# **Bitcoin whitepaper**

* Allow **online transaction without** any traditional **financial institutions**
* **Digital signatures** provide part of the solution,
* 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
* 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.

Introduction:

* 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 nonreversible services
* Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers.

Transactions:

* Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

Timestamp Server:

* A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash.
* The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash.
* Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

Proof-of-Work

* 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
* For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits.
* Once the CPU effort has been expended to make it satisfy the proof-of-work, the block 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.
* The proof-of-work also solves the problem of determining representation in majority decision making.
* Proof-of-work is essentially one-CPU-one-vote
* The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes.

Network

The steps to run the network are as follows:

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.

If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first.

In that case, they work on the first one they received, but save the other branch in case it becomes longer.

The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

Incentive:

By convention, 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, since there is no central authority to issue them.

In our case, it is CPU time and electricity that is expended. The incentive can also be funded with transaction fees.

If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction.

If (output\_value<input\_value){

Transaction fee= input\_value-output\_value

}

Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The statement you provided is describing a key concept in cryptocurrency economics, specifically in the context of blockchain networks that use a Proof of Work (PoW) consensus mechanism like Bitcoin. Let's break it down step by step:

**Predetermined Number of Coins**: In many cryptocurrencies, there is a fixed or predetermined limit on the total number of coins that can ever be created. For example, in the case of Bitcoin, the maximum supply is capped at 21 million coins.

**Incentive for Miners**: In PoW-based blockchain networks, miners are responsible for validating and adding transactions to the blockchain. They do this by solving complex mathematical puzzles, which requires computational power and energy expenditure. In return for their efforts, miners are rewarded with newly created coins and transaction fees.

**Transition to Transaction Fees**: Initially, when a cryptocurrency is launched, miners are mainly rewarded with newly created coins. This serves as an incentive for miners to secure the network and validate transactions. As more coins are mined and enter circulation, the rate of coin creation decreases over time, and the reward for miners in terms of newly created coins diminishes.

**Inflation-Free**: The term "inflation-free" means that the total supply of the cryptocurrency remains relatively stable or grows very slowly. In traditional fiat currencies, governments can print more money, leading to inflation and reducing the value of the currency. In cryptocurrencies with a capped supply, like Bitcoin, the maximum number of coins is limited, so there is no risk of the supply increasing dramatically.

For example, in Bitcoin's case, the maximum supply is 21 million. Once all 21 million coins are mined, no more coins will be created. This limited supply is intended to make Bitcoin deflationary in the long run, meaning its value is expected to increase over time due to scarcity, assuming demand remains constant or grows.

**Reliance on Transaction Fees**: When the predetermined number of coins (e.g., 21 million in Bitcoin) is reached, miners can no longer receive newly created coins as a reward. At this point, they are solely compensated through transaction fees paid by users who want their transactions to be prioritized and included in the blockchain. Transaction fees become the primary incentive for miners to continue securing the network and processing transactions.

In summary, the statement is describing the economic model of certain cryptocurrencies, such as Bitcoin, where the issuance of new coins to miners decreases over time until it reaches a predetermined maximum supply. At that point, miners rely entirely on transaction fees as their incentive, and the cryptocurrency becomes inflation-free, meaning its supply remains stable and predictable, which can have implications for its long-term value and economic stability.

Reclaiming Disk Space:

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be store

The passage you've provided is describing a mechanism used in blockchain technology to save disk space and efficiently manage transaction data within a block. Let's break down the key concepts involved:

**Latest Transaction and Blocks**: In a blockchain, transactions are grouped together into blocks. The "latest transaction" refers to the most recent transaction that has been added to a block.

**Block Burial**: When a transaction is included in a block, it needs to be confirmed by the network through a process called mining. Once a sufficient number of blocks have been added on top of the block containing the transaction (confirmations), it is considered secure, and the transaction is said to be "buried" under these blocks.

**Discarding Spent Transactions**: After a transaction is sufficiently buried, it can be safely discarded from the blockchain's storage to save disk space. This is because older transactions that have been confirmed and are deeply buried are unlikely to be altered or rolled back.

