

BLOCKCHAIN MINI TASK-01

UNDERSTANDING MERKLE TREE AND MERKLE ROOT



**SUBMITTED BY**

NAME- RUDRANARAYAN ROUT

DATE – JUNE 06, 2025

**OVERVIEW OF THE MINI PROJECT**

1. Provides a foundational understanding of blockchain technology.

2. Combines theoretical concepts with practical coding tasks.

3. Covers core topics:

* Blockchain structure
* Cryptographic hashing
* Merkle trees and data integrity
* Consensus mechanisms: PoW, PoS, DPoS

Implements:

* A simple blockchain simulation with linked blocks.
* A mining simulation to demonstrate Proof of Work.
* A validator selection demo for PoW, PoS, and DPoS.

4. Helps learners understand how blockchain ensures trust, resists tampering, and achieves decentralization.

5. Reinforces concepts through real-world code execution.

**OBJECTIVE OF THE MINI PROJECT**

* To **understand the core principles** of blockchain technology.
* To explore the **structure and components** of a block (data, hash, nonce, Merkle root, etc.).
* To gain insights into **different consensus mechanisms**: PoW, PoS, and DPoS.
* To simulate a **basic blockchain using code** with linked blocks.
* To demonstrate **mining through nonce calculation** and understand computational difficulty.
* To compare **validator selection** logic across PoW, PoS, and DPoS models.
* To strengthen learning through a **hands-on, practical approach** to blockchain concepts.

**THEORETICAL PART**

* BLOCKCHAIN BASICS

1. Describe the fundamentals of blockchain from your perspective.

A digital system called blockchain is used to safely and openly store data. It stores information in blocks, and each block is linked to the one before it by a unique code known as a hash, creating a chain. Data security is ensured by this configuration, which makes it extremely difficult to alter any information without impacting all subsequent blocks.  
Since blockchain technology is decentralized, no one entity or individual controls it. Rather, it operates on a network of computers, or nodes, that cooperate to validate and approve data using techniques like Proof of Stake and Proof of Work. This fosters trust without the need for an intermediary. Blockchain is frequently utilized for cryptocurrencies like Bitcoin, but it also aids with voting, supply chains, digital identity, and medical records.

1. Real life use cases of blockchain.

* **Supply Chain Management:** Blockchain ensures transparent and tamper-proof tracking of products from origin to consumer.
* **Digital Identity Verification:** Blockchain enables secure, user-controlled digital identity without central authorities.
* **Healthcare Records Management:** Blockchain provides secure and private sharing of patient medical records.
* **Financial Services & Payments:** Blockchain speeds up and secures cross-border payments while reducing fraud.
* **Voting Systems:** Blockchain creates transparent, tamper-proof digital voting for trustworthy elections.
* **Pharmaceutical industry**: Tracking drug production and distribution to prevent counterfeit medicines.
* **Food safety**: Monitoring food origin and handling to ensure freshness and compliance.
* **BLOCK ANATOMY**

1.Draw a block showing: data, previous hash, timestamp, nonce, and Merkle root.

|  |
| --- |
| **BLOCK** |
| **DATA** |
| **76344aB2er** |
| **TIME STAMP** |
| **NONCE** |
| **MERKLE ROOT** |

|  |
| --- |
| **BLOCK** |
| **DATA** |
| **PREVIOUS HASH** |
| **TIME STAMP** |
| **NONCE** |
| **MERKLE ROOT** |

|  |
| --- |
| **BLOCK** |
| **DATA** |
| **PREVIOUS Hasrt3** |
| **TIME STAMP** |
| **NONCE** |
| **MERKLE ROOT** |

StructureDescription (Block-by-Block):

* **BLOCK** – Label indicating a single block.
* **DATA** – Contains transaction or ledger information.
* **PREVIOUS HASH** – Stores the hash of the **entire previous block**, creating a link (chain) between blocks.
* **TIMESTAMP** – Indicates the exact time the block was created.
* **NONCE** – A random number used during the mining process (proof-of-work).
* **MERKLE ROOT** – A single hash representing all the transactions inside the block, generated using the Merkle Tree.

Flow Between Blocks:

* Left Block ➝ Middle Block ➝ Right Block
* Each arrow (➝) signifies that the current block includes the hashof the previous block in its header.
* This chaining ensures immutability: any tampering with an earlier block will invalidate all subsequent blocks because the hashes won’t match.

1. Briefly explain with an example how the Merkle root helps verify data integrity.

A Merkle root is like a digital signature for all the transactions in a block. It's just one hash value that represents everything inside the block.

* Merkle Root and Data Integrity Verification

1.Ensures that even a tiny change in data alters the root hash.

2.Allows efficient and secure verification of large data sets.

3.Supports lightweight clients to verify data without full download.

4.Protects against tampering by linking transaction hashes in pairs.

**EXAMPLES:**

|  |
| --- |
| **Hash function** |
| **Hash function** |
| **Hash function** |

|  |
| --- |
| **Dog** |
| **Flower** |
| **Dog plays with flower** |

|  |
| --- |
| df23yedfvcghkmfcdert |
| Ksnf520odjzn190okszm |
| Bo777sx56rtuui2ghykloz |

**Sequential Explanation:-**

* **Input Data(Transactions) :-** The inputs in the diagram like dog, flower, and Dog plays with flower represent individual transactions or data blocks in a blockchain.
* **Hash Function Applied :-**Each input is passed through a cryptographic hash function.

