Understanding Bitcoin: A Peer-to-Peer Electronic Cash System

•••

- Mayuri Shimpi

Introduction

A peer-to-peer online cash payment system that allows payments to be sent from one party to another directly.

Prevent double-spending by eliminating third party

Network timestamps transactions by hashing them in an ongoing hashing-based-proof-of-work

CPU power controlled by cooperating nodes

Background

Traditionally, banking systems serve as a trusted third party to process electronic transactions.

It suffers from the inherent problems of a trust based system.

Transactions are not completely non-reversible

High cost of mediation

Small risk of fraud persists

Overview of Bitcoin

A system for electronic transactions without relying on trust.

A framework of e-coin that is made from digital signatures and eliminates double-spending.

A peer-to-peer network based on proof-of-work to record public history of transactions that become computationally impractical for an attacker to change.

Robust network with unstructured simplicity: nodes work without coordination.

Nodes vote with their CPU power, expressing acceptance of valid blocks.

Rules and incentives can be enforced with this consensus mechanism.

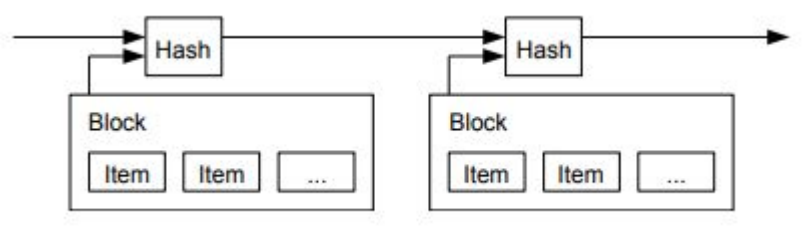
Transactions

E-coin : chain of digital currency

To avoid double-spending, transactions must be publicly announced

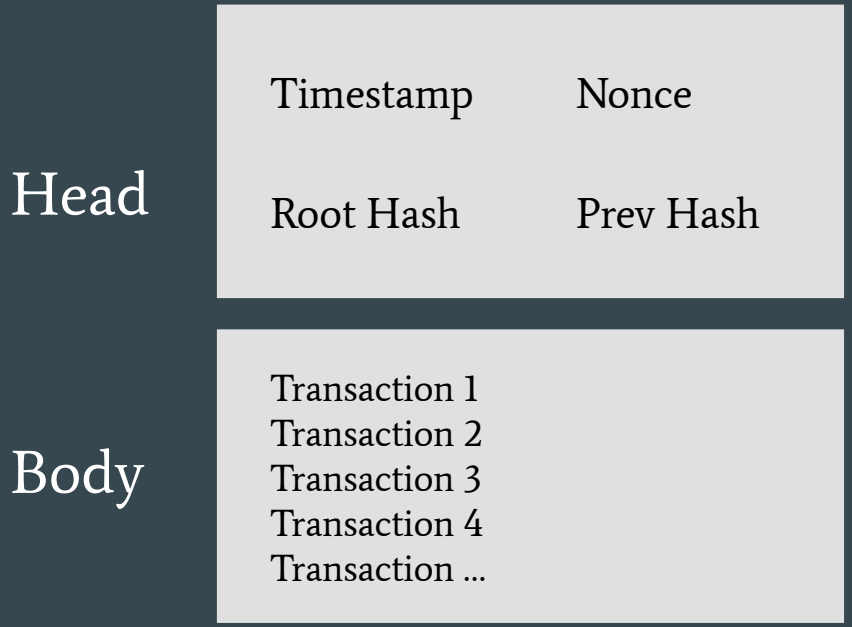
All participants must agree on single history of the order in which transactions arrived.

- Timestamp server



- Proof-of-work
- Network

Block


Head
Timestamp
Nonce
Root Hash
Prev Hash
Body
Transaction 1
Transaction 2
Transaction 3
Transaction 4
Transaction ...

Header

Timestamp: stores the time at which the block got created

Nonce: random value

Prev hash: Current block points to the previous block by storing the value of the hash in the header

Root Hash: Computed using Merkle trees

Transaction of the block are converted to hash using the SHA256

Merkle trees are constructed using these hash values.

Root Hash is the hash value of the root of the merkle tree.

Search operations are performed on the hash value

Byzantine generals problem

Byzantine problems describes the difficulty distributed systems have in agreeing on the same truth.

How can members of a network agree on a specific reality when no one can verify the identities of other members?

Problem:

Multiple generals besiege Byzantium.

They have encircled the city, and need to decide when to attack as a group

Any messages they send or receive can be intercepted or deceptively sent by a defender.

How can they attack simultaneously to win? Reach consensus and act in coordination.

Byzantine fault tolerance

The Byzantine generals problem must be solved if a dispersed group of nodes needs to achieve reliable communications.

Reasons why a distributed computer system could fail:

Software defect, malicious attack, hardware malfunction

These hinder nodes from reaching consensus on distributed network

A system that exhibits different behaviour to different observer: Byzantine Failure

Byzantine fault tolerance is the ability to defend against these conditions.

How could a society build a monetary system that all members can trust and agree on?

How does bitcoin solve the Byzantine Generals problem?

Can be solved using Blockchain: communicate safely and securely in an unpredictable environment.

Handle ownership and avoid double spending.

Cryptographic security and public key encryption.

The identity of a user is verified using public key encryption

Proof-of-work

A system for participants to agree on a single history of the order in which transactions were received.

Proof of work: Implements distributed timestamp server on a peer-to-peer basis

The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits.

This increases the work required exponentially depending on the number of 0 bits.

