UNIT-V

Hyperledger Blockchain Implementation: Introduction, Use Case Car Ownership Tracking, Hyperledger Fabric, Hyperledger Fabric Transaction Flow, FabCar Use Case Implementation, Invoking Chaincode Functions Using Client Application.
Advanced Concepts in Blockchain: Introduction, Inter Planetary File System (IPFS), Zero-Knowledge Proofs, Oracles, Self-Sovereign Identity, Blockchain with IoT and AI/ML Quantum Computing and Blockchain, Initial Coin Offering, Blockchain Cloud Offerings, Blockchain and its Future Potential.

Car Ownership Tracking Use Case

Introduction: -

Here, we will discuss about **vehicle identification use case** and how this solution can be architected. We have chosen a multi-actor network for vehicle identification.

One of the key problems in vehicle business is the missing information about ownership and the complete history of the vehicle. This result in losses for customers and at times legal issues when the ownership of the vehicle changes multiple times.

Consumers are now increasingly looking for transparency and full traceability to make sure that the car they purchase does not have any legal complications and is indeed genuine.

We will consider a **Fabcar network** that will track the ownership of the car, starting from the manufacture till it is sold to various customers. An actual network shall involve multiple players such as car manufacturer, distributor, dealer, agent, road transport authorities, service center, auto insurer, customer, etc.

Let us now go through the fitment assessment.

i)Use Case Description:

The following are the steps that have been identified for Car Ownership Tracking use case:

1. Workflow:

a) Car manufacture can create a new car and send it to the distributor with the help of a transporter.
b) Manufacturer updates regional transport office with details on new car.
c) The distributor now splits the stock received and distributes the cars to various dealers.
d) The dealers sell the car to the customers.
e) The car is now registered at the transport office against the customer.
f) Customer can approach the service center for servicing the car.
g) Customer can sell the car to another customer.
h) Ownership changes are approved and updated in transport office records.

2 Benefits for players:

(a) Availability of complete journey of the car, from the manufacturer till the customer, helps in reducing audit costs for manufacturers, distributors, and dealers.
(b) The Audit Trail can simplify loan and insurance processes.
(c) Helps in quicker recall in case of any faulty equipment that gets discovered at a later stage.
(d) Customers get to make informed decision regarding the car they are purchasing, such as if it is coming from a source that fits their ethical framework. Customers are happy to pay more for this transparency.
(e) Customers buying a used car from another customer can be assured that the used car is genuine and free from any legal complications.
(f) All other participants benefit through value generated and extra money spent by customers.

ii)Blockchain Relevance Evaluation Framework:

1. Does use case involve sharing assets/valuables/data between multiple participants?

Yes. Multiple participants are involved who are not confined to a single organization. Car is the Asset that is shared between various participants.

2. Is there impact due to sharing/hiding information between participants?

Yes, intermediaries are involved. The car does not reach the customers directly. They are passed through intermediaries such as dealers and agents.
In this example, intermediaries can also be accommodated in the blockchain.
It should be noted that not all blockchain use cases focus on eliminating intermediaries. Some of the use cases may still involve intermediaries, but the reliance on intermediaries is reduced.

3. Does the solution require shared write access?

Yes. Car is the asset that moves through the supply chain, that is, the ownership of the car is Changed as it moves through the supply chain.

4. Can the solution work without delete?

Yes. As of today, car supply chain does not mandate delete of data. Also, delete of change of hands will not help build the transparency desired.

5. Is it required to store large amount of non-transactional data?

No. The properties of car can generally be digitized and stored on the blockchain. Any related documents shall be stored on a central database by the needed parties outside the blockchain.

6.Is it required that only a very small group or an entity needs all the control?

Yes. The supply chain functionality is controlled by the parties participating in the supply chain.

7. Does use case involve high-performance (millisecond) transactions?

No. This process spans through various participants over a period of few days to few weeks and does not demand any high-performance transactions.

Now, we know that the assessment revealed that it is a very good use case for blockchain Also, since a small group needs to control the network, unlike a public network such as Bitcoin, a permissioned blockchain such as Hyperledger Fabric would be a good fit for this use case.

iii)Identification of Actors, Roles and Permissions:

Actor	Role Permission
Car Manufacturer	Can create a new car and record it on blockchain. Can sell the car to the customer. Can trace the complete ownership history of the car.
Transporter	Can deliver the car to the distributor. Can trace the complete ownership history of the car.
Distributor	Can send the car to the Dealer. Can trace the complete ownership history of the car.
Transport Authority	Can approve car registration. Can approve car ownership changes.
Dealer	Can sell the car to the Customer. Can trace the complete ownership history of the car.
Agents	Can sell the car to the customer. Can trace the complete ownership history of the car.
Customer	Can buy or sell the car. Can trace the complete ownership history of the car.
Service Center	Can service the car. Can trace the complete ownership history of the car

iv)UML Diagrams:

For the sake of this example, we will consider only 3 three parties - car manufacturer, dealer, and customer. However, the solution approach and concepts can be extended to more number of actors as well. We will be utilizing UML diagrams to capture these details as shown below:

ACTIVITY DIAGRAM:

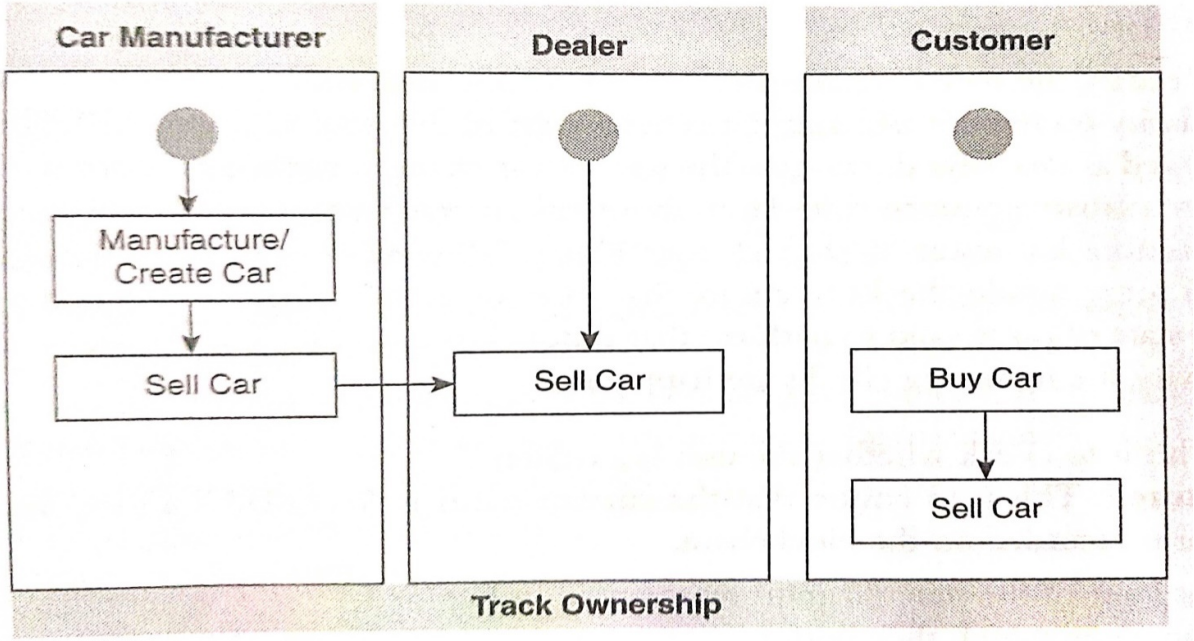


Figure 4.9 Car ownership tracking use case activity diagram.

SEQUENCE DIAGRAM:

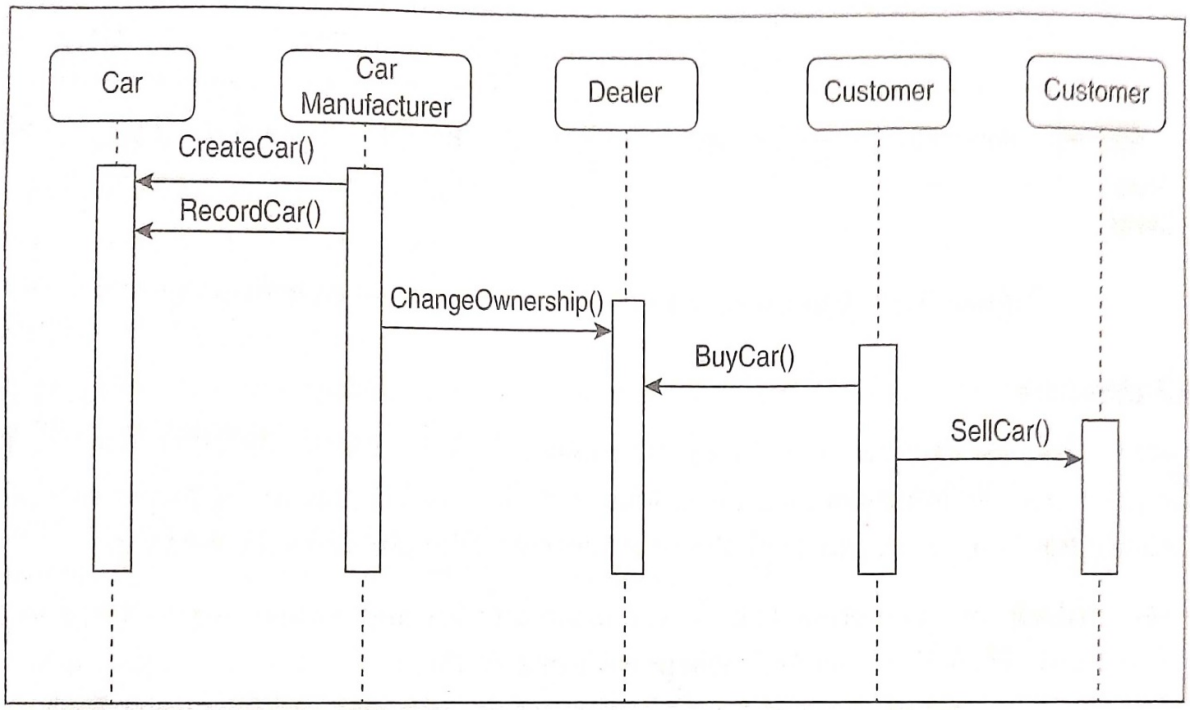


Figure 4.10 Car ownership tracking use case sequence diagram.

