Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Bit Coin: A Decentralized Framework For Secure Electronic Transaction

Vidyashree Akhouri[1] , Dr. Javed Wasim[2] , Prof. Mohammad Ubaidullah Bokhari[3]

[1]Research Scholar (Corresponding Author) Mangalayatan University, Aligarh, India. Email-Id: akhourividyashree392@gmail.com
[2]Department of Computer Engineering & Applications Mangalayatan University, Aligarh, India. Email-Id: javed.wasim@mangalayatan.edu.in
[3]Department of Computer Science Aligarh Muslim University, Aligarh, India. Email-Id: mubokhari.cs@amu.ac.in

A completely peer-to-peer form of digital money would enable online transactions to occur directly between users without the need for a financial intermediary. While digital signatures contribute to solving this issue, they alone are insufficient to eliminate the need for a trusted third party to prevent double-spending. Our proposed solution uses a decentralized peer-to-peer network to tackle the double-spending challenge. The network records timestamps for each transaction by incorporating them into a continuous chain secured by hash-based proof-of-work. This chain forms a ledger that cannot be altered without repeating the computational work. The longest chain serves not just as evidence of the chronological order of transactions, but also as proof that it was supported by the majority of computing power. Provided that most of the network's CPU power is held by honest participants rather than attackers, they will continue to extend the longest chain faster than any malicious actors. The system's structure remains simple—messages are transmitted on a best-effort basis, and nodes can disconnect and reconnect freely, automatically adopting the longest valid proof-of-work chain as the authoritative record of events during their absence.

Keywords: Peer-to-peer, Digital money, Online transactions, Financial intermediary, Digital signatures, Double-spending, Decentralized network, Timestamps, Hash-based proof-of-work, Ledger, Computational work, Longest chain, Chronological order, Majority computing power, Honest participants, Malicious actors, Best-effort transmission, Node reconnection, Valid chain, Authoritative record.

Introduction

Online commerce today largely depends on financial institutions acting as trusted intermediaries to handle electronic payments. Although this approach functions adequately in many cases, it is inherently limited by the trust-based model it relies on. For instance, transactions cannot be made fully irreversible because intermediaries must intervene in disputes when they arise. This necessity increases transaction costs and restricts the viability of small, casual payments. Furthermore, it removes the option for truly final payments in exchange for services that are themselves irreversible. The potential for chargebacks also forces merchants to distrust customers, often requiring unnecessary personal information. As a result, a certain level of fraud is simply accepted as an ongoing cost of doing business. While physical cash allows people to avoid these issues in face-to-face exchanges, there is currently no comparable system for remote transactions that does not require a trusted intermediary.

What is needed is a digital payment solution that relies on cryptographic validation rather than mutual trust. This would allow two parties to deal directly without the need for a third-party authority. Irreversible transactions would protect sellers from fraud and be secured by computational difficulty. Escrow protocols could be used to ensure fairness for buyers. This paper presents a way to address the issue of double spending by introducing a peer-to-peer distributed timestamp network. This network provides cryptographic proof of transaction sequence and ensures the system remains secure as long as the majority of computing power is controlled by honest participants—even in the face of coordinated attacks.

Transactions

An electronic coin can be described as a sequence of digital signatures. Each owner transfers the coin by signing a hash of the previous transaction along with the public key of the next owner. This signed data is then appended to the coin. The recipient can verify the legitimacy of the coin by checking the chain of signatures.