**Merkle Tree**: A Merkle Tree is a data structure used to efficiently store and verify the integrity of a large set of data, such as transactions in a block. It works by hashing pairs of data, then hashing the results together until a single root hash is obtained. This root hash is a compact representation of all the data in the tree.

**Block's Hash**: Every block in a blockchain has a unique identifier called a hash. This hash is calculated based on various components of the block, including the transactions it contains.

**Including Only the Root Hash**: To save space in the block and ensure the integrity of the included transactions, only the root hash of the Merkle Tree is included in the block's hash. This means that even though the Merkle Tree contains all the transaction data, only a single hash value (the root) needs to be stored in the block's header.

**Compacting Old Blocks**: As time goes on and more blocks are added to the blockchain, older blocks become less relevant for transaction verification. To save even more disk space, these old blocks can be "compacted" by removing the branches of the Merkle Tree that are no longer needed. The root hash remains, but the interior hashes can be discarded.

In summary, this mechanism allows a blockchain to efficiently manage its transaction data by using Merkle Trees to create a compact representation of transactions within blocks. This helps save disk space over time by allowing spent transactions to be safely discarded while still preserving the integrity and security of the blockchain.

Simplified Payment Verification:

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.

The explanation you provided describes a method for verifying payments in a blockchain network, specifically in a proof-of-work-based blockchain like Bitcoin. Let's break down the process step by step:

**Block Headers**: In a blockchain, transactions are grouped into blocks. Each block contains a set of transactions, and it has a unique identifier called a "block header." This block header includes important information such as a timestamp, a reference to the previous block (creating a chain of blocks), and a cryptographic hash of the block's contents.

**Longest Proof-of-Work Chain**: In a blockchain, multiple nodes in the network work to create new blocks by solving complex cryptographic puzzles, a process known as proof-of-work. The chain with the most cumulative proof-of-work is considered the valid and longest chain. This chain represents the history of transactions in the network.

**Querying Network Nodes**: To verify a payment, a user doesn't need to run a full network node, which would require downloading and storing the entire blockchain. Instead, they can query network nodes (other participants in the network) to obtain block headers. These block headers represent the "chain" of blocks in the network.

**Merkle Branch**: To verify a specific transaction, the user also obtains a Merkle branch. A Merkle branch is a set of cryptographic hashes that link the target transaction to a specific block within the blockchain. It's a way to prove that a transaction is included in a block without having to verify all transactions in that block.

**Verification**: With the block headers and the Merkle branch, the user can verify the payment. Although they can't check the transaction themselves, they can check that a network node has accepted it. Here's how the verification process works:

The user starts by checking the block header of the block in which the transaction is timestamped. This block header includes a cryptographic hash of the entire block, including the transaction.

Using the Merkle branch, the user can calculate a cryptographic hash that represents the transaction. This hash should match the one included in the block header. If it matches, it means that the transaction is indeed included in that block.

Furthermore, as more blocks are added to the chain after the block containing the transaction, it becomes increasingly secure and accepted by the network. Each subsequent block adds more "confirmations" to the transaction, making it even harder for anyone to dispute its validity.

In summary, this method allows users to verify payments on a blockchain without running a full node by relying on the block headers and Merkle branches. This way, users can have confidence that their transactions are valid and accepted by the network.

While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network.

One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency.

Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

Combining and Splitting Value:

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

The statement you provided is describing how transactions work in a digital currency system, most likely referring to Bitcoin or a similar blockchain-based cryptocurrency. Let's break down the explanation:

**Handling Coins Individually**: This part suggests that in a digital currency system, such as Bitcoin, each unit of the currency (in this case, a cent) could technically be treated as a separate entity. In other words, you could send or receive each cent individually like you would with physical coins or bills.

**Unwieldy to Make Separate Transactions**: However, it's impractical and cumbersome to perform separate transactions for every cent. Imagine having to create a new transaction on the blockchain for each cent you want to send or receive. This would create a lot of unnecessary work and would be inefficient.

**Multiple Inputs and Outputs**: To solve this problem, transactions in digital currencies can contain multiple inputs and outputs. Here's what this means:

**Inputs**: Inputs are the sources of funds for a transaction. Normally, there will either be a single input from a larger previous transaction (meaning you're using funds received from a previous transaction) or multiple inputs combining smaller amounts (combining several smaller previous transactions' funds).