STATE DIAGRAM:

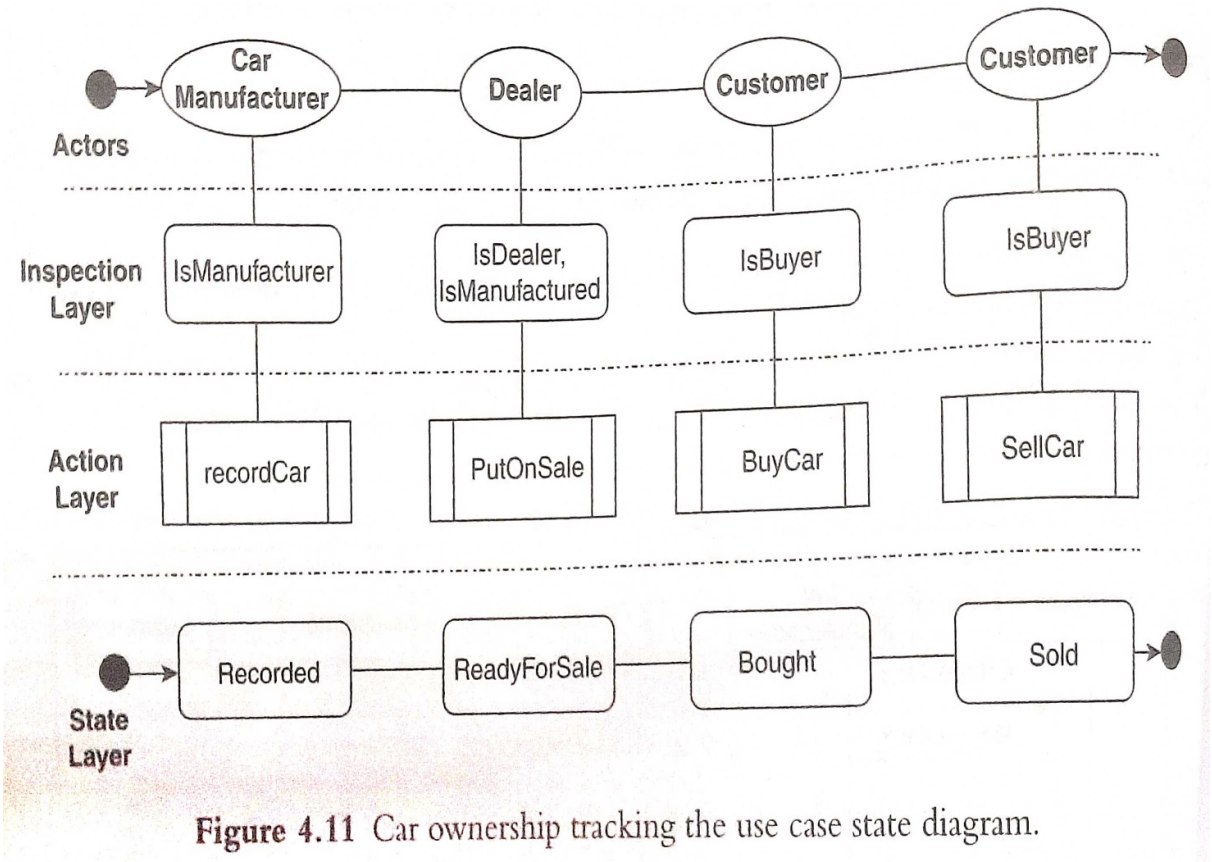


Figure 4.11 Car ownership tracking the use case state diagram.

Hyperledger Fabric

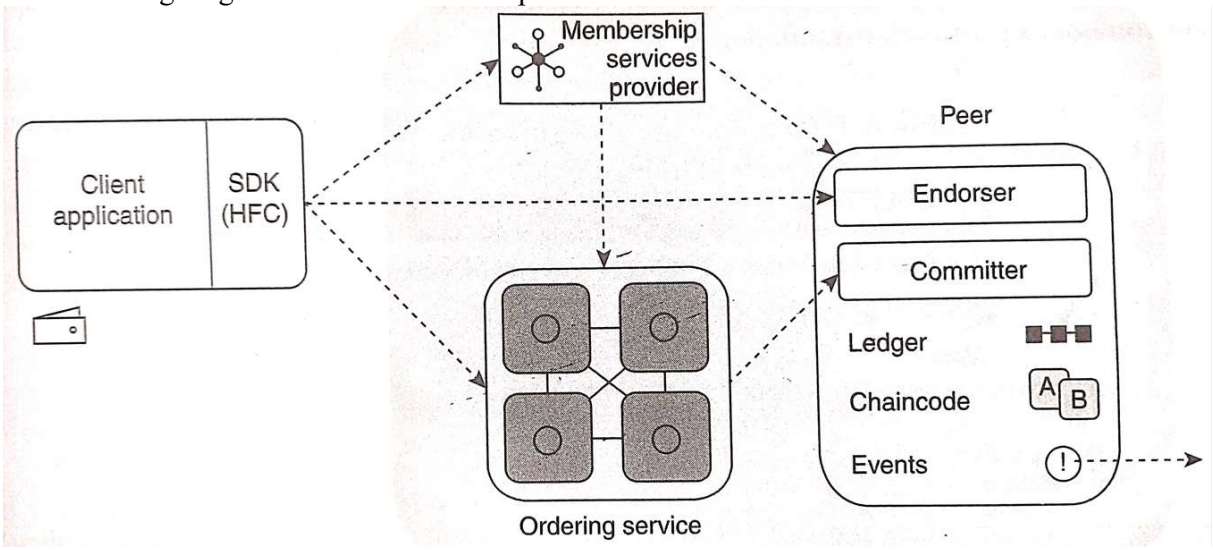
Introduction: -

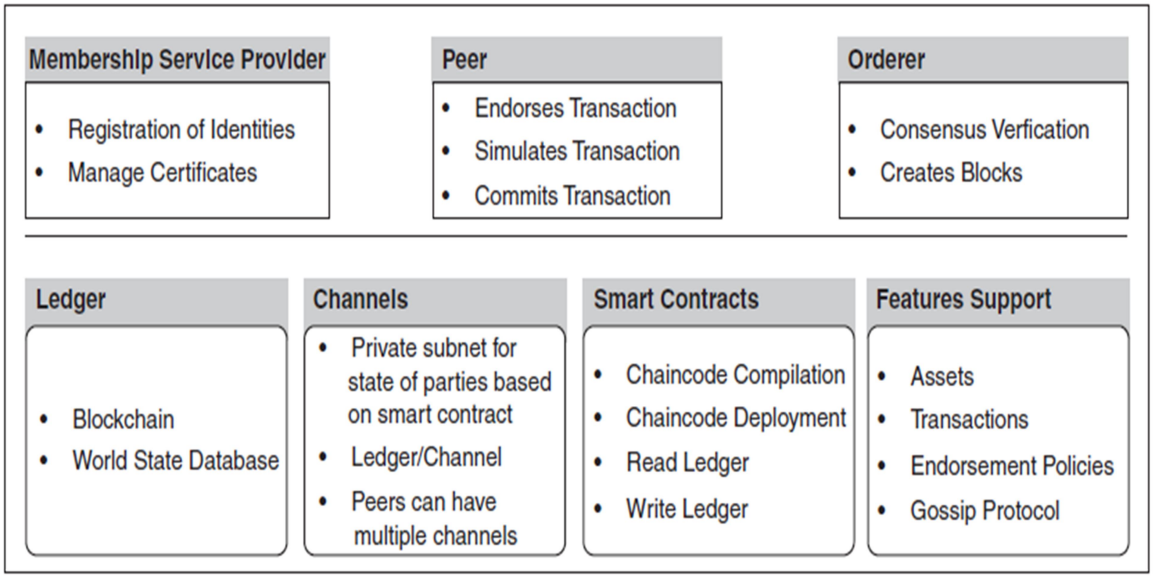
Hyperledger Fabric is an open source distributed ledger technology platform that is permissioned in nature and designed for use in enterprise contexts. It offers some key differentiating capabilities over other popular distributed ledger or blockchain platforms.

The design of Hyperledger Fabric allows adopting different consensus mechanisms and identity providers. It enables storing data in multiple formats within the ledger. Also, it allows participating entities to create separate communication channels between them for additional privacy. Hyperledger Fabric does not have any built-in token unlike Ethereum.

Before proceeding with implementation details, it would be ideal to have a look at the Hyperledger Fabric architecture components and transaction flow.

The following diagram shows various components in Fabric Architecture:





Hyperledger fabric components.

Membership Services Provider issues identity to each component within the network. A peer node can act as endorser node to endorse transactions or a committer node to commit the incoming blocks.

Each peer maintains a copy of ledger that includes chaincode as well. The ordering service receives incoming transactions from a client, orders them, creates a block, and broadcasts it to the peers.

A client application interacts with blockchain network through Application Programming Interfaces (APIs) provided by Hyperledger Fabric Software Development Kit (SDK). Peers may emit events to clients after committing a transaction.

There will be only one ordering service for the entire network which manages transaction ordering and channel specific broadcasting functionality.

Hyperledger Fabric Transaction Flow:-

Introduction: -

Adding an incoming transaction to a block and committing that block to blockchain on a Fabric network involves multiple steps as shown below:

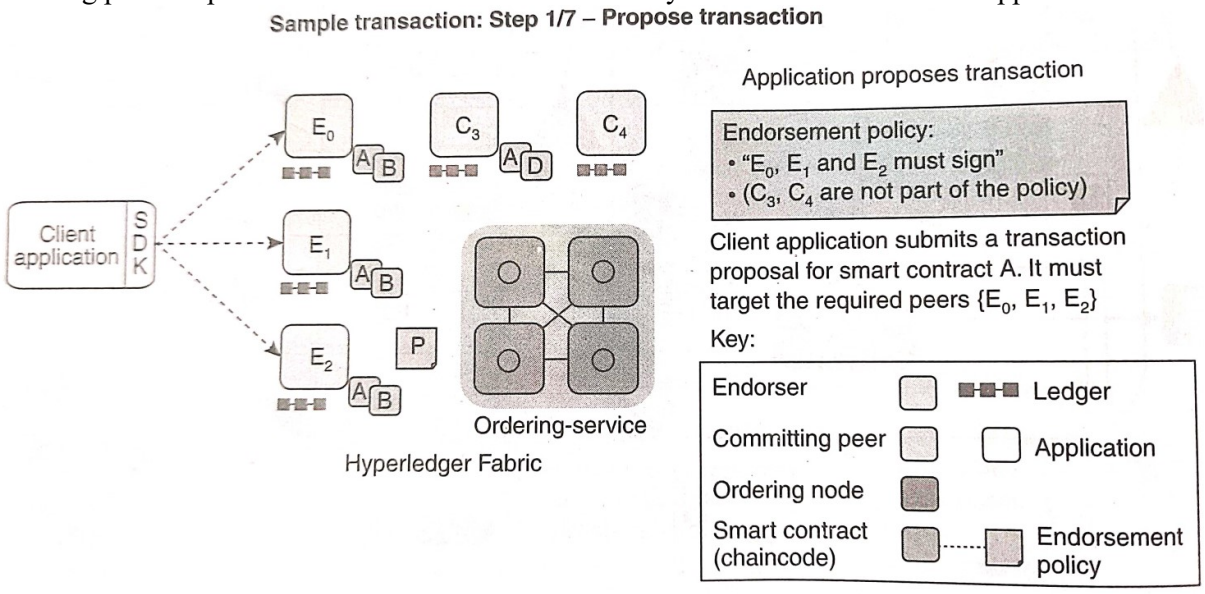
1. Proposing Transaction
2. Executing Proposal
3. Proposal Responses Receiving
4. Ordering Transaction
5. Deliver Transaction
6. Validating Transaction
7. Notify Transaction

1)Proposing Transaction: -

The first step is a client application (any frontend application or DApp) sending request to endorsement nodes in Fabric network.

For a transaction to be part of blockchain, it should meet the criterion defined in the endorsement policy of the network. So, there is no need to send the transaction to committing peers.

Client application can just send it to the endorsing peers. It is implicit that the details about endorsing peer endpoints or service API URLs are already available with the client application.



