

MODULE 3

Concept of Double spending



- Double spending is the risk that the same digital coin is spent more than once.
- Since digital currency is essentially data, it can be copied easily.
- A malicious user may duplicate the same coin and use it for multiple transactions, which creates a serious problem of trust in digital money.
- To overcome this issue, blockchain technology records all transactions in a public ledger, and consensus mechanisms ensure that only one valid transaction is accepted.
- This prevents duplication and maintains the integrity and security of digital assets.

How it happens

- Digital data duplication: Unlike physical cash, digital currencies exist as data. A user can create multiple copies of the same currency file and try to spend them in different places.
- Unconfirmed transactions: A double-spending attack can occur when a merchant accepts a transaction that is not yet confirmed on the network, which can happen when a new block in the blockchain is not considered final.
- Replay attacks: A user might retransmit a valid transaction using the same funds to execute a double-spend.

Why it's a problem

- **Loss of trust**: The possibility of double-spending erodes trust in digital currency systems, as it allows for fraudulent activities and monetary losses.

- **Decentralized systems:** In decentralized systems like blockchain, there is no central authority to trace and stop such fraudulent transactions, making the prevention of double-spending a critical aspect of the system's design.

How it's prevented

- **Blockchain and consensus:** Blockchain technology, with its distributed ledger and consensus mechanisms, helps prevent double-spending by creating a secure, unalterable record of transactions.
- **Block confirmations:** Networks require multiple confirmations of a transaction before it is considered complete and final, a process that takes time but ensures the validity of the payment.
- **Unique identification:** Transactions are often tied to unique cryptographic functions and have to be verified by the network, preventing the same currency from being spent multiple times

Types of attacks

- **Race Attack:** Two conflicting transactions are sent almost simultaneously, hoping one is confirmed while the other is rejected.
- **Finney Attack:** Requires a miner to pre-mine a block containing a transaction and then spend the coins before the block is added to the chain.
- **51% Attack:** An attacker controls more than 50% of the network's hashing power, allowing them to potentially manipulate transactions and double spend.

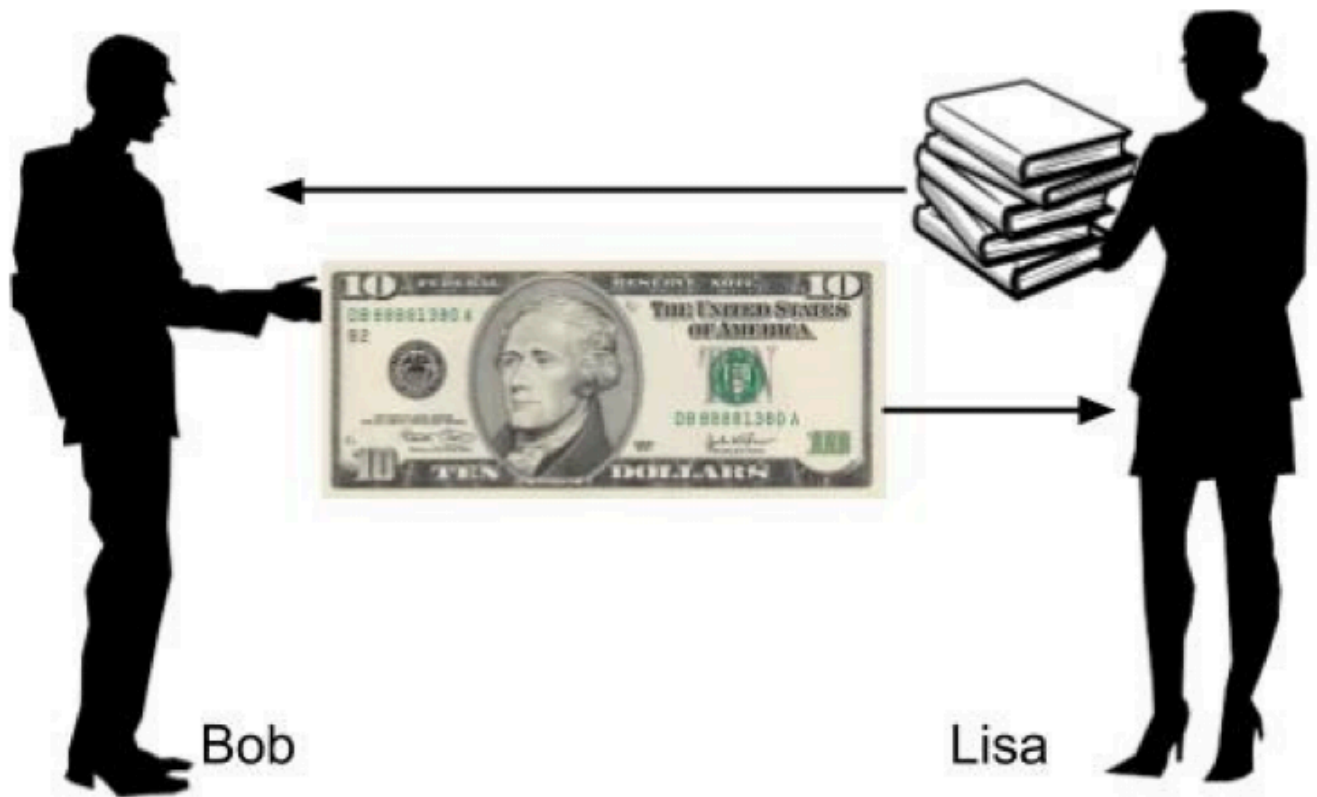
Prevention in blockchain

- **Consensus Mechanisms:** Protocols like Proof of Work (PoW) and Proof of Stake (PoS) make it difficult for malicious actors to alter the blockchain's history.
- **Transaction Verification:** Nodes independently verify each transaction to ensure funds haven't already been spent.
- **Public Ledger:** All transactions are recorded and visible, making double-spending attempts easy to detect.
- **Confirmation Mechanisms:** Transactions become more secure as they receive multiple network confirmations.

Key insights

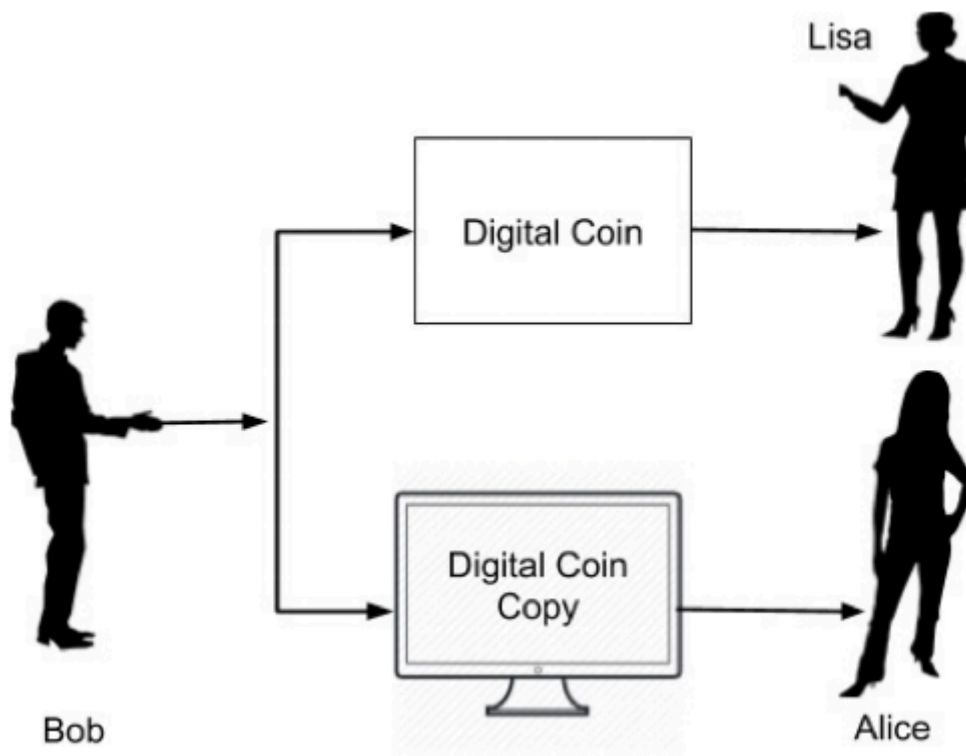
- The Bitcoin network has never experienced a successful double-spending attack due to its strong security measures.
- Larger, more established blockchain networks are more resistant to double-spending attacks, particularly 51% attacks, due to the high cost and difficulty involved in controlling the majority of the network's processing power.
- Individual users can reduce their risk by waiting for multiple confirmations before accepting a transaction as final.