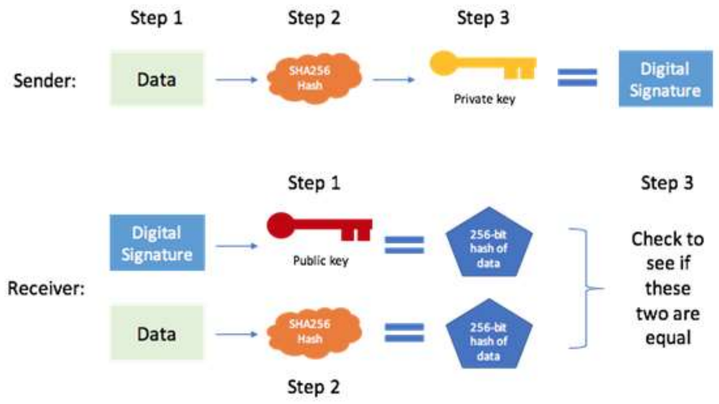


Figure 1: How does a bitcoin transaction actually work?[9]

However, a challenge arises: the recipient has no way of knowing whether a previous owner attempted to spend the same coin more than once. A typical approach to prevent this is to rely on a centralized trusted entity, often referred to as a mint, which validates each transaction to ensure no double-spending occurs. In such a system, coins must be sent back to the mint after each transaction so that new coins can be issued. Only those freshly issued by the mint are considered valid and safe from being spent twice.

This method places the entire system's trust in a single entity, similar to how traditional banks operate. All transactions depend on this central authority, which creates a single point of failure. To eliminate the need for a trusted third party, the recipient must confirm that a coin hasn't been previously spent elsewhere. Since only the first transaction involving a coin needs to be validated, any later double-spending attempts can be ignored. Detecting whether a coin has already been spent requires visibility into all transactions. In a centralized system, this is handled by the mint, which keeps records and determines transaction order. To achieve the same outcome without central control, transactions must be broadcast publicly. A consensus mechanism is then required to ensure all participants agree on a single, shared timeline of transactions. This allows the recipient to trust that, at the time of the transaction, the majority acknowledged it as the first valid use of the coin.

Timestamp Server

The proposed system initiates with a timestamp server that generates a cryptographic hash of a block of data and publicly disseminates it—such as through a newspaper or a Usenet post. This mechanism serves as verifiable proof that the data existed at the moment the hash was created. If a hash is demonstrably produced at a specific point in time, it confirms the existence of the underlying data at that same moment. Each subsequent timestamp includes the hash of the previous one, thereby forming a continuous and immutable chain. This structure enhances the integrity of the data over time, as each new timestamp reinforces the credibility of all prior entries in the chain.

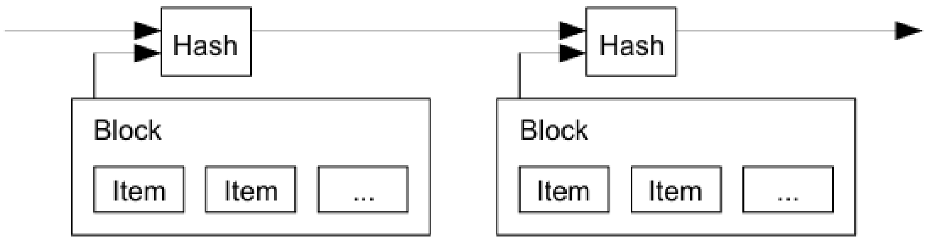


Figure 2: [10]

Proof-of-Work

To achieve decentralization, the system utilizes a peer-to-peer (P2P) network integrated with a proof-of-work (PoW) mechanism. This method draws inspiration from Adam Back's Hashcash protocol, which introduces a computational challenge to validate transactions. In this model, participants are required to identify a specific value, referred to as a nonce, which, when combined with the block's data, generates a hash output beginning with a defined number of leading zero bits. This process is deliberately computationally demanding, requiring significant processing resources.

Despite its complexity, once a valid hash is discovered, it becomes easy for other network nodes to verify. Upon completion of a valid proof-of-work, the associated block is considered finalized or "locked" and cannot be modified without re-performing the entire computational task. Furthermore, each new block added to the blockchain enhances the security and immutability of the preceding blocks, thereby increasing the overall integrity of the system.