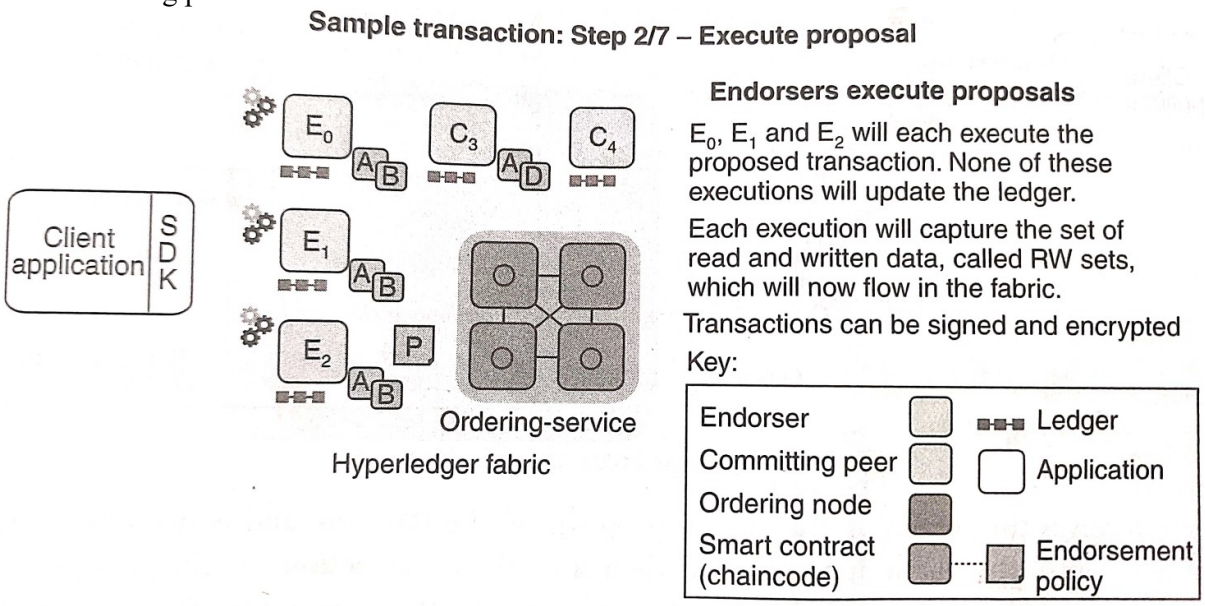
Fabric transaction flow – Step 1.

2)Executing Proposal: -

After receiving the transaction (proposal) from client application, each endorsing peer executes this proposal on its ledger. However, this proposal execution does not actually update anything on the ledger.

It just creates a set known as Read-Write (RW) set. This can represent a snapshot of before and after versions of the ledger because of the proposed transaction. This RW set will be used in further transaction execution processes.

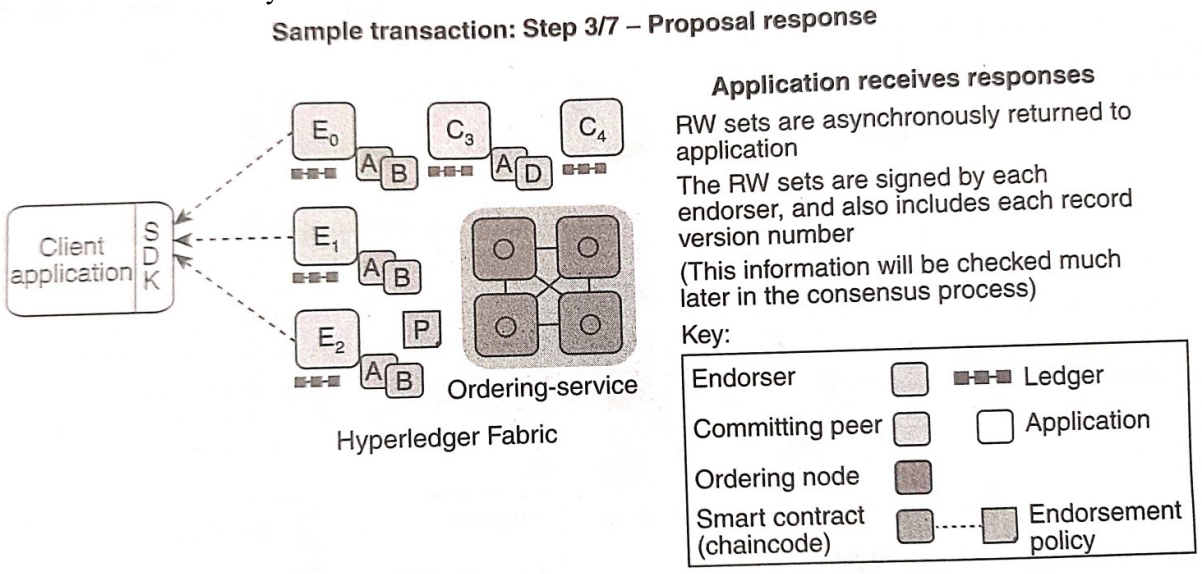
These RW sets are communicated back to the client application with endorsement data from each endorsing peer.



Fabric transaction flow – Step 2.

3)Proposal Responses Receiving:-

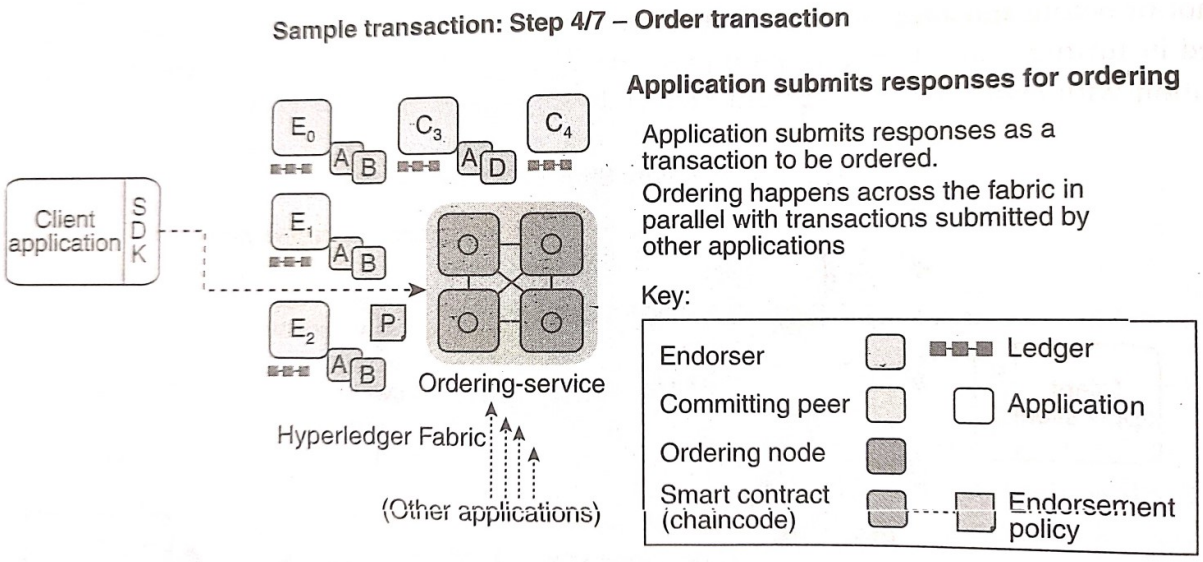
When client application receives RW set with endorsing information, these responses are not guaranteed to be returned in the same order in which client requested. This response also includes another piece of information known as version number that is maintained on the world state database for each entity.



Fabric transaction flow – Step 3.

4)Ordering Transaction:-

Upon receiving response from all the endorsing peers, the client application sends them to the orderer. The orderer collects such details for multiple other transactions happening in parallel within the Fabric network. Again, it is implicit that the client application is aware of the endpoint or Service API details of the orderer.



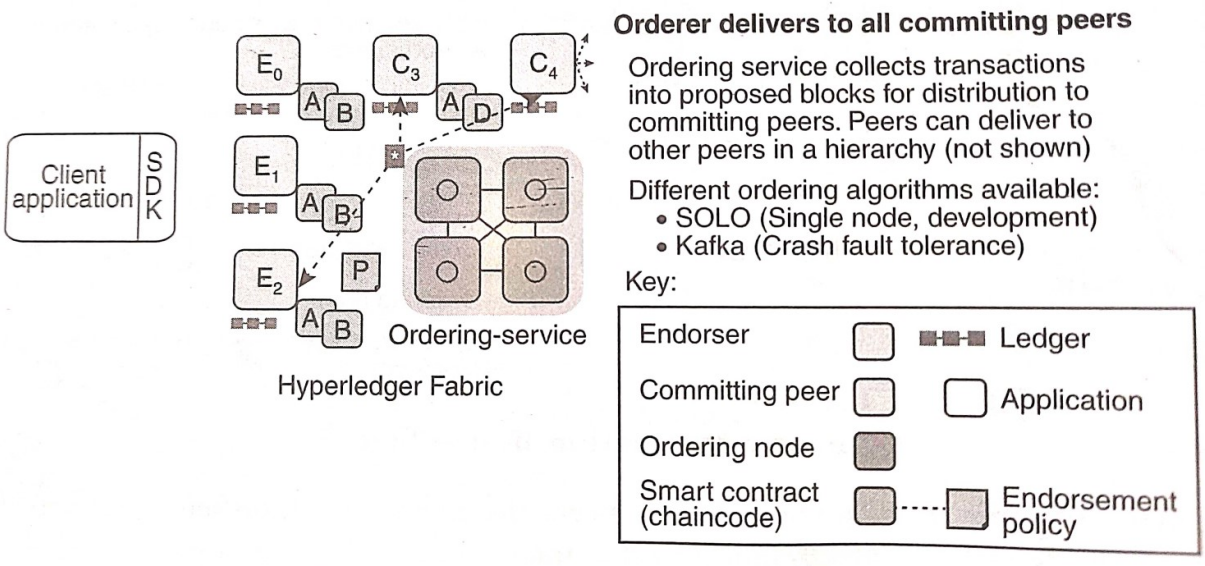
Fabric transaction flow – Step 4.

5)Deliver Transaction:-

The orderer receives the incoming transaction proposal, all the RW sets, and endorsement data related to that proposal from the client application.

It then sorts these transactions in chronological order and constructs a block based on things such as block size limit defined during network configuration.

The orderer does not perform any transaction execution or validation. It just orders transactions, creates a block, and broadcast it to peers in the network.

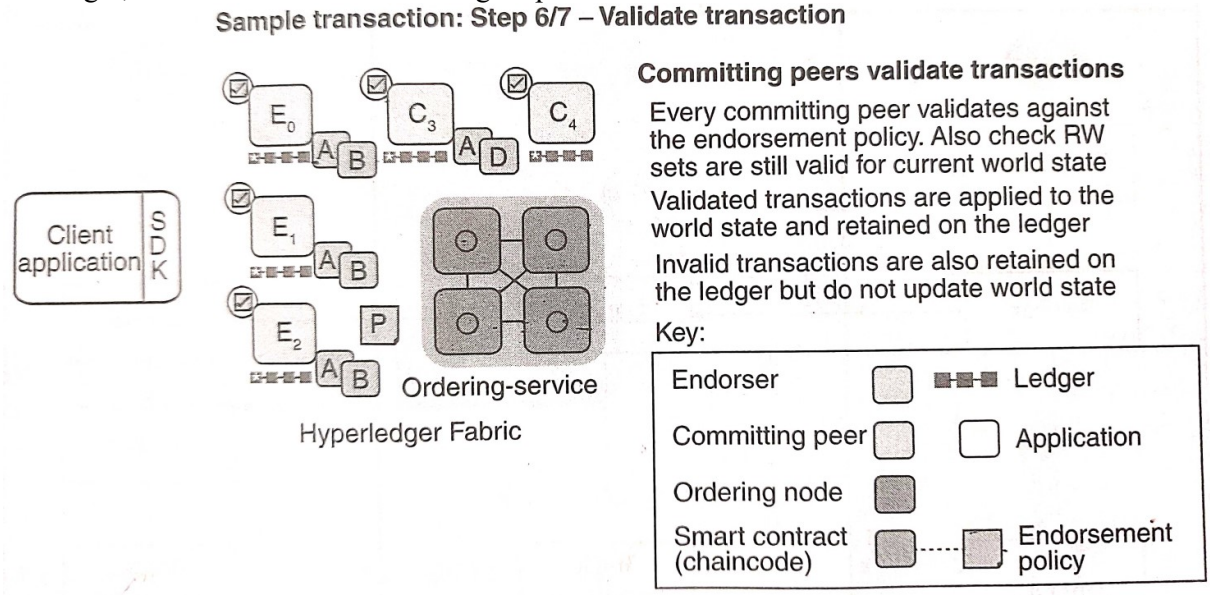


Fabric transaction flow – Step 5.