Example : Consider a situation shown in



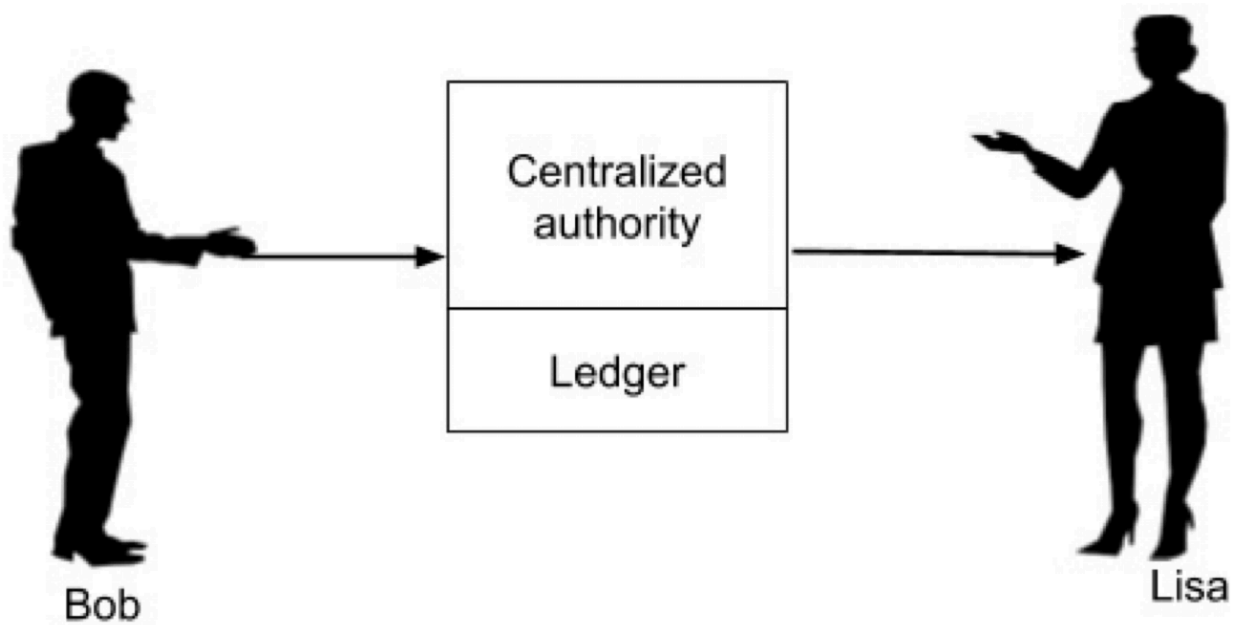
- Bob gives Lisa a \$10 note for a book.
- Once Lisa has the note, Bob cannot use the same \$10 again for another purchase, because the physical cash is now with Lisa.

Now, consider a situation where the money is paid in Digital form. This is illustrated in



- In digital form, money is just a **file stored on Bob's device**.
- Bob sends this digital money file to **Lisa**.
- But Bob can also send a **copy of the same file** to **Alice**.
- Now **both Lisa and Alice believe** they have received valid payment.
- Since there is no direct way to check if the file is original or copied, both may deliver their goods to Bob.
- This problem is called **Double Spending** – using the **same digital money more than once** to get goods/services.

To prevent the problem of double-spending, a centralized authority can be used to monitor and validate all transactions. This is shown in the image.



- A centralized authority (bank) keeps a ledger of all transactions.
- Bob sends his digital money to the bank first.
- The bank verifies Bob's balance and debits his account.
- The bank then credits Lisa's account with the money.
- This process ensures Bob cannot spend the same money twice (no double-spending).
- The bank also validates the authenticity of digital coins, blocking fake or duplicate ones.
- But maintaining a centralized authority is costly.
- Banks charge commissions/fees on every transaction to cover expenses.
- Fees can be especially high in international transfers involving multiple banks.

Hashing

Hashing is the process of converting input data of any length into a fixed-length output called a **hash value** or **message digest**.

- The hash looks like a sequence of numbers and letters.
- Hashing is mainly used in **cryptography**.
- Purpose: to ensure **data integrity**, **authentication**, and **digital signatures**.

Key Properties of a Cryptographic Hash Function

1. **Deterministic** : The same input always produces the same hash output.
2. **Fixed Output Length** : Regardless of the size of the input, the hash function produces a fixed-length output (e.g., SHA-256 always gives a 256-bit hash).

1. **Pre-image Resistance** : It should be computationally infeasible to find the original input given its hash value.
1. **Second Pre-image Resistance** : It should be very hard to find another input that produces the same hash as a given input.
1. **Collision Resistance** : It should be infeasible for two different inputs to produce the same hash value.
1. **Avalanche Effect** : A small change in input (even a single bit) should produce a completely different hash output.

Role of Hashing in Blockchain

1. **Block Identification**
 - Each block in a blockchain has a unique hash value that acts like its digital fingerprint.
 - This ensures that no two blocks are the same.
1. **Data Integrity**
 - If even a small change is made in the block's data, the hash value changes completely (avalanche effect).
 - This prevents tampering of stored data.
1. **Linking Blocks**
 - Every block contains the hash of the **previous block**.
 - This links the blocks together to form a secure chain.
2. **4.Proof of Work**
 - In Bitcoin and other blockchains, miners solve a mathematical puzzle by finding a hash with certain conditions (like leading zeros).
 - This ensures security and prevents easy manipulation.
1. **5.Fast Verification**
 - Instead of checking all the data, blockchain nodes can quickly verify transactions by checking the hash values.

In short: Hashing in blockchain secures the data, links blocks together, prevents tampering, and makes verification fast.