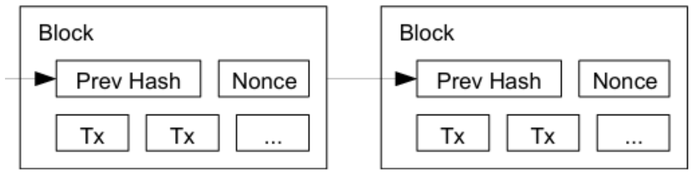


Figure3: [11]

Majority Decision and Security

Proof-of-work (PoW) establishes a fair majority system. If a system were to base votes solely on IP addresses, an attacker could create fake IPs to control the system. However, in PoW, votes are based on computing power, which is why it is referred to as the "one-CPU-one-vote" model. The most valid chain is the one with the most total work done. As long as honest nodes possess the majority of the network's computing power, their chain will grow the fastest.
If an attacker wishes to alter a past transaction, they would need to redo the proof-of-work for that block and all subsequent blocks. As new blocks are added, it becomes increasingly difficult for the attacker to compromise the system. The system automatically adjusts its difficulty to maintain a regular block creation speed.

Network Operation (Simplified)

1. New transactions are shared with all nodes in the network.
2. Each node collects these transactions into a block.
3. Each node then tries to solve a difficult proof-of-work for its block.
4. The first node to solve it sends the new block to the entire network.
5. Other nodes check the block to ensure its correctness.
6. They verify that all transactions are valid.
7. They also check that no transaction has been used before.
8. If the block is valid, the nodes accept it.
9. Once accepted, nodes start working on the next block.
10. The new block is linked to the previous one using its hash.

Chain Selection and Network Agreement

Nodes always follow the chain with the most total proof-of-work. This is usually the longest valid chain. They treat it as the correct version of the ledger and continue building new blocks on top of this chain. Occasionally, two different blocks may be broadcast simultaneously. These blocks might reach different parts of the network first. In such cases, some nodes may work on one block, while others work on the other. Each node continues with the block it received first but retains the other block in case it becomes the longer chain later. When the next block is added to one of the versions, making it longer, all nodes switch to this longer chain. This mechanism allows the network to automatically resolve conflicts.

New transaction messages do not have to reach every single node. As long as a sufficient number of nodes receive them, the transaction will eventually be added to a block. Block messages are also safeguarded against being lost. If a node misses a block, it will recognize the missing block when the next one arrives and will request the block it missed.

Incentive by Convention

In every block, the first transaction is special because it creates new coins and awards them to the block's creator. This provides nodes with a reason to support the network and aids in putting new coins into circulation, given the absence of a central authority for issuance. New coins are added gradually and in fixed amounts. This mechanism is similar to gold mining, where effort is expended to discover gold. In this system, the cost is reflected in CPU power and electricity consumption. Incentives also arise from transaction fees. If a transaction's input value exceeds its output value, the difference becomes a fee. This fee is added to the reward for the node that mines the block. Eventually, as a sufficient number of coins are in circulation, new coin rewards may cease. At that point, transaction fees alone will provide miners with compensation, making the system independent of inflation.

The reward structure also promotes honest behavior. If an attacker controls more CPU power than all other nodes combined, they could attempt to deceive the network. However, it is more profitable for them to act honestly. By mining in accordance with the rules, they earn more coins than they would through attempts to disrupt the system. Any attempt at cheating would undermine the value of their own holdings.Reclaiming Disk Space

Once a coin's latest transaction is deeply embedded in the blockchain, older transactions can be discarded to save disk space. This is safely done by storing transactions in a Merkle Tree, where only the tree's root is part of the block's hash. Consequently, old data can be deleted without affecting the block's hash. Older blocks can be reduced in size by trimming unused branches of the Merkle Tree. The interior parts of the tree, known as interior hashes, do not need to be retained.