6)Validating Transaction:-

When the transaction request reaches committing peers (and endorsing peers as well), they validate the RW set and version number to check if the transaction still remains relevant as per the current world state.

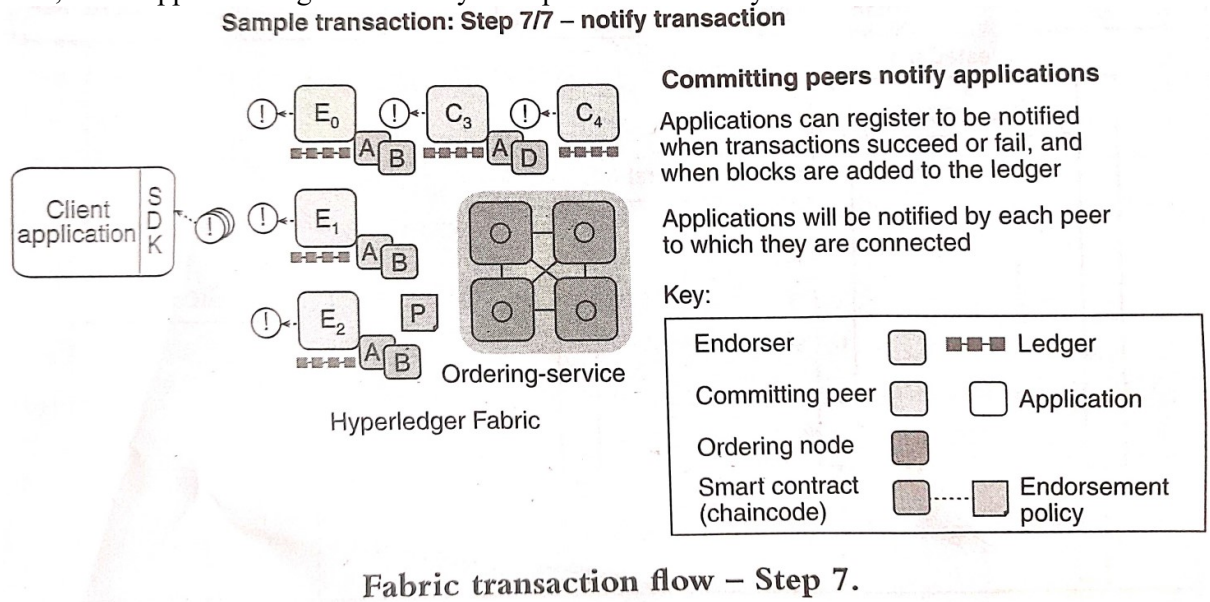
After that, endorsement policies are checked. If a transaction is found to be valid, the peers then update their respective ledger and world state Db. Any invalid transaction also gets written to the ledger, but the world state does not get updated.



Fabric transaction flow – Step 6.

7)Notify Transaction: -

Since the entire process of transaction update on Fabric ledger is asynchronous, client applications usually register for an event that notifies them whenever a transaction is added. In this case, client applications get notified by each peer to which they are connected.



Fabric transaction flow – Step 7.

FabCar Use Case Implementation

Introduction: -

Here, we will discuss about **vehicle identification use case** and how this solution can be architected. We have chosen a multi-actor network for vehicle identification.

One of the key problems in vehicle business is the missing information about ownership and the complete history of the vehicle. This result in losses for customers and at times legal issues when the ownership of the vehicle changes multiple times.

Consumers are now increasingly looking for transparency and full traceability to make sure that the car they purchase does not have any legal complications and is indeed genuine.

We will consider a **Fabcar network** that will track the ownership of the car, starting from the manufacture till it is sold to various customers. An actual network shall involve multiple players such as car manufacturer, distributor, dealer, agent, road transport authorities, service center, auto insurer, customer, etc.

Let us now go through the fitment assessment.

i)Use Case Description:

The following are the steps that have been identified for Car Ownership Tracking use case:

1. Workflow:

a) Car manufacture can create a new car and send it to the distributor with the help of a transporter.
b) Manufacturer updates regional transport office with details on new car.
c) The distributor now splits the stock received and distributes the cars to various dealers.
d) The dealers sell the car to the customers.
e) The car is now registered at the transport office against the customer.
f) Customer can approach the service center for servicing the car.
g) Customer can sell the car to another customer.
h) Ownership changes are approved and updated in transport office records.

2 Benefits for players:

(a) Availability of complete journey of the car, from the manufacturer till the customer, helps in reducing audit costs for manufacturers, distributors, and dealers.
(b) The Audit Trail can simplify loan and insurance processes.
(c) Helps in quicker recall in case of any faulty equipment that gets discovered at a later stage.
(d) Customers get to make informed decision regarding the car they are purchasing, such as if it is coming from a source that fits their ethical framework. Customers are happy to pay more for this transparency.
(e) Customers buying a used car from another customer can be assured that the used car is genuine and free from any legal complications.
(f) All other participants benefit through value generated and extra money spent by customers.

ii)Blockchain Relevance Evaluation Framework:

1. Does use case involve sharing assets/valuables/data between multiple participants?

Yes. Multiple participants are involved who are not confined to a single organization. Car is the Asset that is shared between various participants.

2. Is there impact due to sharing/hiding information between participants?

Yes, intermediaries are involved. The car does not reach the customers directly. They are passed through intermediaries such as dealers and agents.
In this example, intermediaries can also be accommodated in the blockchain.
It should be noted that not all blockchain use cases focus on eliminating intermediaries. Some of the use cases may still involve intermediaries, but the reliance on intermediaries is reduced.

3. Does the solution require shared write access?

Yes. Car is the asset that moves through the supply chain, that is, the ownership of the car is Changed as it moves through the supply chain.

4. Can the solution work without delete?

Yes. As of today, car supply chain does not mandate delete of data. Also, delete of change of hands will not help build the transparency desired.

5. Is it required to store large amount of non-transactional data?

No. The properties of car can generally be digitized and stored on the blockchain. Any related documents shall be stored on a central database by the needed parties outside the blockchain.

6.Is it required that only a very small group or an entity needs all the control?

Yes. The supply chain functionality is controlled by the parties participating in the supply chain.

7. Does use case involve high-performance (millisecond) transactions?

No. This process spans through various participants over a period of few days to few weeks and does not demand any high-performance transactions.

Now, we know that the assessment revealed that it is a very good use case for blockchain Also, since a small group needs to control the network, unlike a public network such as Bitcoin, a permissioned blockchain such as Hyperledger Fabric would be a good fit for this use case.

iii)Identification of Actors, Roles and Permissions:

Actor	Role Permission
Car Manufacturer	Can create a new car and record it on blockchain. Can sell the car to the customer. Can trace the complete ownership history of the car.
Transporter	Can deliver the car to the distributor. Can trace the complete ownership history of the car.
Distributor	Can send the car to the Dealer. Can trace the complete ownership history of the car.
Transport Authority	Can approve car registration. Can approve car ownership changes.
Dealer	Can sell the car to the Customer. Can trace the complete ownership history of the car.
Agents	Can sell the car to the customer. Can trace the complete ownership history of the car.
Customer	Can buy or sell the car. Can trace the complete ownership history of the car.
Service Center	Can service the car. Can trace the complete ownership history of the car

iv)UML Diagrams:

For the sake of this example, we will consider only 3 three parties - car manufacturer, dealer, and customer. However, the solution approach and concepts can be extended to more number of actors as well. We will be utilizing UML diagrams to capture these details as shown below:

ACTIVITY DIAGRAM:

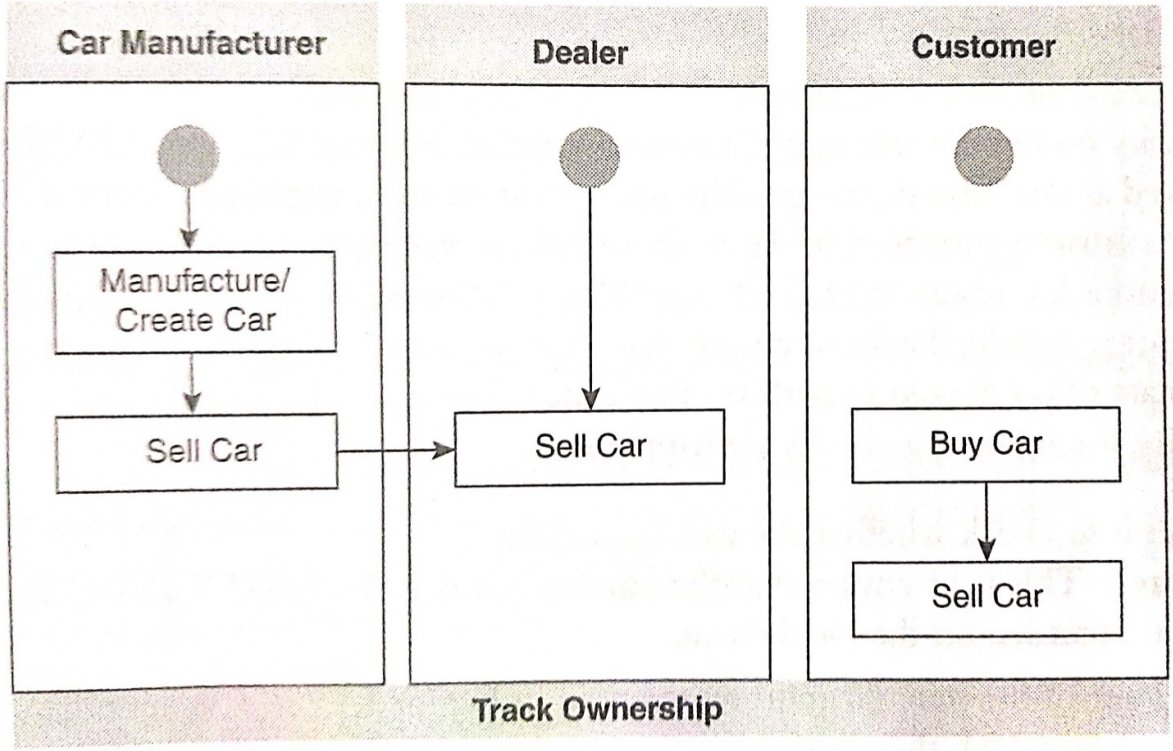


Figure 4.9 Car ownership tracking use case activity diagram.

