Blockchain Technology

Short notes for B.Tech CSE / IT · BCA · MCA · M.Tech

Uncodemy Handwritten Notes · free for students

1. Introduction

- **Blockchain** = a distributed, append-only ledger of records (blocks) linked by cryptographic hashes and shared among many nodes.
- No single central authority. Nodes agree on the ledger using a **consensus** mechanism.
- First used by **Bitcoin** (whitepaper by Satoshi Nakamoto, 2008; network started in 2009).

➤ Key properties

1. **Decentralisation**: no central owner.
2. **Immutability**: past records are practically impossible to change.
3. **Transparency**: participants can see transactions (in public chains).
4. **Security**: cryptography protects data and identity.
5. **Consensus**: all honest nodes agree on one version.

➤ Centralised vs decentralised ledger

Point	Centralised (bank database)	Blockchain
Control	One organisation	Many nodes
Trust	Trust the owner	Trust maths and consensus
Single point of failure	Yes	No
Editing history	Admin can change	Practically immutable
Speed	Fast	Usually slower

➤ Evolution

- **Blockchain 1.0**: cryptocurrency (Bitcoin).
- **Blockchain 2.0**: smart contracts (Ethereum).
- **Blockchain 3.0**: wider applications (supply chain, health, government, identity).

2. Cryptography Basics

Hash function

- Turns any input into a fixed-size output. Bitcoin uses `SHA-256` (256-bit output).
- Properties: deterministic, fast, pre-image resistant, collision resistant, **avalanche effect** (small change → completely different hash).

Public key cryptography

- Each user has a **private key** (secret) and a **public key** (shared).
- A **digital signature** is made with the private key and checked with the public key.
- Bitcoin and Ethereum use **ECDSA** on the curve `secp256k1`.
- A **wallet address** is derived from the public key by hashing.

Merkle tree

- A binary tree of hashes. Leaves = hashes of transactions. Each parent = hash of its two children.
- The top hash = **Merkle root**, stored in the block header.



- Proves a transaction is in a block with only about `log2(n)` hashes (Merkle proof). Used by light (SPV) clients.

> **EXAMPLE** 8 transactions → log2(8) = 3 sibling hashes are needed to prove one transaction.

3. Block Structure

➤ Block header (Bitcoin)

Field	Meaning
Version	Rules version
Previous block hash	Links to the parent block
Merkle root	Summary of all transactions
Timestamp	Time of block creation
Difficulty target (bits)	How hard the PoW puzzle is
Nonce	Number changed by miners to find a valid hash

- **Block body** = list of transactions. First transaction = **coinbase** (miner's reward).
- **Genesis block** = the first block of a chain (block 0).

➤ Why the chain is tamper-evident

```
[Block 1] <-- prev hash -- [Block 2] <-- prev hash -- [Block 3]
```

- Changing data in Block 1 changes its hash, so Block 2's "previous hash" no longer matches, and every later block breaks.
- An attacker must redo the work for that block and every block after it, faster than the honest network.

> **REMEMBER** Hash links make tampering easy to detect. Proof of Work makes rewriting history very costly.

4. Consensus Mechanisms

- Consensus = how distributed nodes agree on the next block without trusting each other.

➤ Proof of Work (PoW)

- Miners try nonce values until the block hash is below the target.
- Hard to find, easy to verify. The longest chain (most total work) is accepted.
- Bitcoin aims for about 10 minutes per block and adjusts difficulty every 2016 blocks.
- Drawback: high energy use.

EXAMPLE If the target needs the hash to start with 4 hex zeros, the chance per try is (1/16)^4 = 1/65536, so about 65536 tries are expected.

➤ Other mechanisms

Mechanism	Who makes the next block	Example
Proof of Work	Miner who solves the hash puzzle	Bitcoin
Proof of Stake	Validator chosen by staked coins	Ethereum (after 2022)
Delegated PoS	Elected delegates	EOS
PBFT	Voting among known nodes, tolerates f faults with 3f + 1 nodes	Hyperledger Fabric (variants)
Proof of Authority	Approved, known validators	Private / test networks

- **51% attack**: a party with most of the mining power (or stake) can double-spend and rewrite recent blocks.
- **Fork**: chain splits. **Soft fork** = backward-compatible rule change. **Hard fork** = not compatible, old nodes reject new blocks.