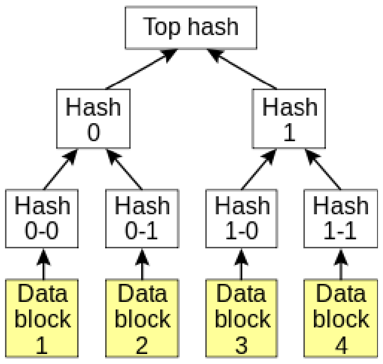


Figure 4:[12] Blockchain- what is the block hash?

Block Header Storage

A block header without any transactions takes up roughly 80 bytes. Assuming a new block is created every 10 minutes, the total annual storage required would be: 80 bytes × 6 blocks per hour × 24 hours per day × 365 days = approximately 4.2 megabytes per year. Given that standard computers in 2008 were typically equipped with 2 gigabytes of RAM, and considering Moore's Law—which projected an annual increase of around 1.2 gigabytes in memory capacity—storing block headers in memory should remain manageable well into the future.

Simplified Payment Verification (SPV)
Payments can be verified without the need to operate a full node. A user only needs to maintain a record of the block headers from the longest chain, which represents the most cumulative proof-of-work. This information can be obtained by querying nodes on the network until the user is confident they have the most up-to-date and valid chain.

To confirm a specific transaction, the user requests the Merkle path that connects it to the block in which it was included. While the user cannot independently validate the transaction, they can confirm that it is part of a block accepted by the network. As additional blocks are added on top of that block, the confirmation becomes more secure, indicating continued acceptance by the network.

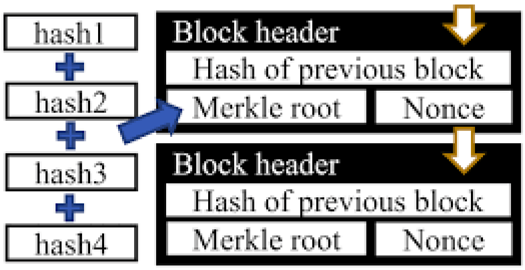


Figure: 5 The hash of the previous block, the Merkle root, and the Nonce are combined into a single hash [**13]**

Security and Verification Reliability
This method of verification remains dependable as long as the majority of the network is controlled by honest participants. However, it becomes more susceptible to manipulation if a malicious actor gains enough power to dominate the network. While full nodes are capable of validating transactions independently, the simplified verification approach can be deceived by false data created by an attacker, provided they maintain control over the network.

To reduce this risk, one possible safeguard is to allow the user's system to listen for warnings from other nodes when a suspicious or invalid block is detected. Upon receiving such alerts, the software can fetch the full block and related transactions to perform a deeper validation.

For organizations or businesses that handle high volumes of transactions, running a full node remains a better choice, offering greater control, stronger security, and faster confirmation times.

Transaction Structure and Efficiency

Handling each coin individually would be inefficient, especially for transactions involving small amounts. To make value transfer more practical, transactions are designed to support multiple inputs and outputs. Typically, a transaction might combine several smaller previous outputs or use a single larger one. The outputs generally include one directed to the recipient and another to return any leftover amount (change) to the sender.

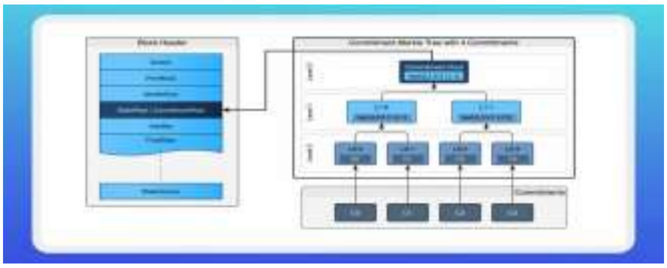
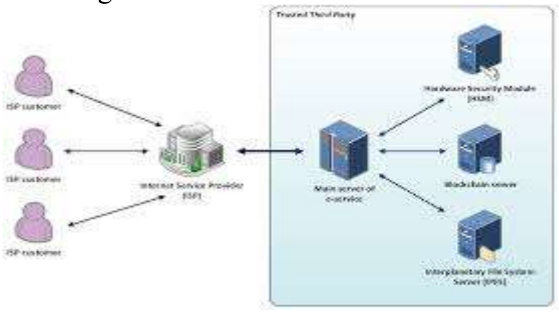


Figure 6: What is Simplified Payment Verification (SPV)? [14]

Transaction Structure and Validation
It is also important to note that the branching structure of transactions—where one transaction draws from several others, which in turn draw from many more—is not an issue. There is no requirement to reconstruct the full transaction history for validation.

