## Introduction to Blockchain

Blockchain is a **distributed and decentralized digital ledger** that records transactions across a network of computers in a way that ensures **security, transparency, and immutability**.
Once information is recorded in a blockchain, it **cannot be altered or deleted** without the agreement of the network.

---

### 🔹 Blockchain Types

| Type | Description | Example |
|---|---|---|
| **Public Blockchain** | Open to everyone, no permission required to join. Fully decentralized. | Bitcoin, Ethereum |
| **Private Blockchain** | Access restricted to authorized participants. Controlled by one organization. | Hyperledger Fabric |
| **Consortium / Federated Blockchain** | Controlled by a group of organizations rather than a single entity. | R3 Corda |
| **Hybrid Blockchain** | Combination of public + private blockchain features. | Dragonchain |

---

### 🔹 Public Key Cryptography

Uses two keys:

- **Public Key** – shared openly
- **Private Key** – kept secret
  Data encrypted with one key can only be decrypted with the other. Used for **security and authentication** in blockchain.

---

### 🔹 Hashing

Hashing converts data into a **fixed-length hash value** using algorithms like **SHA-256.**
Characteristics:

- One-way function (cannot retrieve original data)
- Small change → completely different hash
- Used to link blocks and ensure immutability

---

### 🔹 Digital Signature

A **cryptographic mechanism** used to verify:

- **Sender identity**
- **Integrity of the message**
  Created using the sender's **private key** and verified using **public key**, ensuring **non-repudiation.**

---

### 🔹 Business Networks in Blockchain

A blockchain business network consists of:

- **Organizations / participants**
- **Assets**
- **Ledgers**
- **Transactions**
- **Smart contracts**

### 📌 Key Terminology

| Term | Meaning |
|---|---|

| | |
|---|---|
| **Assets** | Items of value (digital or physical) tracked via blockchain |
| **Ledger** | Record of transactions |
| **Transactions** | Operations that change the state of an asset |
| **Smart Contracts** | Self-executing code that governs business rules |

### Problem with Existing Traditional Networks

| **Issues** |
|---|
| Centralized system – single point of failure |
| Lack of trust between organizations |
| Manual reconciliation |
| Time-consuming and costly |
| Vulnerable to tampering and fraud |

### How Blockchain Solves These Problems

| **Solution** | **Benefit** |
|---|---|
| Decentralization | Removes dependency on central authority |
| Immutability | No tampering of records |
| Transparency | Real-time visibility |
| Automation via smart contracts | Fast, error-free transactions |
| Cryptographic security | Data protection and authenticity |

### Requirements of Blockchain for Business

A blockchain business network must support:

- **Permissioned access**
- **High performance & scalability**
- **Better privacy and confidentiality**
- **Governance & compliance**
- **Interoperability with enterprise systems**

---

## 📌 Blockchain Networks

### 🔥 Overview of Active Networks and Use Cases

| Network | Purpose |
|---|---|
| **TradeLens** | Improving global supply chain and shipping efficiency |
| **IBM Food Trust** | Food safety and farm-to-table product transparency |
| **IBM World Wire** | Real-time international payments using blockchain |
| **Decentralised & Trusted Identity** | Secure digital identity without central authority |

---

### 📍 Examples of Blockchain by Industry

| Industry | Use Case |
|---|---|
| Banking | Cross-border payments, trade finance |
| Supply Chain | Shipment tracking, product traceability |
| Healthcare | Medical record management |
| Retail | Anti-counterfeit systems |
| Government | Land/property record digitalization |

| | |
|---|---|
| Education | Digital certificates |

---

## 🧠 Key Players in Blockchain Adoption

- **IBM**
- **Microsoft Azure Blockchain**
- **Amazon Managed Blockchain**
- **Oracle Blockchain Cloud**
- **R3**
- **ConsenSys (Ethereum Enterprise)**

---

## 📌 IBM and Blockchain

### 🔹 How IBM Helps with a Blockchain Project

IBM provides:

- Consulting and solution design
- Cloud-based blockchain tools
- Security, monitoring and scaling
- Production-grade infrastructure

### 🔹 IBM's Blockchain Strategy

- Focus on **enterprise-grade permissioned blockchain**
- Building **industry-specific solutions**
- Supporting **global consortiums / partnerships**

### 🔹 IBM Blockchain Platform

A **SaaS platform** that enables:

- Developing blockchain applications
- Deploying smart contracts
- Managing network participants
- Monitoring and scaling blockchain systems

---

### 🔥 The Linux Foundation’s Hyperledger Project

Hyperledger is an open-source collaborative project for **enterprise blockchain technologies.**

### 🔹 Hyperledger Fabric

Most widely used Hyperledger framework.
Characteristics:

- **Permissioned network**
- **Supports channels for private transactions**
- **Modular architecture**
- **Pluggable consensus**

Fabric is the foundation for many IBM blockchain solutions.