Blockchain Hashing Example

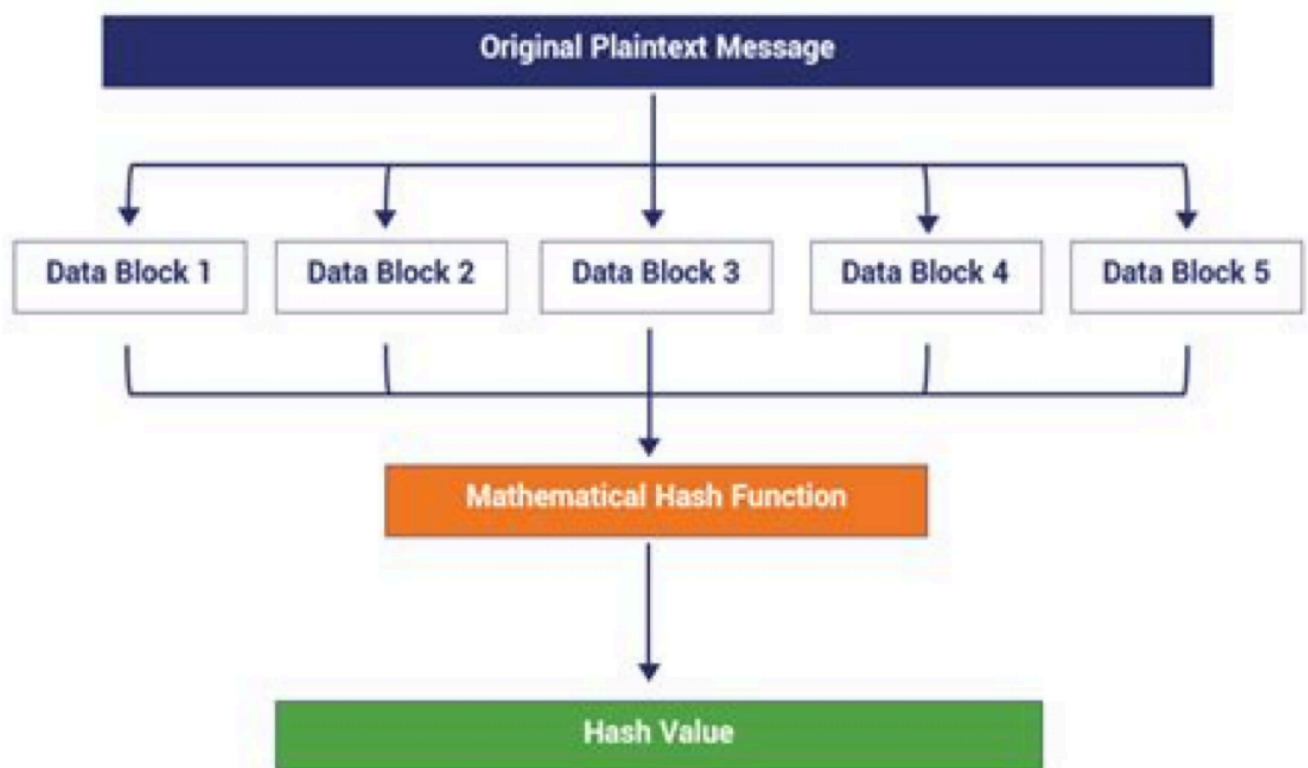
Alice sends 5 BTC to Bob



Hash Υ 3a6eb...9f2c1

- Fixed-length output
- Unique representation

How Hashing Works



 Blockchain Hashing Example

Scenario:

- **Transaction:** "Alice sends 5 BTC to Bob"

Process:

1. **Input Data:**
 - Transaction details: "Alice sends 5 BTC to Bob"
1. **Hashing:**
 - Apply a cryptographic hash function (e.g., SHA-256) to the input data.
1. **Output:**
 - Resulting hash (e.g., 3a6eb...9f2c1)

Key Points:

- **Fixed-Length Output:** Regardless of input size, the hash output is fixed (e.g., 256 bits for SHA-256).
- **Unique Representation:** Even a small change in input data alters the hash completely.
- **Tamper Detection:** Any modification in the transaction data changes the hash, signaling potential tampering.

Mining in Blockchain

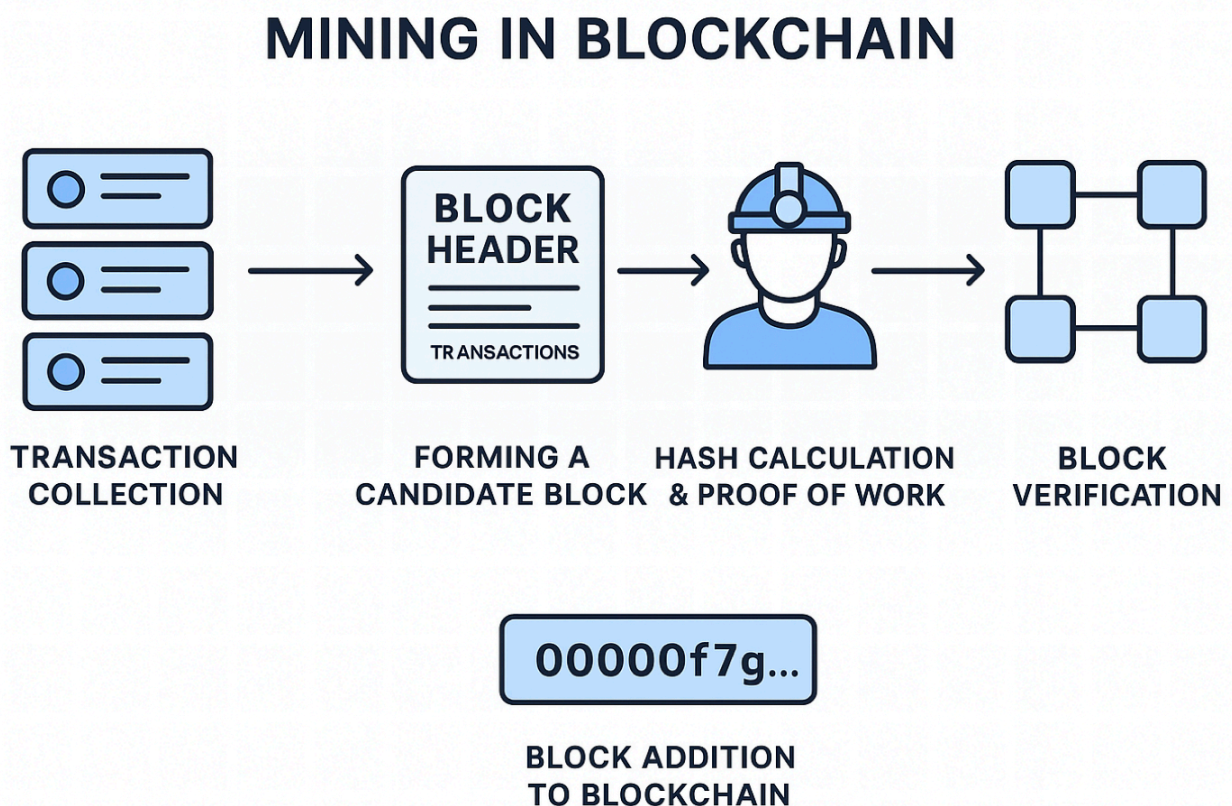


Mining is the process of validating transactions and adding them to a blockchain. It ensures the integrity and security of the network by solving complex computational problems. Miners are rewarded (usually in cryptocurrency) for successfully mining a block.

Mining serves three main purposes:

1. **1.Validation of Transactions:** Ensures that all transactions in the network are legitimate.
2. **2.Creating New Blocks:** Adds verified transactions into the blockchain.
3. **3.Network Security:** Prevents double spending and fraudulent activities.