5. Bitcoin

- **UTXO model**: balance = sum of unspent transaction outputs owned by your keys.
- A transaction uses old UTXOs as inputs and creates new outputs. Fee = inputs − outputs.

EXAMPLE Inputs 0.5 + 0.3 = 0.8 BTC. Outputs: 0.6 to receiver, 0.19 change back to sender = 0.79. Fee = 0.8 − 0.79 = 0.01 BTC.

- **Double spending**: spending the same coin twice. Prevented by consensus and confirmations (waiting for more blocks on top).
- Total supply is capped at 21 million BTC. The block reward **halves** every 210,000 blocks (about 4 years).
- **Bitcoin Script**: simple stack-based language, not Turing complete.
- Nodes: **full node** (stores and checks whole chain), **light / SPV node** (only headers), **miner**.

6. Ethereum and Smart Contracts

- **Ethereum** = blockchain with a Turing-complete virtual machine (**EVM**) to run smart contracts.
- **Smart contract** = program stored on the blockchain that runs automatically when conditions are met.
- Uses an **account model**: externally owned accounts (keys) and contract accounts (code).
- **Gas** = unit for computation. Fee = gas used × gas price. Prevents infinite loops and spam.
- Main language: **Solidity**. Tools: Remix IDE, MetaMask wallet, Hardhat, Truffle.

```
pragma solidity ^0.8.0;
contract Counter {
    uint public count;
    function increment() public {
        count += 1;
    }
}
```

➤ Tokens and DApps

- **ERC-20** = fungible tokens (all units equal). **ERC-721** = NFTs (each token unique).
- **DApp** = decentralised application: front end + smart contracts on chain.
- **DAO** = organisation run by rules in smart contracts and token voting.
- **Oracle** = service that brings outside data (prices, weather) into smart contracts.

EXAM POINT Smart contract code cannot be changed easily after deployment, so bugs are costly. Test and audit before deploying.

7. Types of Blockchain

Type	Who can join	Speed	Example
Public (permissionless)	Anyone	Slower	Bitcoin, Ethereum
Private (permissioned)	Invited members, one organisation	Fast	Internal company ledgers
Consortium	A group of organisations	Fast	Trade networks
Hybrid	Mix of public and private parts	Varies	Mixed setups

Hyperledger Fabric

- A permissioned blockchain framework (Linux Foundation project).
- **Chaincode** = smart contracts. **Channels** = private sub-ledgers between some members.
- Flow: execute → order → validate. Peers endorse, the ordering service orders, peers validate and commit.
- **MSP (Membership Service Provider)** manages identities and certificates.

8. Applications, Issues and Trends

Applications

- Cryptocurrency and cross-border payments
- Supply chain tracking (origin of goods)
- Healthcare records sharing
- Digital identity and certificates
- Land records, voting, insurance claims (smart contracts)
- DeFi (decentralised finance): lending, exchanges

Challenges

- **Scalability trilemma**: hard to get decentralisation, security and scalability together.
- Low transaction speed, high fees at busy times, energy use (PoW).
- Privacy on public chains, regulation, lost private keys mean lost funds.
- **Scaling solutions**: Layer 2 (Lightning Network, rollups), sharding, sidechains.

Wallets

- **Hot wallet**: online (apps, browser). **Cold wallet**: offline (hardware, paper). Cold is safer for storage.
- "Not your keys, not your coins": whoever holds the private key controls the funds.

9. Most Asked Questions (practice)

1. What is blockchain? Explain its key properties.
2. Explain the structure of a block and the block header fields.
3. What is a Merkle tree? How does it help light clients?
4. Explain Proof of Work. Compare PoW and PoS.

5. What is a 51% attack? How is double spending prevented?
6. Explain Bitcoin's UTXO model with an example.
7. What is a smart contract? Explain gas in Ethereum.
8. Public vs private vs consortium blockchain.
9. Explain the architecture of Hyperledger Fabric.
10. Applications and challenges of blockchain.

TIP Block structure, Merkle tree, PoW vs PoS and smart contracts carry most marks. Draw the chained-block and Merkle tree diagrams in every answer.