---

# Hyperledger Composer

## 🔹 What is Hyperledger Composer?

Hyperledger Composer is a **development framework and toolset** for building blockchain business networks quickly on **Hyperledger Fabric**.
It provides high-level abstractions that allow developers to:

- Model business networks
- Define assets, participants, and transactions
- Implement smart contracts (transaction logic)
- Interact with blockchain through REST APIs

⚠ Note: Hyperledger Composer is now **deprecated**, but it remains very important in academics for understanding blockchain modeling concepts.

---

## 🔹 Components and Structure of Composer

| Component | Purpose |
|---|---|
| **Model File (.cto)** | Defines business domain model — assets, participants, and transactions |
| **Script File (.js)** | Contains transaction logic written in JavaScript |
| **Access Control File (.acl)** | Specifies permissions and access rights |
| **Query File (.qry)** | Defines queries for the ledger |
| **Business Network Archive (.bna)** | Packaged deployable file of the entire business network |
| **Playground** | GUI to develop and test the network |
| **REST Server** | Auto-generates REST APIs to interact with blockchain |

### ✔ Structure of a Typical Composer Business Network

```
my-network/
 ├─ models/
 │   └─ org.example.auction.cto
 ├─ lib/
 │   └─ logic.js
 ├─ permissions.acl
 ├─ queries.qry
 └─ package.json
```

---

## 🔹 Example Business Network — Car Auction Market

The **Car Auction Network** demonstrates how Composer models real-world scenarios.

| Element | Example |
|---|---|
| **Participants** | Buyers, Sellers, Auctioneers |
| **Assets** | Cars, Auction Listings |
| **Transactions** | PlaceBid, CloseAuction |
| **Smart Logic** | Highest bid wins when auction ends |

Workflow:

1. Seller lists a car for auction.
2. Buyers place bids.
3. Auctioneer closes auction.
4. Smart contract transfers asset ownership to highest bidder.

This illustrates **automation, transparency, and trust** using blockchain.

---

## 🔹 Extensive, Familiar, Open Tool Set (Advantages of Composer)

| Feature | Benefit |
|---|---|
| Extensive Tooling | Fast development and deployment |

| | |
|---|---|
| Familiar Languages | Modeling in JSON/DSL, logic in JavaScript |
| Open Source | Free, community supported |
| Integration Ready | Auto-generated REST API and Angular generator |

---

# 📌 Blockchain Fabric Development

Hyperledger Fabric is a **permissioned blockchain platform** designed for business networks.

---

## 🔹 Participants & Components Overview in Fabric

| Component | Role |
|---|---|
| **Clients (Applications)** | Submit transaction requests |
| **Peers** | Maintain ledger + smart contracts |
| **Orderers (Ordering Service)** | Order transactions and form blocks |
| **MSP (Membership Service Provider)** | Identity management and authentication |
| **CA (Certificate Authority)** | Issues digital certificates |
| **Channels** | Private communication groups for selective members |
| **Chaincode** | Smart contract logic written in Go/Node.js/Java |

---

## 🔹 Developer Considerations

A Fabric developer must decide:

- Data model (assets, participants)

- Smart contract language (Chaincode)
- Use of channels for privacy
- Transaction endorsement policy
- Integration with existing systems via SDK

---

## 🔹 Blockchain Architecture (Fabric)

Fabric architecture includes:

```
Application → Client SDK → Endorsing Peers → Ordering Service
→ Committing Peers → Ledger
```

Ledger contains:

- **World State (current value of assets)**
- **Blockchain (complete transaction history)**

---

## 🔹 Administrator (Operator) Considerations

Admins manage:

- Network setup, channel creation
- Security rules and policy enforcement
- Identity/permission management using MSP
- Node deployment and scaling
- Monitoring transactions and logs

---

## 🔹 Security: Public vs Private Blockchains

| Feature | Public | Private |
|---|---|---|
| Access | Open to anyone | Permission-based |
| Consensus | Mining/PoW/PoS | Efficient, BFT/RAFT |
| Speed | Slow | Fast, scalable |
| Privacy | Low | High |
| Use Case | Cryptocurrency | Business applications |

Hyperledger Fabric belongs to **Private/Permissioned Blockchain**.

---

## 🔹 Architect Considerations

A blockchain architect must plan:

- Governance rules and identity model
- Ledger structure and smart contract logic
- Channel configuration and privacy strategy
- Scalability and deployment model (cloud/on-premise)
- Performance and consensus mechanism

---

## 🔹 Network Consensus Considerations

Consensus ensures **agreement on valid transactions**.

In Fabric, consensus consists of:

1. **Endorsement Phase –** Peers simulate and sign transaction proposal

2. **Ordering Phase –** Orderers sequence transactions into blocks

3. **Validation Phase –** Peers verify endorsements and update ledger

Fabric supports pluggable consensus:

| Algorithm | Characteristics |
|---|---|
| **RAFT** | Leader-based, high performance |
| **Kafka** | Crash-fault tolerant message ordering |
| **PBFT (conceptual)** | Byzantine fault tolerant, high trust |