Process of Creating a New Block (Block Mining)



The process generally involves the following steps:

- **Transaction Collection:**
 - All pending transactions are collected from the network's memory pool.
- **Forming a Candidate Block:**
 - Transactions are grouped into a block with a **block header**, which contains:
 - Previous block hash

- Timestamp
- Nonce (a number miners adjust to find the valid hash)
- Root hash of transactions (Merkle root)
- **Hash Calculation & Proof of Work (PoW):**
 - Miners repeatedly calculate the hash of the block header.
 - The goal is to find a hash that meets the **difficulty target** (e.g., a hash starting with a certain number of zeros).
 - This process is computationally intensive and requires trial and error.
- **Block Verification:**
 - Once a valid hash is found, the miner broadcasts the new block to the network.
 - Other nodes verify that all transactions in the block are valid and the hash meets the required difficulty.
- **Block Addition to Blockchain:**
 - After verification, the new block is added to the blockchain.
 - Miners receive a reward (cryptocurrency) and transaction fees.

Mining in Proof of Work (PoW) Systems

- **Mining** is the process of validating transactions and adding them to a blockchain by solving complex cryptographic puzzles. It is a key component of **Proof of Work (PoW)** consensus mechanisms, used by cryptocurrencies like Bitcoin.
- Mining in PoW systems **rewards miners with coins and transaction fees**, while **securing the blockchain, preventing fraud, and maintaining decentralization**. Without mining, the PoW blockchain would collapse, as there would be no mechanism to validate transactions or maintain trust.

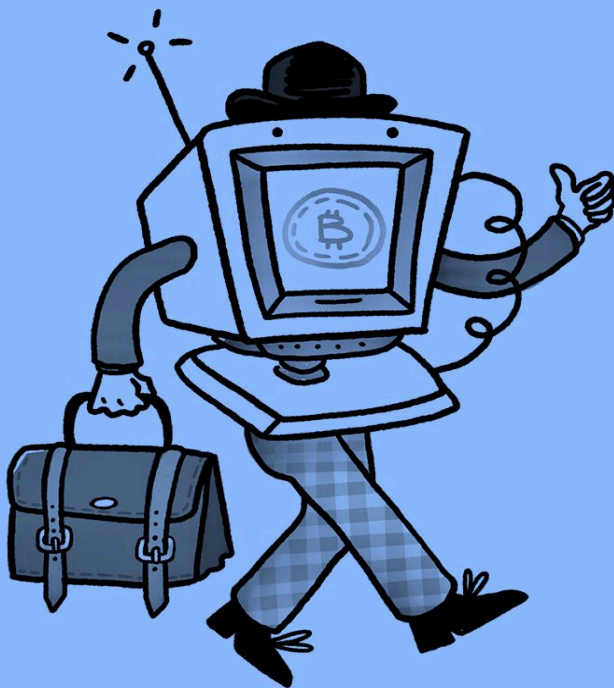
Rewards of Mining

1. **Block Rewards (New Coins)**
 - Miners are rewarded with newly created cryptocurrency for successfully mining a block.
 - Example: In Bitcoin, miners currently receive a fixed number of bitcoins for each block mined (this reward halves approximately every 4 years in an event called the *halving*).
1. **Transaction Fees**
 - Miners also earn fees attached to the transactions included in the block.
 - Each transaction can carry a small fee, which is collected by the miner who adds the block to the chain.
 - Example: If a block contains 2,000 transactions with an average fee of 0.0001 BTC, the miner earns 0.2 BTC in fees, in addition to the block reward.
1. **Network Incentives**
 - Mining rewards incentivize miners to participate in maintaining the blockchain, making the system decentralized and secure.

Significance of Mining in PoW

- **Transaction Verification**
 - Mining ensures that all transactions are valid, preventing double spending (where someone tries to spend the same coin twice).
- **Security and Trust**
 - The PoW puzzle requires computational effort, making it costly and difficult for any malicious actor to alter the blockchain.
 - To change a block, an attacker would need more than 50% of the network's computing power, which is practically infeasible for large networks.
- **Decentralization**
 - Mining allows anyone with computing power to participate in validating transactions, avoiding reliance on a central authority like a bank.
- **Consensus Achievement**
 - Mining helps the network agree on a single version of the blockchain, resolving conflicts and maintaining consistency across nodes.
- **Blockchain Integrity**
 - Each mined block contains a hash of the previous block, linking the chain. Mining ensures that the chain is tamper-resistant and chronological.

Proof of Work (PoW) Consensus Mechanism



Proof of Work (PoW)

['prʊf əv 'wɜrk]

A blockchain consensus mechanism in which computing power is used to verify cryptocurrency transactions and add them to the blockchain.

Proof of Work (PoW) is a **consensus mechanism** used in blockchain networks where miners solve complex mathematical puzzles to validate transactions and add new blocks to the blockchain. It ensures that the network agrees on a single version of the blockchain without a central authority.

How it works:

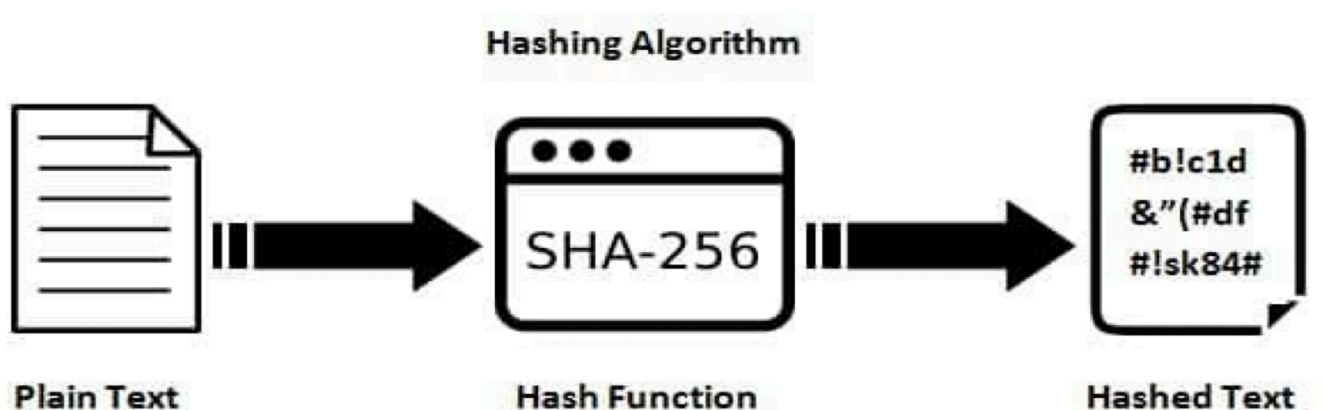
1. Transactions are broadcast to the network.
2. Miners collect these transactions into a block.
3. Miners compete to solve a **cryptographic puzzle** (finding a nonce such that the block hash meets a specific difficulty).
4. The first miner to solve the puzzle broadcasts the block to the network.
5. Other nodes verify the solution and add the block to their copy of the blockchain.
6. The miner receives **block rewards** and **transaction fees**.

How PoW prevents double spending:

- **Double spending** is attempting to spend the same cryptocurrency twice.
- PoW prevents this because:
 - Once a block is added, changing it would require redoing the PoW for that block **and all subsequent blocks**.
 - This requires enormous computational power (more than 50% of the network's total), making double spending practically impossible.
- Therefore, only one version of each transaction becomes valid and permanent in the blockchain.

SHA-256

SHA-256 (**Secure Hash Algorithm 256-bit**) is a cryptographic hash function from the SHA-2 family, developed by the NSA. It takes an input of any size (text, file, or data) and generates a **fixed 256-bit (32-byte) hash**, represented as a string of hexadecimal characters



Key Properties:

1. **Deterministic:** Same input → same hash.
2. **One-way function:** Cannot reverse the hash to get the original data.
3. **Collision-resistant:** Extremely unlikely for two different inputs to produce the same hash.
4. **Avalanche effect:** A small change in input drastically changes the hash.

