Bitcoin: A Peer-to-peer Electronic Cash System

Author: Satoshi Nakamoto

Presenter: Yiqin Qiu
Motivation

- Traditional trust-based systems (banks)? Published in 2008, where a financial crisis happened.
  - On the “Genesis of Bitcoin” marks “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks”

- A new electronic transaction system
  - Fully decentralized
  - Cryptographic-based
  - Fraud-proofing (non-reversible, prevent double-spending)
What is Bitcoin?

- An electronic currency
  - Value depends on market

- Based on a chain of blocks that contains the whole transaction history
Some Crypto Primitives

- **Hash Functions (SHA-256)**
  - A function that maps input to a 256-bit output
  - The outputs appear uniformly random
  - Collision resistant
  - Computationally impossible to reverse the output

- **Digital Signature**
  - The signer has a pair of (public key (pk), secret key (sk)); pk is known by all others
  - For each message m the signer wants to send, the signer compute s=Sign(m, sk), and sends(m, s)
  - For each message (m, s), the receiver check whether Verify(s, pk)=m
  - Computationally impossible to forge s without sk
The Design of Bitcoin

- Coin & Transaction
- Structure of the block/chain
- Timestamp server
- Proof of work
- Network agreements
- Incentive
- Fraud proofing
- Privacy
- Some optimizations
**Coin & Transactions**

- **What is a coin?**
  - A chain of signatures
  - Representing ownership of this coin

- **From single cent to lots of money**
  - Combining and splitting them in a single transaction
  - Inputs are previous transactions' outputs

---

**Structure of a coin**

---

**Hash**

- d514ff14c1e18778d9f064452b2dad707925e50bb689876a4370d2... (2020-12-06 17:50)
  - 1FMU6xQvEtxTQzka3CBB.JW3c.JKcJzJbr5: 0.00068096 BTC
  - 1FMU6xQvEtxTQzka3CBB.JW3c.JKcJzJbr5: 0.00137521 BTC
  - 1FMU6xQvEtxTQzka3CBB.JW3c.JKcJzJbr5: 0.00379343 BTC

**Fee**

- 0.00088876 BTC
  (180.241 sat/B - 45.060 sat/WU - 482 bytes)
Structure of the block/chain

- **Block**
  - Previous hash
  - Transactions in this block
  - Nounce, for the proof of work

- **Block Chain**
  - Attach the blocks one by one to form a chain
  - Should be the single history of the order of transactions
Timestamp Server

- Taking hash of a block to be timestamped
- Widely spread this hash
- Proves the existence of the block at that time
- Forms a chain for verification purpose
Proof of Work

Not all blocks are valid!

- Prev hash helps to identify the chain
- Transactions in this block should all be valid
- A correct **nounce** should be computed to prove your work
Nounce

- Block hash = H(everything in the block)

- The ONLY thing you can change is the nounce.

- A valid block has a block hash with a certain number of preceding 0s.
  - Hash appears uniformly random so you can only guess
  - The number \( N \) of preceding 0s controls the difficulty (average amount of work)

\[
P(\text{the hash has } N \text{ preceding 0s with any nounce}) = 2^{-n}
\]
Network Agreements

1. Transactions are signed by the sender for verification
2. Broadcast the transaction to all nodes in the network
3. Other nodes collect the transaction into a pending block
4. All nodes work to find the proof-of-work
5. If a node succeeds, it broadcasts the new block to the network
6. All nodes check the validity of the new block
7. If it is valid, all nodes express their acceptance by working on a new block with this block's hash as prev_hash
Network Agreements

- All correct nodes take the longest chain as the correct one
  - In case of a tie, take the one that first reaches
  - Tie will break after the next block is attached to one of them

- Tolerate drops/delays
  - As long as it reaches many nodes, the chain can still be extended
  - The dropped block can be re-requested after the next cycle
Incentive

Building and chaining blocks costs us:

- CPU (computation) power
  - Electricity fee
  - Facility fee
  - etc.

Why do we help others at our own cost?

Building and chaining blocks gives us:

- MORE MONEY (BTC)!

Two sources:

- Each transaction contains transaction fee
- A valid new block awards BTC to the creator
Creating Blocks

- We don’t want blocks to be created too fast
  - Network delays
  - Transaction rate

Therefore, the systems dynamically set $N$ to have an average block creation time of 10 min.

- We also do not want the bitcoin to be unlimited
  - Cause inflation

Therefore, ~ each 4 years (210000 blocks), the reward of creating valid blocks halves.
Fraud Proofing

- Due to digital signature, attackers cannot forge transactions
  - Cannot let others send BTC to you without their secret key
  - Only way to make money is to send BTC to someone for some exchange, and reverse the transaction
- To reverse the transaction in block $K$
  - The attacker have to re-compute all the proof-of-work from block $K$ to the current block
- If block $K$ is the current block
  - It must compute faster than all other nodes by two blocks
  - First recompute block $K'$ without the transaction
  - Then compute the block $K+1$ attaching to block $K$
- If is a previous block
  - Even harder, probability drops exponentially w.r.t. the gap of blocks
- Key to fraud proofing
  - Do not trust a transaction until several blocks are attached
- If you really have such computation power
  - Why don't you create valid blocks to earn money?
Fraud Proofing Case Study

- B2 contains a transaction A->B, B2’ doesn’t contain this transaction
- B2 is sent to B, B2’ is not sent to B
- However, B is smart enough
  - He waits until some blocks are attached

- The only way to trick B
  - Keep pace with the chain
  - Keep B on the top chain
  - Keep B away from the bottom chain
  - Until B trust that enough blocks have been attached
Calculations

- \( q \) is the fraction of attackers’ cpu power
- \( z \) is the leading blocks of the longest chain

Solve for \( P < 0.001 \)

\[
\begin{align*}
q &= 0.1 & z &= 0 & P &= 1.0000000
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 1 & P &= 0.2045873
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 2 & P &= 0.0509779
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 3 & P &= 0.0131722
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 4 & P &= 0.0034552
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 5 & P &= 0.0009137
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 6 & P &= 0.0002428
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 7 & P &= 0.0000647
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 8 & P &= 0.0000173
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 9 & P &= 0.0000046
\end{align*}
\]
\[
\begin{align*}
q &= 0.1 & z &= 10 & P &= 0.0000012
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 0 & P &= 1.0000000
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 5 & P &= 0.1773523
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 10 & P &= 0.0416605
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 15 & P &= 0.0101008
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 20 & P &= 0.0024804
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 25 & P &= 0.0006132
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 30 & P &= 0.0001522
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 35 & P &= 0.0000379
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 40 & P &= 0.0000095
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 45 & P &= 0.0000024
\end{align*}
\]
\[
\begin{align*}
q &= 0.3 & z &= 50 & P &= 0.0000006
\end{align*}
\]
Privacy

- Only thing needed for transaction is your key pair and your virtual address
  - No identity revealing

- If you want, change your address & key pair often
Disk Space Optimizations

- Transactions can grow “fast”
- Old enough transactions can be discarded
- For old enough blocks, we only store the transaction’s merkle tree root hash
Simplified Payment Verification

- Only keeps block headers of the longest chain
- Cannot check transactions by himself
- Query others by the timestamp of transactions
Conclusion

- Verify ownership
  - Digital signature
- Prevent double-spending
  - Chain of block contains all transactions
- Make reversing transactions computationally impossible
  - Proof-of-work
  - Accepting the longest chain
- No strong assumption of the network and the nodes
  - As long as honest nodes have a majority of computational power
- A new perspective of consensus problem against attackers