The PoW difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases

Weak points?

Block size is fixed 1 MB, limiting the number of transactions in one block.

- Soft fork
- Hard fork Bitcoin scalability problem - Wikipedia

Energy waste

bitcoin miners are expected to consume roughly 130 Terawatt-hours of energy (TWh), which is roughly 0.6% of global electricity consumption.

Alternative of PoW?

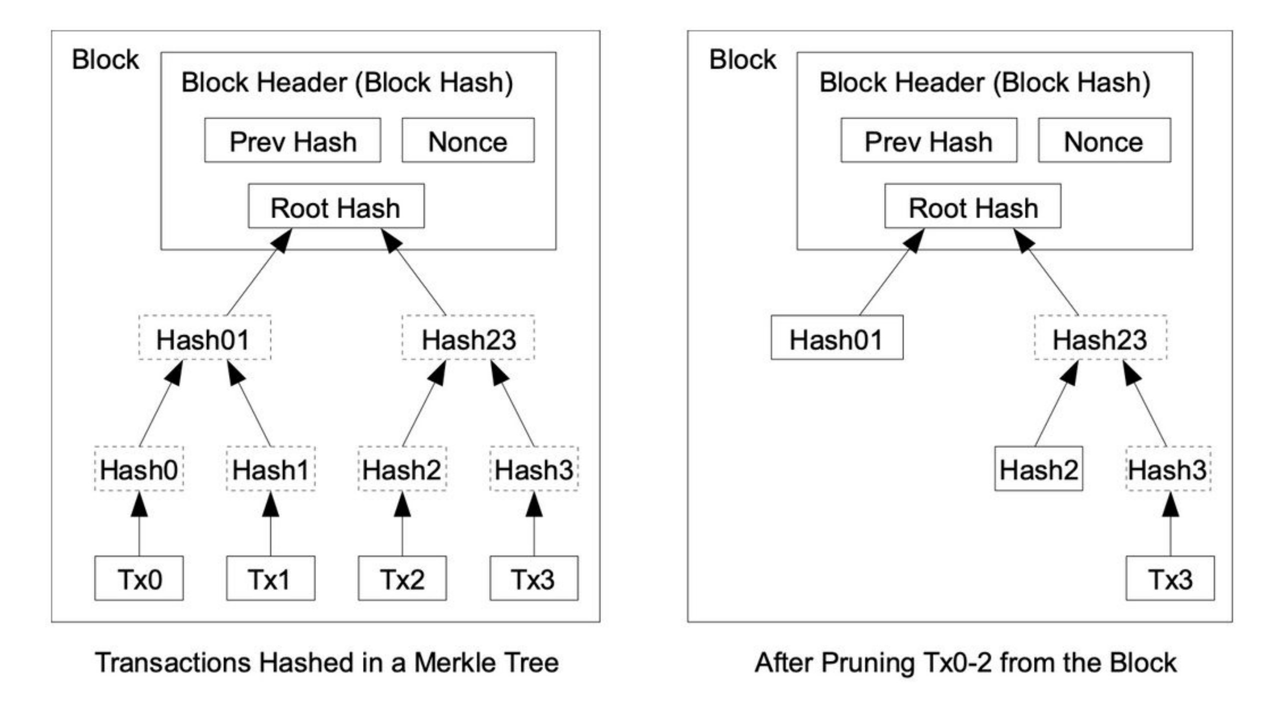
Proof of Stake? Ethereum Blockchain

Hyperledger in a permissioned network?

Optimization

Merkle Tree

- Only keep root hash
- Delete the interior hash values to save disk



Summary

Public, signed transaction log = digital currency

Fraud ⇔ Computationally Infeasible

The PoW agreement is very new and influential