SEQUENCE DIAGRAM:

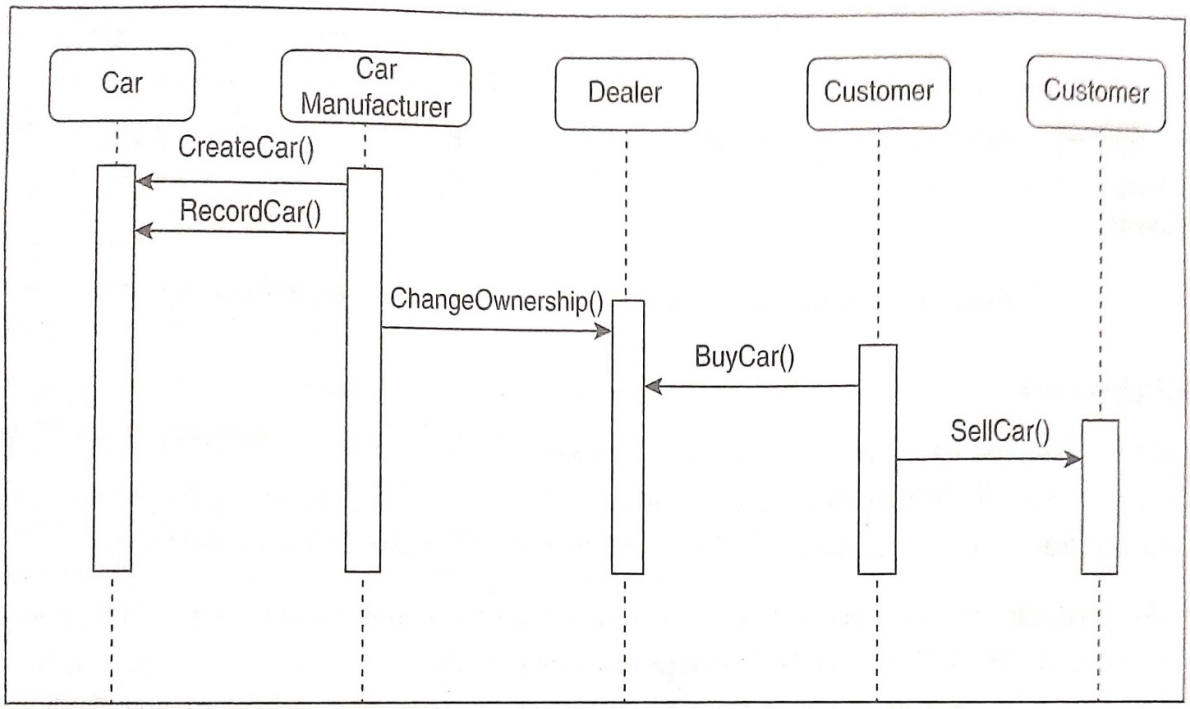


Figure 4.10 Car ownership tracking use case sequence diagram.

STATE DIAGRAM:

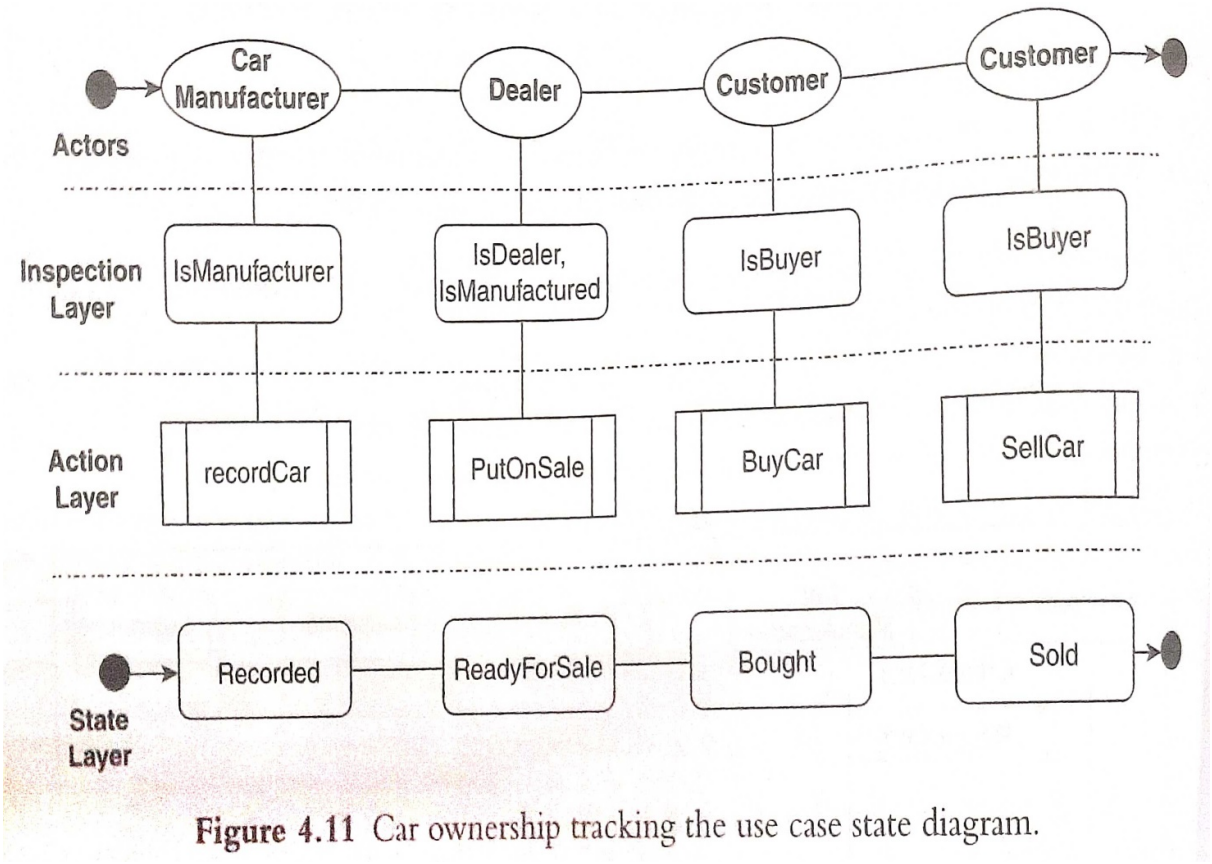


Figure 4.11 Car ownership tracking the use case state diagram.

v)Developing Smart Contract Coding:

Smart contracts in Fabric are known as chaincode. They contain the logic to perform transactions and change the state of an asset.

Currently, Fabric supports four programming languages to write

-GO,
-Node.js,
-Java, and
-Solidity

vi) Developing Application:

Developing applications using Hyperledger involves the following

1. Installing Hyperledger Fabric.
2. Setting the Ledger
3. Issuing identity to participants (peers and orderer).
4. Setting the channel.
5. Defining access control lists (ACL).
6. Writing chaincode.
7. Installing chaincode.
8. Invoking chaincode.
9. Creating APIs for client applications.

Access Control List (ACL) is a set of rules used in Fabric to define and manage access to different resources/entities within a Fabric network.

Invoking Chaincode Functions Using Client Application

Introduction: -

Hyperledger Fabric provides SDK for Java and Node programming languages that can be used by clients to interact with Fabric network. Here we use Node.js SDK. This SDK provides APIs in the form of Node.js packages to interact with the Fabric blockchain.

Developers should be able to install and use these packages like any other package within a Node.js application.

There are mainly four modules within these packages:

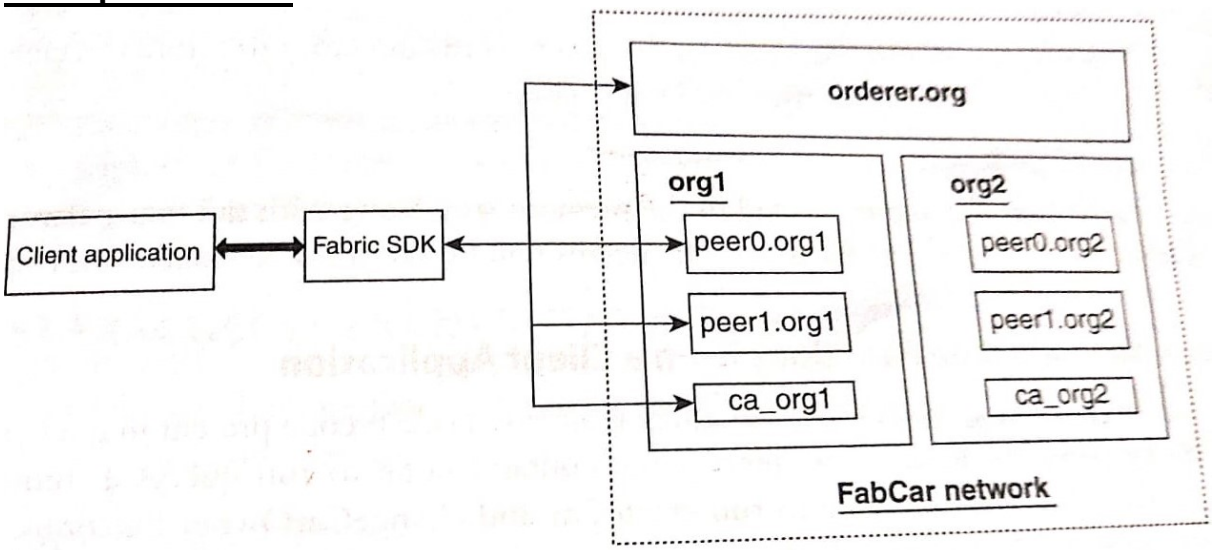
1. fabric-network: Provides high-level APIs for client applications to submit transactions and evaluate queries for a smart contract (chaincode).

2. api: Pluggable APIs for application developers to supply alternative implementations of key interfaces used by the SDK. For each interface there are built-in default implementations.

3. fabric-client: Provides APIs to interact with the core components of a Hyperledger Fabric-based blockchain network, namely the peers, orderers, and event streams.

4. fabric-ca-client: Provides APIs to interact with the optional component, fabric-ca, which contains services for membership management.

Conceptual View: -



Conceptual view of interaction between client application and FabCar network components.

The client application is already included within the Fabric installation and is available inside fabric- samples/fabcar/ directory. There are three flavors of this application represented by three folders - javas- cript, typescript, and javascript-low-level. Here we use JavaScript version of this application.

Fabric, being a permissioned network, requires every entity to establish its identity to interact with the blockchain network. So, our client application also needs an identity to interact with fabcar chaincode.

The javascript folder already contains files named enrollAdmin.js and registerUser.js to assist us. The code within enrollAdmin.js helps in creating identity for Admin. It uses the APIs within fabric-ca-client and fabric-network along with the first-network configuration details to create new cert and key for Admin user. The resulting credentials are stored within wallet/admin folder. This admin is used to register new client user.

Setting up Admin and Client User: -

To begin setting up client application, the below mentioned steps can be followed:

1. **Open a terminal console** and navigate to javascript folder inside fabcar directory.
2. Run **npm install** command which would install all the required node packages.
3. Run the command **node enrollAdmin**.
4. Run the command **node register**.

Invoking Chaincode Functions from a Client Application

In order to invoke fabcar chaincode functions, there is already **node.js** code present in **query.js** and invoke. js files within the **javascript folder**.

The **query.js** file contains code to run queryCar function. On the other hand, **invoke.js** file contains code to run createCar and changeCarOwner functions. Within these files, the inputs to be supplied to the chaincode functions are hardcoded and can be modified as needed.

Use the commands for the following to run from a node console to interact with fabcar chaincode from client application created using node.js.

1. **query AllCars** chaincode execution from client application.
2. **Adding New Car** from Client Application
3. **query AllCars** execution from client application after adding new Car

Inter Planetary File System (IPFS)

Introduction: -

InterPlanetary File System (IPFS) is a distributed storage and access system for files. The file could be of any type a computer can store such as a webpage, document, e-mail, or database records.

It offers the features of **BitTorrent** with the ability to transfer large files quickly over a network and **Git** with the built-in versioning of data.

In today's world, file sharing happens via HTTP communication protocol which is centralized. Any website or a file that a user wants to access is normally stored on servers managed by some central authority. The user has to access this content based on its location (i.e., using its URL). This establishes a single point of failure. Thus, if the servers are down, the content will not be available.

To address these limitations, IPFS uses a different addressing mechanism, which is content-based addressing instead of location-based addressing. The file is split into multiple data objects and these objects are distributed across multiple peers on the IPFS network. This enables better resistance to fault tolerance. The files are shared across users in a peer-to-peer fashion, which takes advantage of proximity of peers for faster delivery unlike HTTP.