Role in Proof of Work (PoW) and Blockchain Security

1. Mining and PoW:

- In Proof of Work systems, miners compete to solve a puzzle: finding a **nonce** so that the SHA-256 hash of the block header meets a difficulty target.
- The first miner to find a valid hash adds the block to the blockchain and receives rewards.
- SHA-256 ensures that this puzzle is **computationally difficult to solve but easy for others to verify**.

2. Overall Significance:

- SHA-256 underpins the security, trust, and reliability of PoW-based blockchains by ensuring data integrity, network consensus, and resistance to attacks.

3. Ensuring Blockchain Security:

- **Immutability:** Each block contains the hash of the previous block. Changing any block would require recalculating SHA-256 hashes for all subsequent blocks, which is practically impossible.
- **Transaction Integrity:** SHA-256 ensures that transaction data cannot be altered without detection.
- **Protection Against Double Spending:** Tampering with a confirmed block would break the chain and invalidate the altered transactions.

In short: SHA-256 is the cryptographic “engine” that makes Proof of Work secure, ensures block immutability, and prevents fraud like double spending

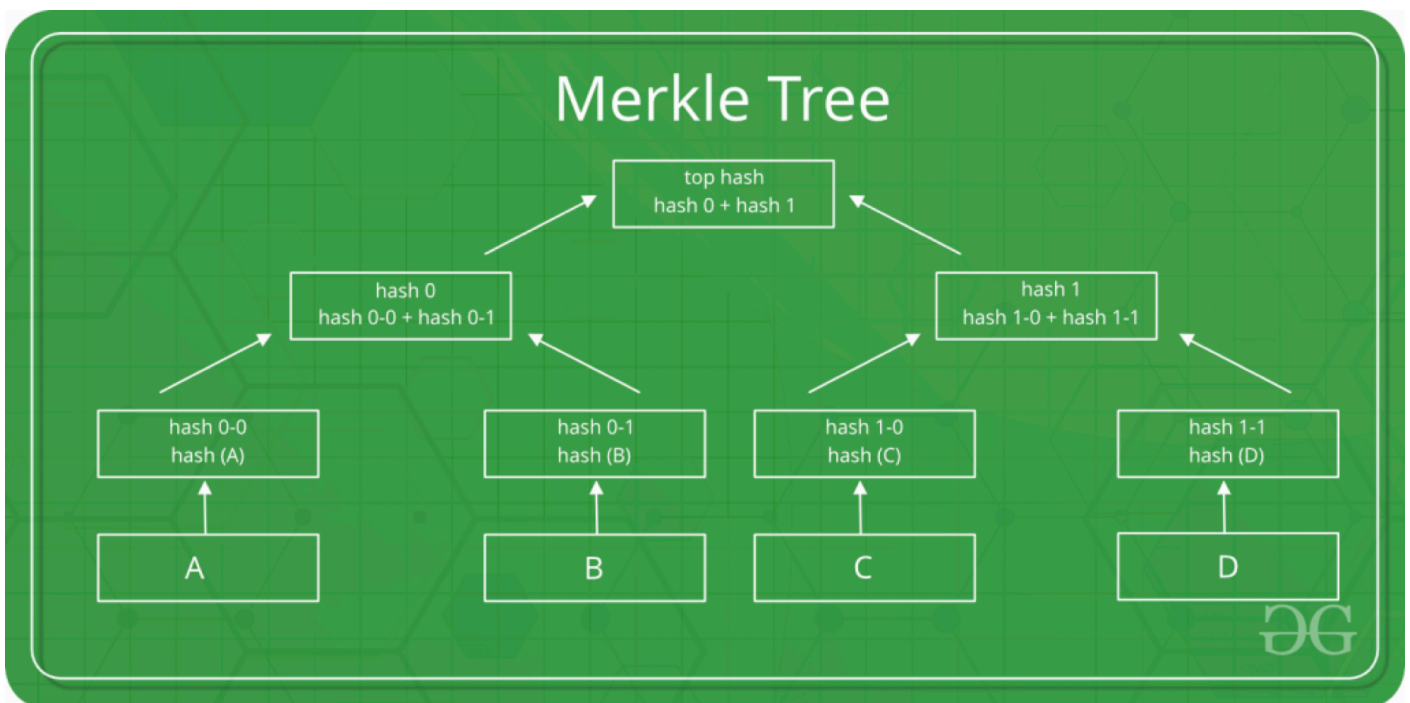
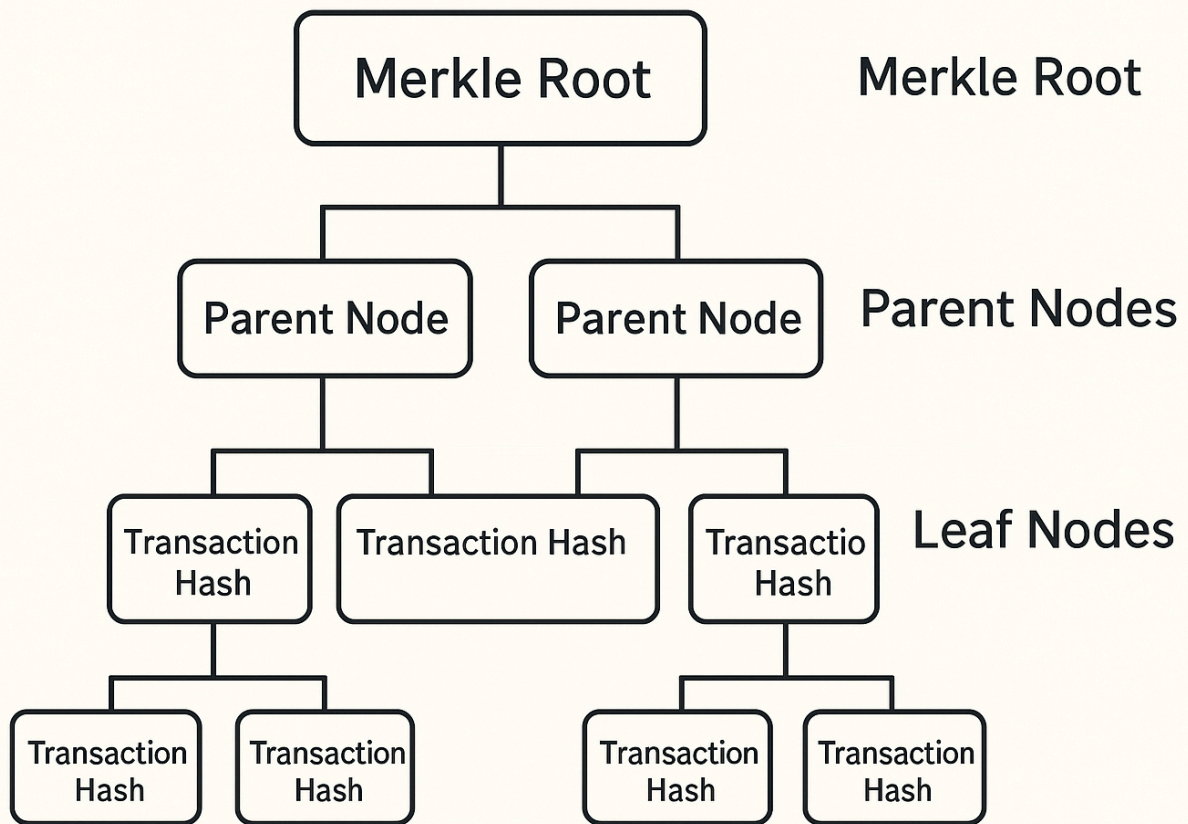
Merkle Tree



- A Merkle tree, or hash tree, is a cryptographic data structure that efficiently verifies the integrity and contents of a large data set
- Named for its inventor, computer scientist Ralph Merkle
- It organizes data into a hierarchical, tree-like structure
- Each pair of data blocks is **hashed together**.
- Hashing continues until one final hash, called the **Merkle root**, is formed.
- The Merkle root represents the **entire dataset securely**.
- Merkle Trees structure transaction data into a binary tree of cryptographic hashes, with the single, final "Merkle root" acting as a digital fingerprint for an entire block of transactions.
- Their importance in blockchain lies in enabling efficient, tamper-proof verification of data integrity and transaction inclusion without needing to download or process the entire dataset, enhancing network scalability and security by allowing light-client nodes to use Merkle proofs.