This converts the input into a fixed-length, unique hash, regardless of the size of the input.

E.g., dog → df23yedfvcghkmfcdert.

* **Leaf Nodes of Merkle Tree:-** The resulting hashes become the leaf nodes of the Merkle Tree, the foundational level of the tree structure.
* **Combining Hashes Up the Tree:-**Leaf hashes are paired and combined, then hashed again to form parent nodes.

This process continues recursively until a single hash remains at the top — the Merkle Root.

If there's an odd number of leaves, the last hash is duplicated before pairing.

* **Merkle Root = Proof of Integrity**

The final Merkle Root summarizes all the transactions in a block.

It is stored in the block header of the blockchain.

* **Integrity Verification**

If any transaction is altered, its hash changes.

This change propagates up the tree, resulting in a different Merkle Root.

Comparing the old and new Merkle Roots helps detect tampering instantly.

**CONSENSUS CONCEPTUALIZATION**

1.What is Proof of Work and why does it require energy?

Blockchain uses a consensus process called Proof of Work (PoW) to approve transactions and append new blocks to the chain. Participants (miners) must use processing power to solve challenging mathematical riddles. By making data alteration expensive and time-consuming, this procedure guarantees security. Because solving these riddles requires sophisticated machinery to operate constantly, energy is used. This energy-intensive procedure preserves the blockchain network's integrity and aids in the prevention of fraud.

2.What is Proof of Stake and how does it differ?

The quantity of bitcoin that validators "stake"—or lock up as collateral—determines which of them are selected to produce new blocks under the Proof of Stake (PoS) consensus process. It doesn't need figuring out difficult puzzles or using a lot of energy, as contrast to Proof of Work. Rather, validators are chosen according to their stake and occasionally additional criteria, such as the length of time they have held it. Because of this, PoS is faster and uses less energy than PoW. It also promotes sustained network involvement and lessens the demand for costly hardware.

3.What is Delegated Proof Stake and how are validators are selected?

In the Delegated Proof of Stake (DPoS) consensus process, token owners chose a select few reliable delegates (validators) to produce and verify new blocks. The voting power of each token holder is correlated with the quantity of cryptocurrency they own. The quantity of votes that validators acquire from the community determines which ones are chosen. Although it depends on active participation and community confidence, this technique increases speed and efficiency.

**PRACTICAL PART (CODE-BASED TASKS)**

**1.Block Simulation in Code**

* **Block Class**: Represents each unit in the blockchain. Contains:
* index, timestamp, data, previous\_hash, nonce, and hash.
* **SHA-256 Hashing**: Used to ensure data integrity.
* Each block generates its unique hash based on content.
* Changing any data recalculates a new hash.
* **Chaining Blocks**:
* Each block stores the hash of the previous block.
* This forms a linked, tamper-evident structure.

**Tampering Simulation**

* When Block 1 is tampered (data changed), its hash also changes.
* Block 2 still holds the old hash of Block 1.
* Hence, Block 2 becomes **invalid** because the chain breaks.

**Goal and Outcome**

* This simulation shows how blockchain ensures data immutability.
* Even a small data change breaks the chain.
* Recomputing all following hashes is required to restore validity which mimics real blockchain systems.

**2**. **Nonce Mining Simulation**

**Proof of Work (PoW)**:

* A cryptographic technique that secures blockchain by requiring miners to do work.
* Prevents spamming and tampering.

**Nonce**:

* A number that miners change to find a hash that satisfies the difficulty condition.
* The only variable in mining; rest of the block data remains constant.

**Difficulty**:

* Higher difficulty (more zeroes) = more time and attempts needed.
* Simulates increased security and energy demand in real blockchains.

**Hash Functions (SHA-256)**:

* Produces a fixed-length hash for any input.
* Even a small change in input drastically changes the hash (avalanche effect).

**Results**

* At difficulty 4 ("0000" prefix), mining may take hundreds or thousands of nonce attempts.
* Time increases exponentially with difficulty.
* Mining is time-consuming and computationally expensive, just like in real systems.

**Goal & Learning Outcome**

* Demonstrated that **Proof-of-Work requires real computation**.
* Higher difficulty increases:
  + Number of hash attempts.
  + Time and CPU usage.
* Learned how **blockchain security** relies on this costly process to prevent tampering and spam.

3. **Consensus Mechanism Simulation**

**How the Simulation Works:**

**Proof of Work (PoW):**

* Each miner is assigned a **random "power"** (represents computational strength).
* The one with the highest power is selected.
* Simulates mining with strong hardware winning the block.

**Proof of Stake (PoS):**

* Each validator is given a random stake (e.g., 100 to 1000 coins).
* The one with the largest stake is selected to propose the block.
* Encourages large stakeholders to be honest.

**Delegated PoS (DPoS):**

* A few validators (delegates) are voted by users.
* Each of 3 mock voters randomly votes.
* The delegate with the most votes wins.
* Simulates a democratic selection among limited representatives

**Conclusion**

* PoW rewards computational effort.
* PoS rewards economic commitment (stake).
* DPoS introduces **voting** and faster block selection.
* Each consensus mechanism affects speed, fairness, energy use, and decentralization differently.