Content-based addressing:-

IPFS uses content-based addressing instead of location-based addressing. It means that instead of requesting a file available at a particular location/URL, a user would request peers to provide a file with a particular content. The file stored on IPFS has an identifier known as Content Identifier (CID). This is nothing but the hash of the file itself, which serves as a unique fingerprint to that file contents. Any peer hosting the file with that requested CID delivers the content to the requestor.

Hashing the file contents brings in another feature to IPFS network known as deduplication. It means that when multiple users are trying to create the same file on IPFS network, it will be created only once making the IPFS network very efficient. IPFS offers another feature known as Self-certifying File System (SFS), which ensures authenticity of data being shared. An IPFS peer node uses its private key to sign the IPFS object it publishes. This signature can be verified using the sender's public key to check the authenticity of data.

Data Storage and Retrieval on IPFS:-

LIPFS stores data in chunks of size 256 kB known as IPFS Objects (Figure 7.1). This object mainly contains two fields:

1. Data of type Binary Large Objects (BLOB).
2. Links or type array that links to other IPFS objects

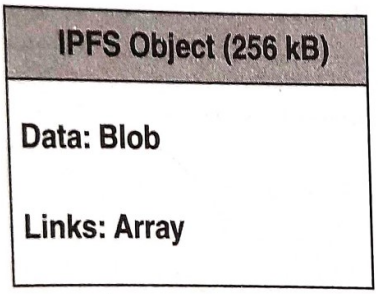
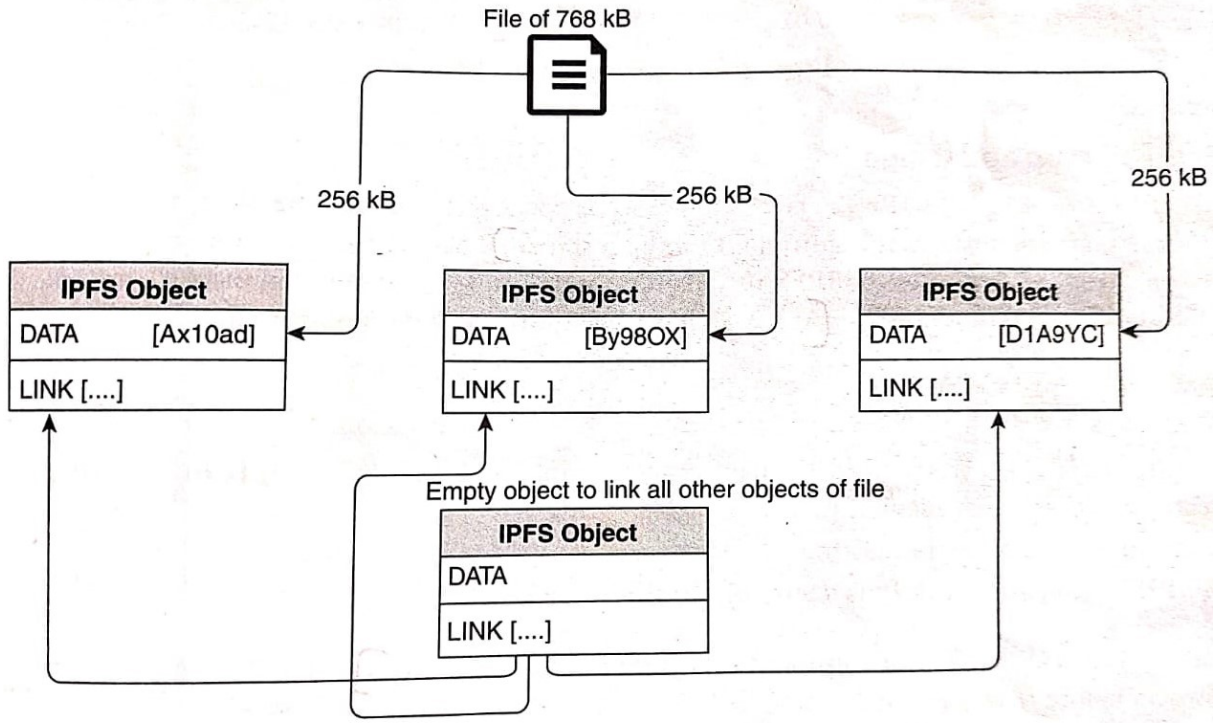


Figure 7.1 Conceptual view of IPFS object.

If the file size is more than 256 kB, IPFS creates multiple objects till the entire file is accommodated. In the end, it creates an empty object which contains just links to all other objects.



Representation of 768 kB file storage on IPFS.

IPFS stores file contents in an immutable fashion which means when a file is published, it is not possible to alter the contents of that file as it changes the hash or content ID of that file.

One may wonder, how do updates happen in IPFS? The version updates to a file are handled in IPFS using commit objects. Whenever a file is added to the network, a commit object is created and when the file is updated, a new commit object gets created which points to the previous commit object (Figure 7.2).

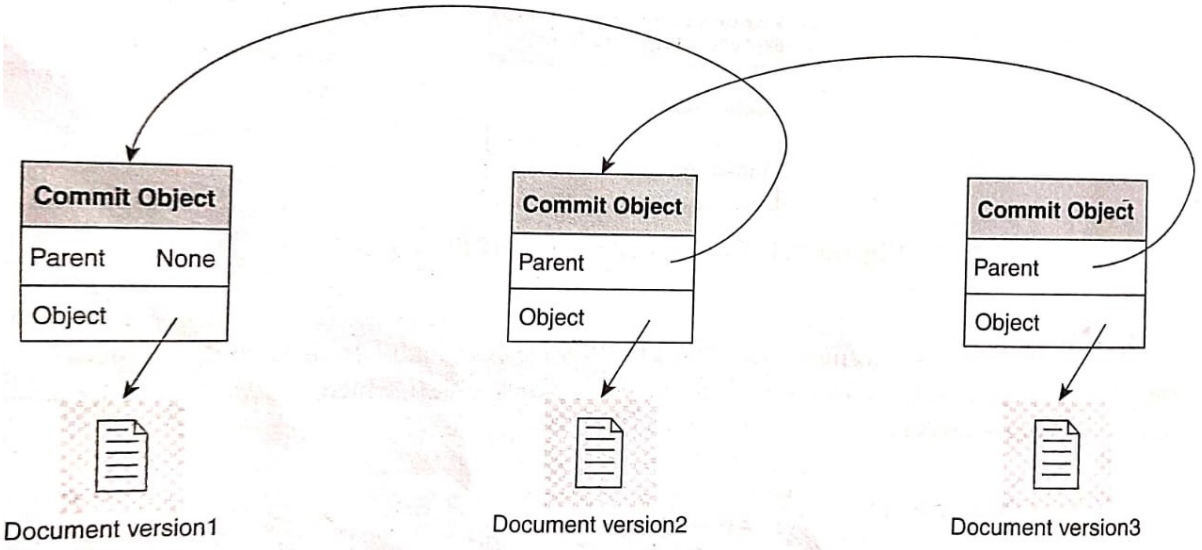


Figure 7.2 Versioning in IPFS.

In this way, the entire history of a file is available on the network. All these tasks are handled automatically by the IPFS software.

IPFS Setup and usage:-

In order to become part of IPFS network, one needs to install and run the IPFS software on local computer. As a prerequisite it is required GO to be installed.
At a high level, the steps involved in setting up IPFS are as follows:

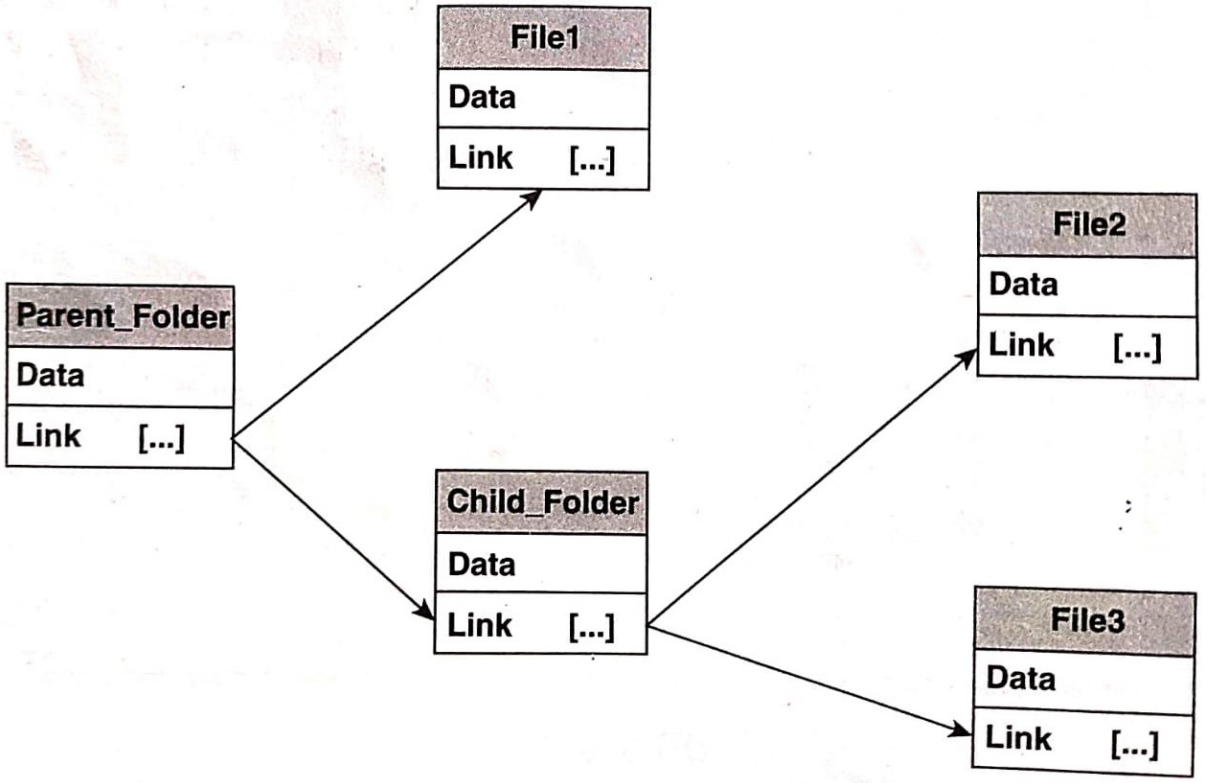
1. Download dependencies.
2. Install IPFS.
3. Initialize IPFS - navigate to ipfs binary from a terminal window and run ipfs init command. This creates a hash for peer identity.
4. Verify initialization by requesting a file [ipfs cat /ipfs/<HASH>/readme].
5. Join IPFS public network [ipfs daemon].

IPFS supports a wide variety of commands for different functionalities. Table 7.1 provides information on how to explore them.

Table 7.1 Commands for different functionalities

Command	Description
Ipfs	Shows a list of basic IPFS commands
ipfs commands	Shows a list of all IPFS commands
ipfs <command_name> —help	To get help about a specific command. Ex: ipfs daemon —help

A hash will be created for the parent directory and this hash value can be used with the gateway URL to view folder contents.



IPFS objects created for the sample folder structure.

IPFS in future and Relevance to Blockchain:-

The primary objective of IPFS is decentralization of data storage. Blockchain plat- forms such as Ethereum decentralize the computation.

IPFS brings in the decentralization aspect to data storage requirements of blockchain and enables creation of truly decentralized applications.

IPFS can be conceptualized as a collection of good features such as decentralization from Bitcoin, peer-to-peer sharing of large data from BitTorrent, and versioning from Git.