Structure

1. **Leaf Nodes** : The process starts with individual transaction hashes, which become the leaf nodes at the bottom of the tree.
2. **Parent Nodes** : These leaf nodes are then paired up, and a new hash is created by combining the hashes of each pair.
3. **Hierarchical Hashing**: This process continues, creating parent nodes from child nodes until only one hash remains.
4. **Merkle Root** : This final, top-level hash is the Merkle root, a compact summary of all transactions within the block.



Importance

- **Data Integrity** : The Merkle root acts as a single, unique fingerprint for the entire block's data. If even a tiny piece of data within the block is altered, the leaf node hash will change, leading to a different Merkle root, which the network can quickly detect.
- **Efficient Verification**: Instead of checking every single transaction in a large block, nodes can use a Merkle proof to verify if a specific transaction is included in the block. This involves just the hash of the transaction and the hashes of its sibling nodes, significantly reducing the amount of data needed to verify.
- **Scalability**: This verification method supports light clients and allows them to function on the blockchain without downloading the entire ledger, which improves network scalability.
- **Security**: By creating a tamper-evident summary of data, Merkle trees enhance the overall security and trustworthiness of a decentralized network, as they make it nearly impossible to alter data without detection.
- **Scalability**: This verification method supports light clients and allows them to function on the blockchain without downloading the entire ledger, which improves network scalability.
- **Security**: By creating a tamper-evident summary of data, Merkle trees enhance the overall security and trustworthiness of a decentralized network, as they make it nearly impossible to alter data without detection.

example

Consider the following scenario: A, B, C, and D are four transactions, all executed on the same block. Each transaction is then hashed, leaving you with:

- Hash A
- Hash B
- Hash C
- Hash D

The hashes are paired together, resulting in:

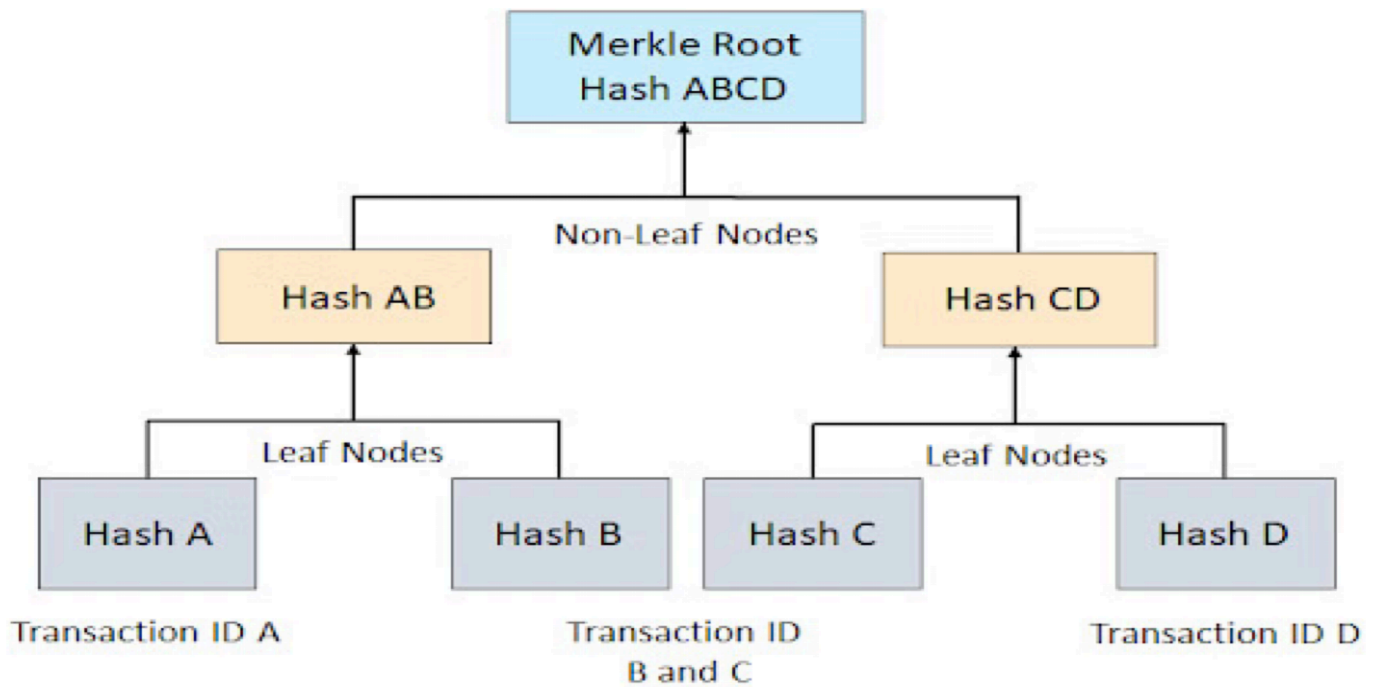
- Hash AB

and

- Hash CD

And therefore, your Merkle Root is formed by combining these two hashes: Hash ABCD.

Example



SPV (Simplified Payment Verification)



- **SPV (Simplified Payment Verification)** is a process allowing light clients to verify a transaction's presence in a blockchain without downloading the entire blockchain. It achieves this by using a Merkle Tree to create Merkle proofs.