**Outputs**: Outputs are where the funds are going. Typically, there are at most two outputs in a transaction:

One output is for the payment, which is the intended recipient of the funds.

The other output is for returning any change, if there is any, back to the sender. This is similar to when you pay for something with a larger bill and receive change in return.

So, instead of creating a separate transaction for each cent or small unit of currency, you can combine inputs from multiple sources and specify where the funds should go in the outputs. This makes transactions more efficient and practical for handling various amounts of currency in digital form, while also ensuring that any leftover funds are returned to the sender as change when necessary. This is a fundamental concept in how digital currencies like Bitcoin work and helps maintain the integrity and efficiency of the blockchain system.

It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

The statement you provided appears to be discussing a concept related to transactions in a specific context, possibly in the realm of data processing, blockchain technology, or distributed systems. Let me break down the explanation for you:

**Fan-Out**: Fan-out refers to a situation where a transaction depends on several other transactions, and those transactions, in turn, depend on even more transactions. In other words, it's a chain or network of transactions where one relies on many others in a cascading manner.

**Not a Problem Here**: The statement is asserting that in the particular system or scenario being discussed, this fan-out phenomenon is not a problem. This means that the system can handle transactions that have multiple dependencies without any significant issues.

**No Need for Complete Standalone Copy**: The crucial point being made is that in this system, there's no requirement to extract and maintain a complete, self-contained copy of the entire transaction history for each transaction. In some systems, especially those with complex dependencies, it might be necessary to maintain a full copy of all the data involved in each transaction to ensure its integrity and consistency. However, in this specific context, such a requirement is not present.

In simpler terms, the system being described is capable of efficiently handling transactions that rely on many other transactions without the need to create and manage separate, complete copies of the entire transaction history for each transaction. This design could be more efficient in terms of storage and processing, as it avoids the overhead of duplicating all the data for every transaction.

Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.

As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

The passage you've provided explains how the traditional banking model achieves privacy through limited access to information, and how cryptocurrencies, like Bitcoin, maintain privacy through different means.

**Traditional Banking Model Privacy**: In traditional banking, privacy is achieved by restricting access to financial information. Only the parties directly involved in a transaction and the trusted third party, which is typically the bank, have access to the details of the transaction. This means that your bank knows about your financial activities, but other people don't.

**Cryptocurrency Privacy**: Cryptocurrencies like Bitcoin operate differently. They rely on a public ledger called the blockchain, which records all transactions. However, the necessity to announce all transactions publicly means that anyone can see when someone is sending cryptocurrency to someone else. To maintain privacy in this context, several strategies are employed:

a. **Anonymous Public Keys**: While the public can see that transactions are happening, the identities of the people involved are not disclosed. Instead of using real names, cryptocurrency transactions use public keys, which are long strings of numbers and letters. These public keys serve as pseudonyms for users, making it difficult to link a transaction to a specific individual.

b. **New Key Pairs for Each Transaction**: To enhance privacy, it is recommended to use a new set of public and private keys for each transaction. This way, even if someone were to somehow link one of your public keys to your identity, they wouldn't be able to link it to your other transactions because each one uses a different key pair.

c. **Multi-Input Transactions**: In some cases, multiple inputs (sources of funds) are used for a single transaction. This can potentially reveal that those inputs are owned by the same person or entity. So, there is still some level of linking that is unavoidable in such situations.

d. **Risk of Deanonymization**: If someone manages to link a public key to a real-world identity, it could expose other transactions associated with that public key. This is why it's crucial for users to take steps to protect the anonymity of their keys.

To draw a parallel, the passage mentions stock exchanges, where information about individual trades (time and size) is made public, but the identities of the parties involved are not disclosed. This is similar to how cryptocurrencies provide a level of transparency regarding transactions while preserving user privacy through the use of pseudonymous public keys and other techniques.

In summary, while cryptocurrencies like Bitcoin provide a different model of financial privacy compared to traditional banking, they aim to maintain user privacy by using pseudonyms (public keys), generating new keys for each transaction, and limiting information leakage, although some degree of risk remains in certain situations.