Also, the deduplication feature of IPFS ensures that only one copy of a content is created on the network.

The following are some of the prominent projects built on IPFS:

1. **Akasha:** A next-generation social network.
2. **Balance3**: A triple-entry accounting platform.
3. **BlockFreight**: An open network for global freight.
4. **Digix:** A platform for tokenizing physical gold.
5. **Infura:** An infrastructure provider for decentralized application (DApp).
6. **Livepeer:** A decentralized live video-streaming platform.
7. **Origin:** A peer-to-peer marketplace for the sharing economy.
8. **uPort:** An SSI system.

Zero-Knowledge Proofs

Introduction: -

One of the main concerns associated with physical identities and their digitized versions is that they disclose more information than is needed. A simple example is when someone wants to purchase alcohol in the United States, all they have to prove is that he/she is over 18 years of age.

However, when passport or driving license or state identification document (ID) is used to prove this fact, apart from disclosing what exactly is needed, these identity forms disclose a lot of other information which the buyer would not be interested to do so. This is where zero-knowledge proofs (ZKPs) fit in. They help in proving a fact without actually revealing the data that is not needed.

A person or entity possesses some information without actually revealing the exact details to the other person or entity. All these are handled by cryptography algorithms which deal with ZKPs.

Properties: -

A zero-knowledge proof must satisfy the following properties:

1. **Completeness:** It should convince the verifier that the prover knows what they claim they know.
2. **Soundness:** If the information is false, it cannot convince the verifier that the prover's information is true.
3. **Zero-knowledgeness**: It should reveal nothing else to the verifier other than proving that the stated information is true.)

Practical Application: -

One of the practical applications of ZKP is in a private blockchain network known as ZCash. Bitcoin and Ethereum blockchain networks use public address of a user to hide the actual identity. However, the transactions are visible to everyone. Also, using some data correlation and behavior analysis techniques it may still be possible to find out some information about the users in question.

However, ZCash uses a different approach in handling transactions. It uses a modified version of ZKP known as zero knowledge succinct non-interactive argument of knowledge (zk-SNARK). The zk-SNARK technology reduces the size of the proofs and the amount of computation required to validate them.

It guarantees the validity of transactions while keeping the sender, recipient, and other transaction details anonymous. zk-SNARK converts the transaction content that needs to be verified into a proof that two polynomial products are equal, combined with homomorphic encryption and other advanced techniques to protect the hidden transaction amount while performing transaction verification.

To implement zk-SNARK on Ethereum blockchain, there is a toolbox available known as Zokrates, which helps in generating proofs in a high-level language from DApp and verifying them from Solidity. Developers can install **Zokrates Docker** image and run some pre-configuration steps. In the resulting code file, the Zokrates code can be modified as per the needs

The following are some of the use cases where ZKPs add value.

1. **Anonymous voting**: Eligible voters can prove their voting right using ZKP algorithms and the voting itself can be recorded on a blockchain.
2. **Privacy:** During transactions such as peer-to-peer transfer, digital asset exchange, etc.
3. **Selective data disclosure**: Add capabilities to pick and share only certain information from an SSI issued to a user. For example, proving good credit history or credit rating without actually sharing and details, selectively sharing health related info, etc.

Oracles

Introduction: -

Oracles play an important role in connecting blockchain systems with external systems. Consider a smart contract for travel insurance which automatically initiates claims when the bus or train gets delayed or cancelled. In order to get the status of bus/train status, a blockchain network will have to interact with an external application.

These external applications or systems that provide reliable data to blockchain network are known as oracles. An oracle itself is not a data source, but it serves as a gateway for blockchain to interact with external data sources.

Oracle Gateway:-

As illustrated in Figure 7.5, a smart contract requests one or more oracles to provide data.

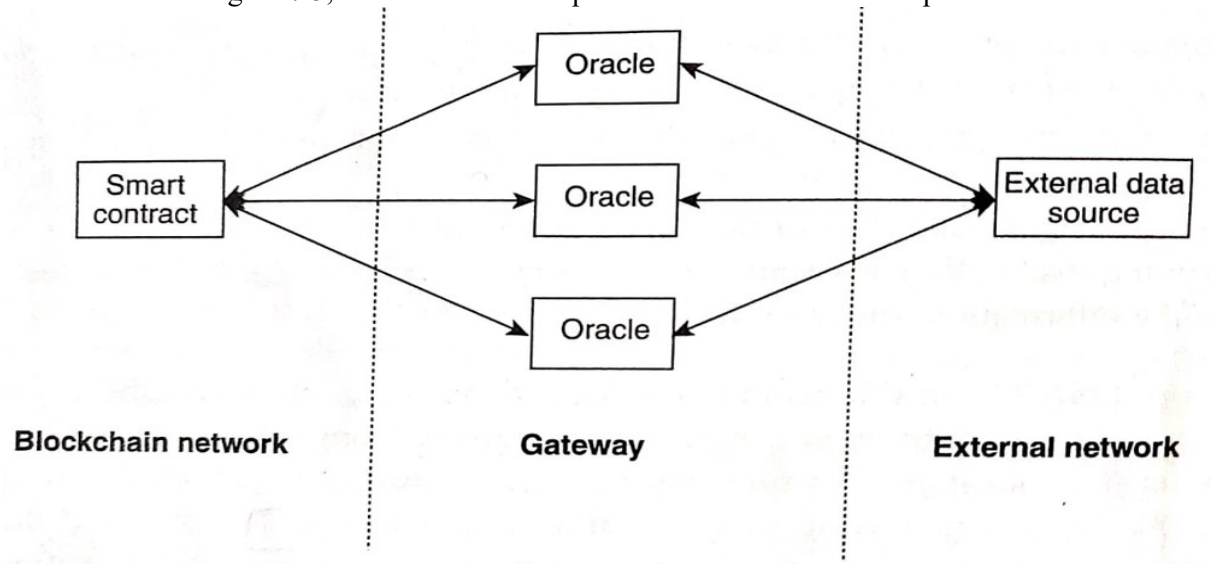


Figure 7.5 Oracle as gateway.

The oracle may then invoke API of an external data source, translate it to the format needed by smart contract, and then sends the response containing requested data back to the smart contract.

Primarily there are two types of Oracles:

1. Trusted oracles.
2. Trustless oracles.

Table 7.2 highlights the difference between these two types

Table 7.2 Difference between trusted oracles and trustless oracles

Trusted Oracles	Trustless Oracles
Usually setup by enterprises and controlled by an administrator	Public and not controlled by a central authority
Centralized in nature and less fault tolerant	Decentralized in nature and more fault tolerant
Secure and reliable	May be malicious or compromised
Suitable for private blockchain applications	Suitable for public blockchain applications

Public oracles are subject to some threats, which may lead to problems such as loss of trust or reduced data quality. Some of the common attacks are as follows:

1. **Sybil attack:** A dishonest entity impersonates an oracle.
2. **Cartel attack**: A group of entities with malicious intentions form an oracle pool in the pretext of influencing the output data.
3. **Free loading:** An entity may get the data from an oracle without directly querying the data from external data source.

Oracle Interaction:-

The following diagram depicts oracles interaction with smart contract.

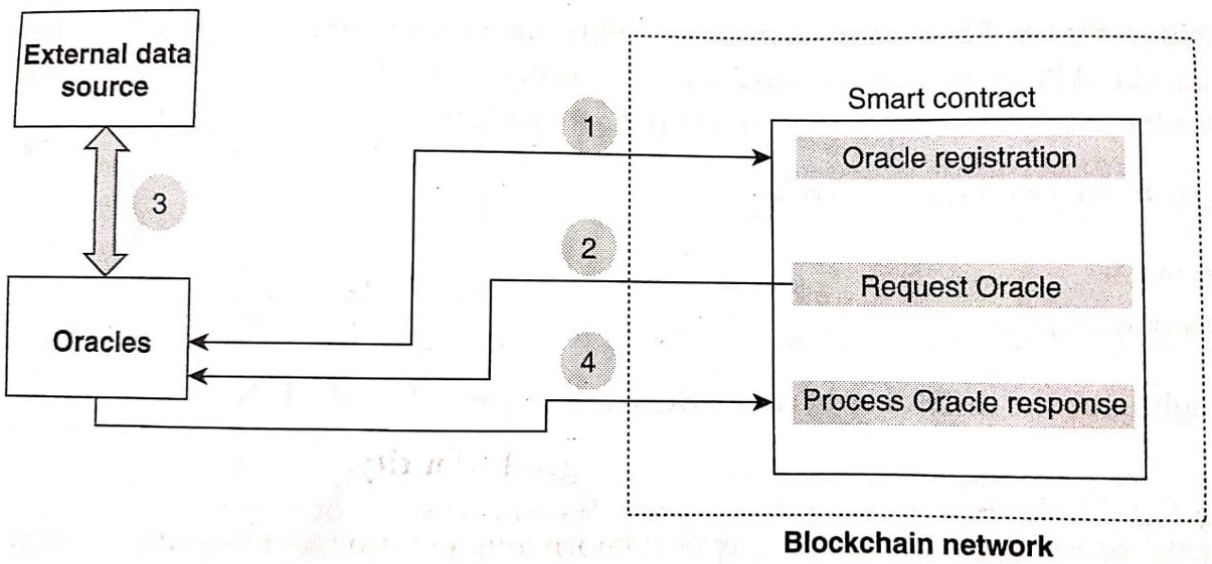


Figure 7.6 Oracle and smart contract interaction.

1. An oracle (or oracles) must register with the smart contract. A smart contract may provide a key or an identifier to the oracle during this process, which would be used for validation when the contract receives response from the oracle. This registration is a one-time process.

2. When a smart contract needs data from the outside world, it submits a request to the oracle. Unlike any other distributed application, a smart contract cannot invoke an API to interact with external applications. So, the interaction with oracle happens through event emitted by the contract. This event contains the request for oracle.

3. On the other side, an oracle would have subscribed to these events. An event listener on oracle intercepts these events from smart contracts and then triggers its internal flow to fetch data from an external source say via API.

4. The oracle transmits data to smart contract back by invoking a contract function that processes oracle response and this completes the data flow cycle. Optionally, a smart contract can emit another event as an acknowledgement to the oracle upon processing its response.

While building blockchain solutions, it is important to identify the scenarios where and oracle is needed and also choose the type of oracle depending on the private or public blockchain implementation.

Self-Sovereign Identity

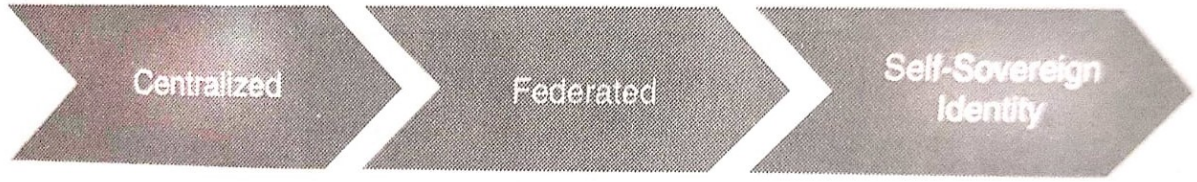
Introduction: -

Various forms of identity such as passport, driver's license, educational certificates, and voter's identity Card associated with an individual exist in today's world. These identities are mostly physical credentials in the form of documents.

Though an individual completely owns these credentials, they have some drawbacks, which are as follows:
-Prone to forgery
-Can be lost or damaged
-Difficult to verify them online
-Recreating or reissuing them is expensive.
-Disclose more information than needed.

Evolution of Digital Identity:

Digital identities started gaining traction over the last few decades with the aim of satisfying three basic requirements - security, control, and portability.