Privacy and Transparency in Cryptocurrency Systems
In traditional banking, privacy is maintained by restricting transaction details to the involved parties and a trusted intermediary. In contrast, cryptocurrency systems require all transactions to be publicly visible, making that approach unfeasible. However, privacy can still be preserved in another way—by keeping public keys anonymous. While the public can observe that funds are being transferred, they cannot identify the individuals behind the transactions. This approach is comparable to how stock markets disclose the timing and size of trades without revealing the identities of the traders.


Trusted Third Party
ISP customer
ISP customer
ISP customer
Internet Service Provider (ISP)
Main server of e-service
Hardware Security Module (HSM)
Blockchain server
Interplanetary File System Server (IPFS)
Figure 6: Trusted Third Party Application in Durable Medium e-Service [15]

Enhancing Privacy with Key Rotation
To enhance privacy further, it is recommended to generate a new public-private key pair for every transaction. This practice helps prevent transactions from being linked back to the same user. That said, complete unlinkability is not always possible, especially in transactions involving multiple inputs, as they indicate common ownership. If the identity associated with one key becomes known, it could potentially expose other transactions belonging to the same individual.

Attacker Versus Honest Chain
Let us consider a case where an attacker tries to create a fake blockchain that grows faster than the real (honest) chain. Even if the attacker succeeds in making a longer chain, they still cannot

do anything they want. They cannot create fake money or steal coins they never owned. Honest nodes will never accept an invalid transaction. Therefore, the attacker can only try to reverse their own recent transaction—like trying to take back money they already spent.

The honest chain and the attacker's chain are like a race. Each time the honest network finds a block, it gets +1 ahead. Each time the attacker finds a block, the gap reduces by -1. This race is similar to a binomial random walk.

Chance of Catching Up (Gambler's Problem)

This situation is similar to the Gambler's Ruin problem. Imagine a gambler starts with a loss and keeps playing, trying to break even. We want to know the chance that the attacker (the gambler) can catch up from being behind by a certain number of blocks.

Let:

- p = chance that an honest node finds the next block
- q = chance that the attacker finds the next block
- z = how many blocks the attacker is behind

If the attacker is faster or equal ($q \geq p$), then he can definitely catch up (chance = 1). But if $p > q$ (honest nodes are faster), then the chance the attacker catches up becomes: (q / p)^z

This means the more blocks the attacker is behind (z increases), the less chance they have. The chance drops exponentially—very fast.

Waiting Time to Confirm a Transaction

Let us now look at how long a person should wait before being sure a payment cannot be reversed by an attacker. We assume the attacker wants to fool the receiver. He wants to make it look like he paid, but later replace that transaction with another one that sends the money back to himself. The receiver will find out eventually, but the attacker hopes it will be too late.

To protect against this, the receiver creates a new key pair and shares the public key right before the transaction. This makes sure the attacker cannot plan everything far in advance. Once the transaction is sent, the attacker secretly starts building an alternate chain that excludes that payment. The receiver waits until the transaction is in a block and then waits for z more blocks to be added after it. This makes the transaction deeper in the chain and harder to reverse.

Mathematical Model (Poisson Distribution)

Now we calculate the real chance that the attacker can still catch up. We use something called a Poisson distribution to model this.

The expected progress of the attacker is:

$\lambda = z \times (q / p)$

This is the average number of blocks the attacker might have found in that time. We then calculate the attacker's total success chance using this formula:

P = 1 − Σ (from k = 0 to z) [Poisson probability × (1 − (q / p)^(z − k))]

This removes the need to add up infinite values and gives a clean result. The attacker's chance is less than 0.1%.