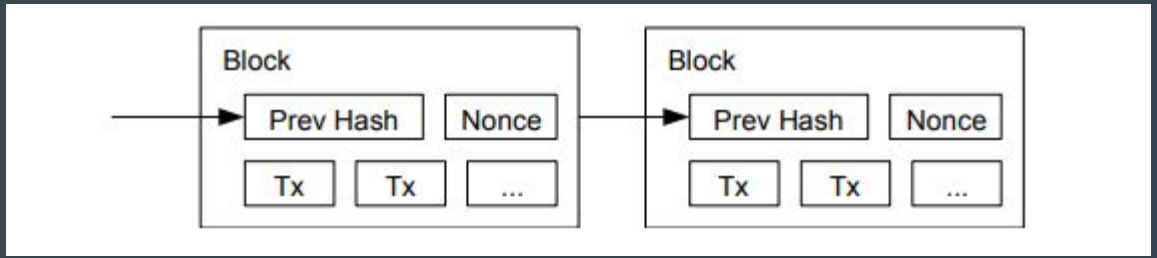
Can we verified using a single hash.

Proof-of-work

In this system (a timestamp network), the value of nonce of the block is incremented by one until a value is found that gives the block's hash the required zero bits.

One the proof-of-work is satisfied, the value of the hash cannot be changed without redoing the work.

As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



Proof-of-work

Proof-of-work is essentially one-CPU-one-vote.

Majority decision represented by longest chain: greatest PoW.

If majority of the network is controlled by honest nodes, honest chain will grow the fastest and outpace any bad actors.

To modify a past block, the bad actor will have to redo all the PoW of the block and previous blocks.

Probability diminishes exponentially as more blocks are added to the network.

Proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour.

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.

Nodes always consider the longest chain to be the correct one and will keep working on extending it.

Incentive

1. The first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation.
2. Incentive is funded with the transaction fees.

The incentive can encourage nodes to stay honest.

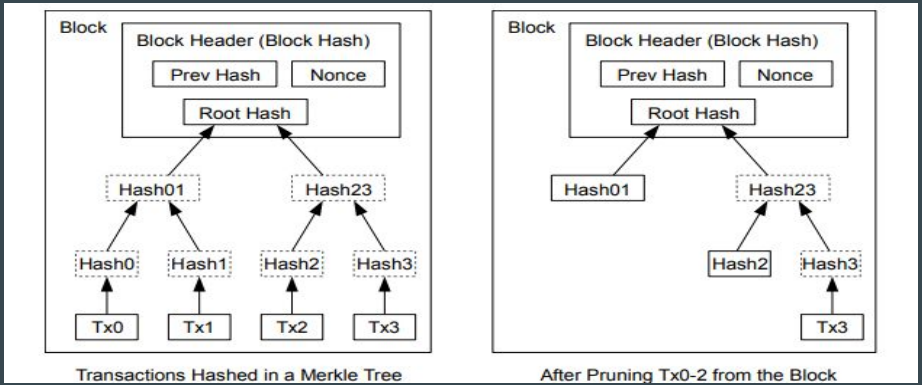
Playing by the rules might favour the bad actor (who has assembled more CPU power than rest of the nodes) by being able to generate new coins, than to undermine the system and the validity of his own wealth.

Reclaiming disk space

Once we have enough blocks, disk space can be reclaimed by discarding the transactions that occurred before the current block.

Transactions are hashed using Merkle trees.

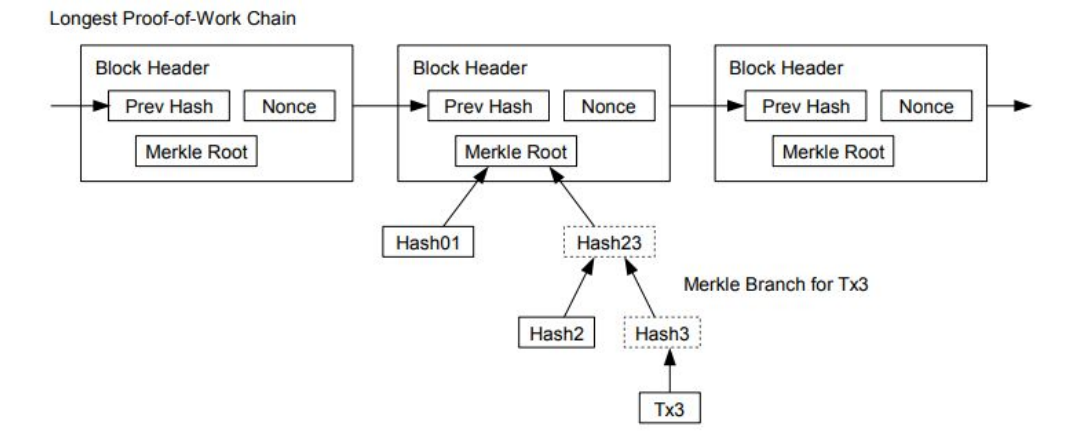
Only the root hash is included in the current block. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



Payment verification

To verify a transaction, user needs a copy of the headers of the longest PoW chain and obtain a merkle branch linking the transaction to the block that it is timestamped in.

Linking it to its place in the chain, can help verify that the network has accepted this and the blocks running after it have confirmed this.



Privacy

Traditional banking models achieve privacy by limiting access to information to the parties involved and the trusted third party.

In Bitcoin, all transactions are announced publicly

Privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous.

A new key pair should be used for each transaction to keep them from being linked to a common owner.

Risk: if the owner of a public key is revealed, then by linking it to other transactions, one can confirm that they belong to the same owner.