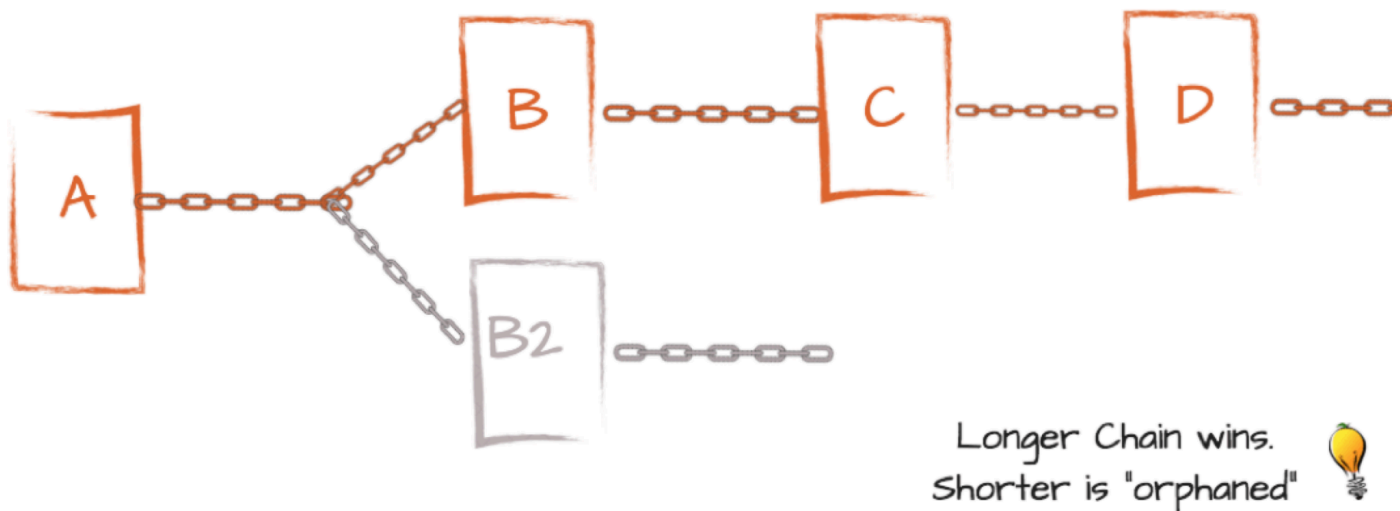
- A Merkle Tree is a cryptographic hash tree where each block of transactions is hashed, and these hashes are then combined and hashed up to a single Merkle root.
- SPV clients download only block headers and request a Merkle proof for a specific transaction, which includes the Merkle root, allowing them to efficiently confirm inclusion in the block.

How SPV and Merkle Trees Work Together

- 1. Block Headers:** SPV clients download only the block headers of the blockchain, which are much smaller than the full blockchain and contain the Merkle root for each block.
- 2. Merkle Trees:** Inside each block, a Merkle tree summarizes all the transactions. The leaves of the tree are hashes of the individual transactions, and each parent node is a hash of its children.
- 3. Merkle Root:** The top-level node of the Merkle tree is the Merkle root, which serves as a single hash representing all transactions in the block.
- 4. Merkle Proofs:** When a user wants to verify a transaction, the SPV client requests a Merkle proof from a full node. This proof is a Merkle branch, a path of hashes from the transaction's hash to the block's Merkle root.
- 5. Verification:** The SPV client then hashes the transaction along with the provided Merkle branch and compares the resulting Merkle root to the one in its downloaded block header. If they match, it confirms the transaction is included in that block.
- 6. Longest Chain Confirmation:** The client also checks the block headers on the longest chain. By verifying that more blocks have been built on top of the transaction's block, the client confirms that the transaction is in the most valid, agreed-upon version of the blockchain.

HARD FORK VS. SOFT FORK: WHAT'S THE DIFFERENCE?

	Hard Fork	Soft Fork
Goal	A fundamental change to the blockchain protocol	An update to the existing protocol
Backward Compatibility	Not backward compatible	An update to the existing protocol
Chain Split	Yes, creates a new blockchain	No, retains the same blockchain
Upgrade Process	Requires nodes to upgrade to remain compatible.	Nodes can still function even if not upgraded.
Consensus Requirement	In Bitcoin, requires consent from all node runners, and the majority of miners	Requires less consensus
Complexity	More complex to manage due to the chain split	Simpler to implement
Risk of Network Split	High, can lead to community division	Low



The **longest chain rule** means that if a blockchain splits into two different chains, the network will choose the one with the **most blocks** (the longest) as the correct version.

- This helps all nodes stay on the same history of transactions and keeps the system consistent.
- Nodes continue working on the longest chain they know, while blocks on the shorter chains are dropped. This process is called a **chain reorganization**.

How it works:

- **Forks and Disagreements:** Sometimes two miners create blocks at the same time, causing a temporary split in the blockchain.
- **Competing Chains:** Each split shows a different, but still valid, transaction history, which may cause confusion about which one is correct.
- **The Longest Chain Rule:** All nodes follow the rule of building only on the longest chain. If a node sees two or more chains, it will continue adding blocks only to the longest one.
- **Conflict Resolution and Reorganization:** The longest chain becomes the winner, while the shorter ones are abandoned (called orphaned). Nodes that were on the shorter chain then move to the longest chain, so everyone ends up with the same history.

How it resolves conflicts:

- **Ensures Consensus:** The longest chain rule makes sure all participants agree on one common version of the transaction history.
- **Maintains Data Integrity:** It protects against fraud like double-spending, since an attacker would need more than half of the total computing power to create a longer fake chain.
- **Guides Transaction Confirmation:** Transactions on the longest chain are treated as valid and permanent. Any transactions on shorter chains are dropped and must be re-mined into the main chain.

How conflicts are resolved in Bitcoin blockchain

In Bitcoin, conflicts happen when the blockchain temporarily splits into two or more valid versions. This usually occurs if two miners find a valid block at nearly the same time.

To resolve such conflicts, Bitcoin uses the **Longest Chain Rule**:

- Every node in the network always considers the **longest chain** (the one with the most proof-of-work) as the correct one.
- Shorter chains are dropped, and any transactions in them that aren't in the longest chain are put back into the **mempool** to be mined again.

Example of a fork:

1. Fork Creation:

- Miner A and Miner B both solve a block (say Block 101) almost at the same time.
- Half the network hears about Miner A's block first, and the other half hears about Miner B's block.
- This creates a **fork**: two different versions of the blockchain exist temporarily.

2. Competing Chains:

- Both chains are valid, so miners keep working to add the next block (Block 102) on top of whichever block they received first.

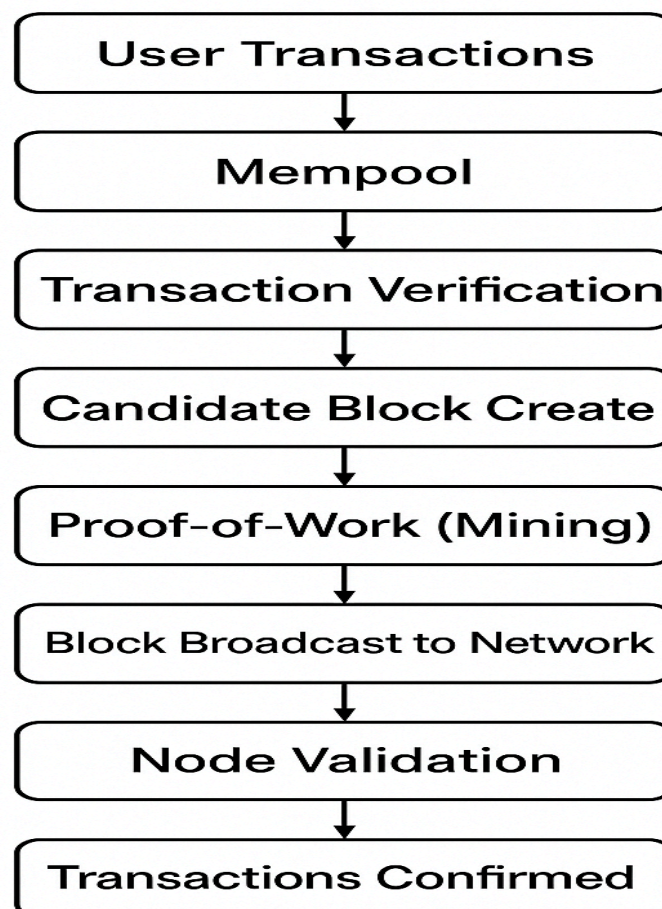
3. Resolution:

- Suppose Miner C adds the next block on top of Miner A's chain, making it longer.
- Now, all nodes recognize Miner A's chain as the **longest chain** and switch to it.
- Miner B's chain becomes an **orphan chain**, and the transactions in it (if not in Miner A's chain) return to the mempool.

