Unit 2

1. Structure of a Block

Each block in a blockchain consists of the following components:

- **Header:** Contains metadata about the block.
    - **Block Hash:** A unique identifier of the block, created using a cryptographic hash function. This ensures that any alteration to the block's content results in a completely different hash, maintaining security and integrity.
    - **Previous Block Hash:** Links the current block to the previous one, creating a continuous and tamper-proof chain. This linkage is fundamental to the blockchain's immutability.
    - **Timestamp:** The time when the block was created. It helps maintain chronological order and ensures consistency across the blockchain.
    - **Nonce:** A number used during the mining process to solve the cryptographic puzzle. Miners iterate through different nonces to find a value that satisfies the hash requirement.
    - **Merkle Root:** A single hash that represents all transactions within the block. It is derived from the Merkle Tree structure and is crucial for efficient verification of transactions.
- **Body:** Contains the list of transactions or data.
    - **Transactions:** Each transaction is digitally signed by the sender and validated by the network before inclusion in a block. Transactions typically include details such as sender address, recipient address, amount, and digital signature.
    - The number of transactions per block depends on the blockchain's size, block size limit, and specific protocol rules (e.g., Bitcoin's block size limit of 1 MB).

2. Cryptographic Hash Functions

Cryptographic hash functions are fundamental to blockchain security and play a vital role in ensuring data integrity, security, and immutability. Key properties include:

- **Deterministic:** The same input always produces the same output, ensuring consistency.
- **Fixed Size:** Outputs are of a fixed length regardless of the size of the input data (e.g., 256 bits for SHA-256). This standardization ensures uniformity.
- **Pre-image Resistance:** It is computationally infeasible to derive the original input from its hash. This protects sensitive data.
- **Collision Resistance:** It is highly unlikely that two different inputs will produce the same hash. This ensures the uniqueness of data.
- **Avalanche Effect:** A minor change in input results in a drastically different hash, adding to the security and unpredictability.

Use in blockchain:

- Hash functions are used to link blocks securely by including the "Previous Block Hash."
- Proof of Work (PoW) involves solving hash-based puzzles to validate new blocks.
- Hashes ensure transaction integrity and prevent unauthorized modifications.

Popular Hash Functions:

- **SHA-256:** Used in Bitcoin and other cryptocurrencies.
- **Keccak-256:** Used in Ethereum.

3. Merkle Trees

Merkle trees are hierarchical data structures that enable efficient and secure verification of large amounts of data, such as blockchain transactions. Key features include:

- **Structure Transactions:** Each transaction is hashed, and pairs of transaction hashes are combined and hashed repeatedly to form a binary tree structure.
    - At the base (leaf level), each leaf node represents the hash of a transaction.
    - At higher levels, each node represents the hash of its two child nodes.
- **Generate Merkle Root:** The top-most hash, known as the Merkle Root, summarizes all transactions within the block. It is stored in the block header.
- **Efficient Verification:** To verify the inclusion of a transaction, only a small portion of the tree (proof path) needs to be checked instead of all transactions. This significantly reduces computational effort.

Example:

- Transactions T1, T2, T3, T4 are hashed to form H1, H2, H3, H4.
- H1 and H2 combine to form H12; H3 and H4 combine to form H34.
- H12 and H34 combine to form the Merkle Root (H1234).

Benefits:

- Scalability: Handles large data efficiently.
- Security: Ensures data integrity and tamper-proof verification.

4. Blockchain Nodes: Miners, Validators, Full Nodes

Nodes are critical components of the blockchain network, as they maintain, validate, and propagate data. Types of nodes include:

- **Miners:**

    - Responsible for adding new blocks to the blockchain by solving cryptographic puzzles in Proof of Work (PoW) systems.
    - Compete to find a nonce that, when combined with other block data, produces a hash below a target value.
    - Receive rewards for their work, including block rewards (newly minted cryptocurrency) and transaction fees.
- **Validators:**
    - Essential in Proof of Stake (PoS) and similar consensus mechanisms.
    - Validate transactions and propose new blocks based on their staked cryptocurrency.
    - Validators' integrity is maintained by penalizing malicious behavior (slashing their staked assets).
- **Full Nodes:**
    - Store the entire blockchain history, including all blocks and transactions.
    - Independently validate transactions and blocks, ensuring network security and decentralization.

Other Types of Nodes:

- **Light Nodes:** Store only a portion of the blockchain data and rely on full nodes for transaction verification.
- **Master Nodes:** Perform specialized functions, such as governance, voting, and processing private transactions. Often require collateral.

5. Smart Contracts and DApps

- **Smart Contracts:**
    - Self-executing programs with predefined rules and conditions.
    - Stored and executed on the blockchain, ensuring transparency, security, and automation.
    - Common Features:
        - Immutable once deployed.
        - Trustless execution without intermediaries.
        - Examples: Escrow systems, supply chain tracking, token issuance.
    - Programming Languages: Solidity (Ethereum), Vyper, Rust (Solana).
- **Decentralized Applications (DApps):**
    - Applications built on blockchain platforms that operate on a peer-to-peer network.

- Combine front-end user interfaces with smart contracts on the back-end.
- Categories:
  - **DeFi (Decentralized Finance):** Lending platforms, decentralized exchanges.
  - **Gaming:** Blockchain-based games with tokenized assets.
  - **Social Media:** Decentralized platforms prioritizing user privacy.

6. Forks and Consensus Algorithms

Forks

- **Definition:** A fork occurs when the blockchain's state diverges into two separate paths due to changes in protocol or disagreements among participants.
- Types:
  - **Hard Fork:**
    - An irreversible split resulting in two separate blockchains with differing rules.
    - Requires all participants to upgrade their software to stay compatible.
    - Example: Ethereum and Ethereum Classic.
  - **Soft Fork:**
    - A backward-compatible update where only one chain remains valid.
    - Non-upgraded nodes can still process transactions but may face limitations.

Consensus Algorithms

Consensus mechanisms ensure agreement on the blockchain's state among distributed participants. Examples include:

1. **Proof of Work (PoW):**
   - Miners solve complex mathematical problems to add blocks.
   - Highly secure but energy-intensive and slower.
   - Used by Bitcoin and Ethereum (before Ethereum 2.0).
2. **Proof of Stake (PoS):**
   - Validators stake cryptocurrency to gain the right to propose and validate blocks.
   - Energy-efficient and scalable.
   - Used by Ethereum 2.0, Cardano, Polkadot.
3. **Delegated Proof of Stake (DPoS):**
   - Stakeholders vote for delegates who validate transactions and maintain the blockchain.

- Faster and more democratic.

4. **Proof of Authority (PoA):**
    - Validates blocks based on the identity and reputation of validators rather than staking or computational power.
    - Suitable for private and consortium blockchains.