Java Code – AttackerSuccessProbability.java

```
import java.lang.Math;
```

```
public class AttackerSuccessCalculator {

    public static double attackerSuccessProbability(double q, int z) {
        double p = 1.0 - q;
        double lambda = z * (q / p);
        double sum = 1.0;

        for (int k = 0; k <= z; k++) {
            double poisson = Math.exp(-lambda);

            for (int i = 1; i <= k; i++) {
                poisson *= lambda / i;
            }

            sum -= poisson * (1 - Math.pow(q / p, z - k));
        }

        return sum;
    }

    public static void main(String[] args) {
        // Example runs
        double[] qValues = {0.1, 0.3};
        int[] zValues = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50};

        for (double q : qValues) {
            System.out.println("q = " + q);
            for (int z : zValues) {
                double result = attackerSuccessProbability(q, z);
                System.out.printf("z = %-3d  P = %.7f%n", z, result);
            }
            System.out.println();
        }
    }
}
```

Example Output

```
q = 0.1
z = 0   P = 1.0000000
z = 1   P = 0.2045873
z = 2   P = 0.0509779
...
q = 0.3
z = 0   P = 1.0000000
```

```
z = 5    P = 0.1773523
...
```

References

1. Dai, W. (1998). b-money. Retrieved from http://www.weidai.com/bmoney.txt
2. Massias, H., Avila, X. S., & Quisquater, J.-J. (1999). Design of a secure timestamping service with minimal trust requirements. 20th Symposium on Information Theory in the Benelux.
3. Haber, S., & Stornetta, W. S. (1991). How to time-stamp a digital document. Journal of Cryptology, 3(2), 99–111. https://doi.org/10.1007/BF00196791
4. Bayer, D., Haber, S., & Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time-stamping. In Sequences II: Methods in Communication, Security and Computer Science (pp. 329–334). Springer.
5. Haber, S., & Stornetta, W. S. (1997). Secure names for bit-strings. In Proceedings of the 4th ACM Conference on Computer and Communications Security (pp. 28–35).
6. Back, A. (2002). Hashcash – a denial of service counter-measure. Retrieved from http://www.hashcash.org/papers/hashcash.pdf
7. Merkle, R. C. (1980). Protocols for public key cryptosystems. In Proceedings of the 1980 IEEE Symposium on Security and Privacy (pp. 122–133). IEEE. https://doi.org/10.1109/SP.1980.10006
8. Feller, W. (1957). An introduction to probability theory and its applications (2nd ed.). Wiley.
9. Marshall, B. L. (n.d.). How does a bitcoin transaction actually work? Medium. Retrieved from https://medium.com/@blairlmarshall/how-does-a-bitcoin-transaction-actually-work-1c44818c39969
10. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf
11. Ethereum Stack Exchange. (2016). What is a block hash? Retrieved from https://ethereum.stackexchange.com/questions/2100/what-is-a-block-hash
12. Avin, R., Liu, X., Mohanty, H., Norman, L., & Krishnamachari, B. (2018). The hash of the previous block, the Merkle root, and the Nonce are combined into a single hash. In Analysis of trade-offs in systems design of blockchain implementations in healthcare: A systematic review. Retrieved from https://www.researchgate.net/figure/The-hash-of-the-previous-block-the-Merkle-root-and-the-Nonce-are-combined-into-a-single_fig3_332101919
13. Cypherpunk Times. (2024). Can a crypto wallet be lightweight, private, and secure? The SPV model leads the way. Retrieved from https://www.cypherpunktimes.com/can-a-crypto-wallet-be-lightweight-private-and-secure-the-spv-model-leads-the-way/
14. Wang, M. (2024). Trusted third party application in durable medium e-service. Applied Sciences, 14(1), 191. https://www.mdpi.com/2076-3417/14/1/191