4. Final Result:

- The fork is resolved, and the entire network agrees on one consistent history of transactions.

Steps Involved in Block Creation and Validation in Bitcoin



Steps Involved in Block Creation and Validation in Bitcoin

1. Transaction Initiation:

- Users broadcast transactions to the Bitcoin network.
- Transactions are collected by miners in a **memory pool (mempool)**.

2. Transaction Verification:

- Miners verify that each transaction is valid:
 - Sender has enough balance
 - Transaction is correctly signed with the sender's private key
 - No double-spending

3. Block Formation:

- Miners bundle verified transactions into a **candidate block**.
- The block contains:
 - List of transactions
 - Hash of the previous block
 - Timestamp
 - Nonce (for Proof-of-Work)

4. Proof-of-Work (Mining):

- Miners compete to solve a **cryptographic puzzle** by finding a nonce that makes the block's hash meet the network difficulty target.
- This requires intensive computation and ensures the block is hard to forge.

5. Block Broadcasting:

- The first miner to solve the puzzle broadcasts the new block to the network.

6. Block Validation by Nodes:

- Other nodes verify:
 - The block's hash meets the difficulty requirement
 - All transactions are valid
 - The previous block hash matches the current blockchain

7. Block Addition:

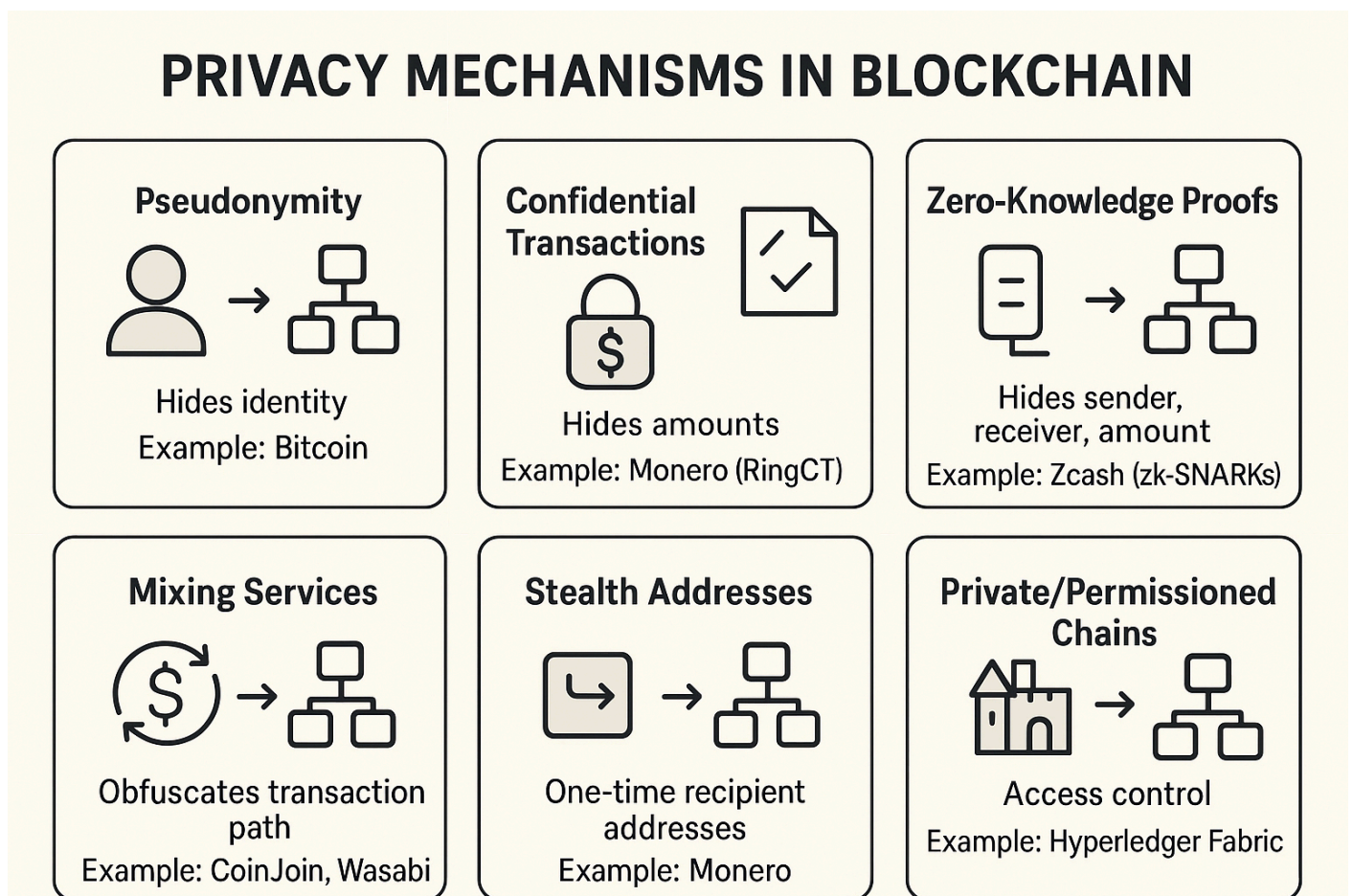
- If valid, nodes add the block to their copy of the blockchain.
- The miner receives the **block reward** (newly minted bitcoins + transaction fees).

8. Confirmation of Transactions:

- Transactions in the block are considered **confirmed**.
- Subsequent blocks added on top increase the confirmation count, making transactions more secure.

Privacy Mechanisms in Blockchain

Blockchain is inherently **transparent** because all transactions are recorded on a public ledger. However, users often require **privacy** to protect their identity, transaction details, or sensitive business information. Blockchain uses several mechanisms to enhance privacy.



1. Pseudonymity

- In blockchain, users operate using pseudonyms (usually alphanumeric addresses) instead of real identities.
- **Purpose:** Hides the real identity of participants.

- **Example:** Bitcoin allows users to transact using wallet addresses without revealing their real-world identity.
- **Note:** While pseudonymous, transactions can still be traced if the identity is linked elsewhere.

2. Confidential Transactions

- Techniques that hide the transaction amounts while keeping the network verifiable.
- **Purpose:** Protects financial privacy without revealing how much money is transferred.
- **Example:** Monero uses **Ring Confidential Transactions (RingCT)** to conceal the amount being sent.

3. Zero-Knowledge Proofs (ZKPs)

- Cryptographic method that allows one party to prove they know certain information without revealing the actual information.
- **Purpose:** Hides sender, receiver, and transaction amount.
- **Example:** Zcash uses **zk-SNARKs** to enable fully private transactions.

4. Mixing Services

- Services that mix multiple transactions together to obscure the transaction path.
- **Purpose:** Prevents tracking of transaction history.
- **Example:** CoinJoin and Wasabi Wallet combine transactions from multiple users to increase privacy.

5. Stealth Addresses

- One-time addresses generated for each transaction.
- **Purpose:** Ensures that each transaction has a unique recipient address, preventing linking of multiple payments to the same user.
- **Example:** Monero supports stealth addresses for enhanced privacy.

6. Private/Permissioned Chains

- Blockchain networks where access is restricted to authorized participants.
- **Purpose:** Provides controlled access and ensures sensitive data is shared only with authorized users.
- **Example:** Hyperledger Fabric is a permissioned blockchain used in enterprises.