Evolution of digital identity.

Centralized Identity: Most of the identities are centralized in nature and normally issued and controlled by a single entity such as an e-commerce website or a social network. Users do not actually own such identities and can be taken away from them anytime.

For example, login credentials to a social networking site or any web- site.

Federated Identity: It basically means that an identity used by one entity can be used to access the services provided by multiple other entities.

For example, signing into other websites using a Google account. Though federation addresses some of the problems of centralized identity to some extent (particularly the portability), the identity itself is still controlled by another entity.

Both the approaches described above create honeypots of data as the identity information is still maintained by a single entity. If that information is lost or compromised, it puts the user identity at risk. Also, an individual's interactions and behavior over the web can be monitored and misused by the identity providers without user's knowledge.

Self-Sovereign Identity: SSI is the next step in this evolution of digital identities. It pushes control to edge users and advocates the complete control and ownership of a digital identity to the actual user. It aims to realize the decentralized identity concept of web 3.0.

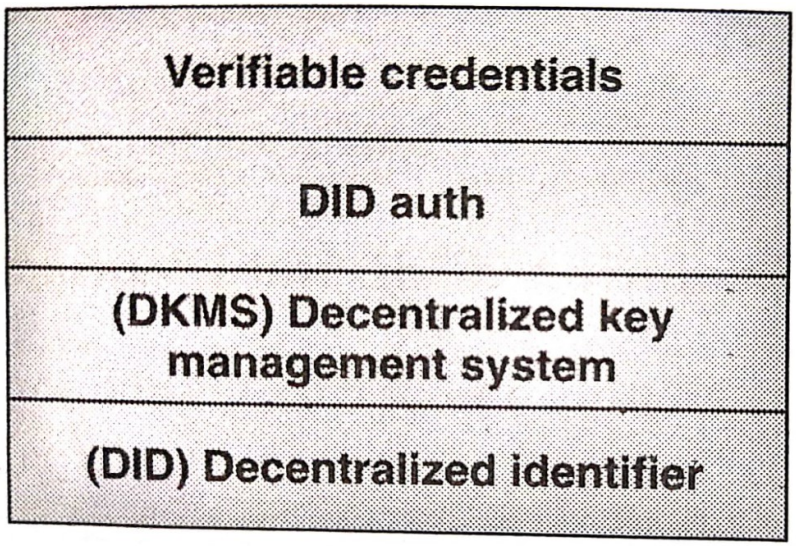
Christopher Allen, a noted decentralized identity technologist, mentions the following 10 guiding principles of SSI:

1. Existence: Users must have an independent existence.
2. Control: Users must control their identities.
3. Access: Users must have access to their own data.
4. Transparency: Systems and algorithms must be transparent.
5. Persistence: Identities must be long-lived.
6. Portability: Information and services about identity must be transportable.
7. Interoperability: Identities should be as widely usable as possible.
8. Consent: Users must agree to the use of their identity.
9. Minimization: Disclosure of claims must be minimized.
10. Protection: The rights of users must be protected.

Building Blocks of SSI:

A distributed ledger technology such as blockchain acts as a key enabling layer in SSI. The synchronized ledger and tamper-proof nature of blockchain helps in the process of creation and verification of decentralized identities.

Apart from blockchain technology, there are few core concepts which enable SSI as shown in Figure 7.7.



Decentralized Identifier: The Decentralized Identifier, or DID, forms the basis of SSI. It is a new type of identifier associated with a digital identity.

Each entity can be identified by a DID. It is a string text containing different parts separated by a colon. A general syntax for DID is **did:{method-name}:{method-specific-id}.**

Decentralized Key Management System: Dealing with digital identities at the very basic level involves using public-private key pair for data encryption or applying digital signatures. These keys associated with a user identity (in this particular case, a DID) should be completely owned and controlled by the end user. A suitable approach for this would be to provide key management feature through a mobile app installed and controlled on the end user's device.

DID Auth: DID Auth specifies various authentication cryptographic mechanisms to be used when the owner's DID is trying to prove its ownership. DID Auth may be a one-way interaction where party A proves control of DIDA to party B, or a two-way interaction where mutual proof-of-control of DIDs is achieved. In the latter case, party A proves control of DIDA to party B and party B proves control of DIDB to party A.

Verifiable credentials: Verifiable credentials are the actual digital credentials that are issued to an individual or entity by another entity. They represent the main visible component of SSI powered by the other components listed earlier.

Blockchain with IoT and AI/ML

Introduction: -

The Internet of Things (IoT) is another emerging technology that has potential to alter the ways in which data is captured and handled. It represents a wide range of tools such as sensors or smart electronic devices that can track geospatial data, temperature, speed, storage, and processing conditions and various details about an item or entity. This data can be transmitted to an external system using the built-in hardware and network gateway elements of lot devices.

They can even receive recalibration or other operational instructions from a trusted and reliable controller. These devices can also act as an electronic seal for commodities or the objects to which they are attached. Any attempt to tamper with this electronic seal can be easily identified and in some extreme cases they can even be configured to damage the object itself if tampered.

IoT coupled with blockchain:-

IoTs are going to be game changers when coupled with blockchain, particularly for supply chain use cases.

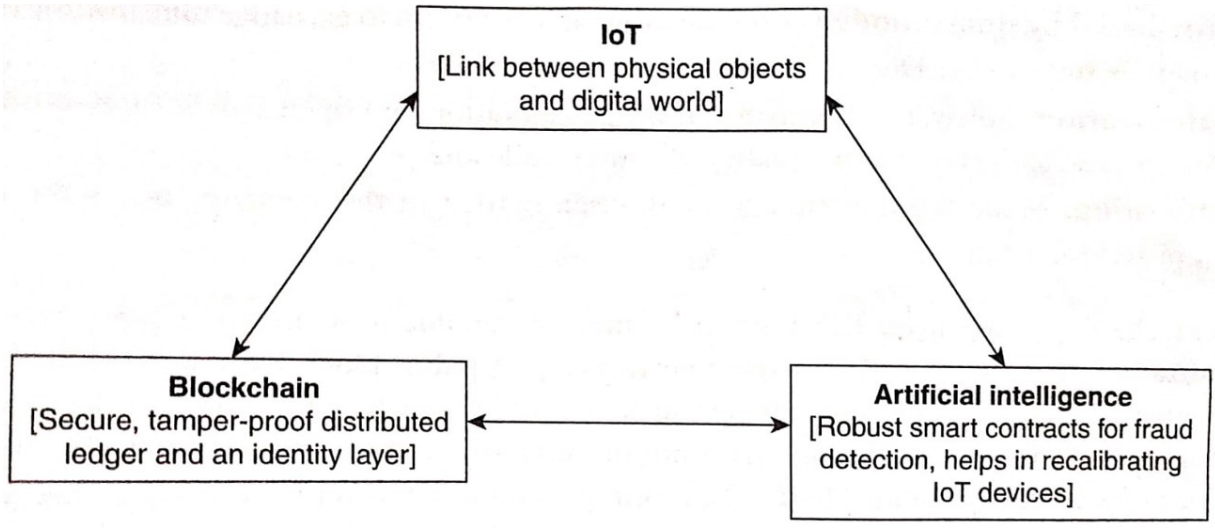
Going back to the Tuna fish use case, IoTs fitted to fishing devices can keep track of Tuna fish right from its originating place. The fish caught can be batched and for each batch, a sensor can be attached which can act as an electronic seal as well.

These sensors keep track of various travel paths and the temperatures at which the fish gets processed and transported. These IoT devices can periodically, keep recording their readings on blockchain.

This would provide a real-time update to multiple parties involved in the tuna tracking supply chain and also improves the quality of data available to different parties by presenting a single version of truth. In this supply chain, every participating entity (i.e., vari- ous actors and lot devices) needs to establish their identity. An SSI solution powered by blockchain can provide this secure identity layer for such interactions.

Trust Ecosystem of a future supply chain tracking Solution:

Artificial intelligence can play a key role in creating robust smart contracts on blockchain. These artificial intelligence based smart contracts can identify any fraud or violation of code of conduct during the whole supply chain tracking process and act accordingly



Trust ecosystem of a future supply chain tracking solution.

Quantum Computing and Blockchain

Introduction: -

A Quantum computer would be **capable of performing what is regarded as computationally highly intensive and complex calculation to a normal computer with much ease and in lesser** time. Since blockchain is predominantly based on cryptography, availability of commercial Quantum computers which is expected to happen in a decade or so is a high risk to existing blockchain systems.

However, it is also a threat to other systems such as secure websites and encryption channels that run on cryptographic algorithms. To alleviate this risk, new cryptographic algorithms will have to be designed and the existing blockchain systems will have to be upgraded to implement these algorithms.

Also, any new blockchain system will have to be designed to incorporate Quantum-resistant principles once they are available and standardized. One of the key challenges in building Quantum-resistant ledger is digital signatures.

Post-Quantum signature schemes are going to be larger than the existing (Elliptic Curve Digital Signature Algorithm) ECDSA. Simply plugging a post-Quantum signature scheme may work fine for a relatively small blockchain but will create major scalability issues for large blockchain networks such as Bitcoin.

Most experts are of the opinion that it would require around 5 to 10 years before Quantum comput- ers can pose a threat to the integrity of blockchain. Most are of the opinion that a gradual shift to Quantum-resistant cryptography will be necessary, as well as building the infrastructure that will support it. Blockchain will have to evolve, but it is unlikely that Quantum computing technology will fundamentally threaten their existence.

Initial Coin Offering

Introduction: -

Initial Coin Offering (ICO) is a business practice used normally by blockchain innovators to raise funds. This is similar to Initial Public Offering (IPO), but is much simpler and not as much regulated. Generally, cryptocurrency is sold in the form of tokens during ICO.

Anyone can invest in a project by purchasing tokens during an ICO. Thus, it offers token-based funding models for businesses driven by the following aspects:

1. **Platform-based business models:** Provide a digital mechanism to exchange information and value.
2. **Peer-to-peer networks:** Decentralized and trustless networks.
3. **Open innovation:** Software systems based on collaboration and open source models that allows everyone to audit and improve the quality of source code and protocols.
4. **Crowdfunding:** Monetary donation from an open market in the exchange of a stake in future product or service.

Blockchain technology empowers this kind of business model due to its inherent characteristics such as decentralization, trustless, and peer-to-peer networking. A public blockchain network such as Ethereum which has a built-in token support is very ideal to build token-based business models for ICOs.

It should be noted that a token-based economy is not immune from frauds. In order to realize an authentic shared business model envisioned by ICOs, it is important to address the risks associated with them via strict regulations. It also is important to understand that the coins that promoters who are trying to sell might never pick up and in such cases, it becomes a risky investment.

Blockchain Cloud Offerings

Introduction: -

Blockchain cloud service offerings, generally termed as blockchain-as-a-service, aim to increase block- chain adoption by reducing technical complexities and overhead in creating and operating blockchain networks.

Organizations can focus more on the core business functionality of the blockchain while worrying less about infrastructure and operational aspects of blockchain networks. Different cloud service providers are offering blockchain solutions on their platforms.

The following list highlights some of them:

1. **Amazon Managed Blockchain:** It is a fully managed service that makes it easy to create and manage scalable blockchain networks using the popular open-source frameworks Hyperledger Fabric and Ethereum. Amazon Web Services (AWS) also provides templates to build blockchain solutions.

2. **Microsoft:** It has blockchain-as-a-service offering on Azure, which helps in creating, configuring, deploying, and managing blockchain networks with a few simple clicks.

3. **IBM Bluemix:** It is a blockchain platform that provides different offerings to help all types of users on their blockchain journey and move their applications into production.

4. **Oracle:** This Platform as a Service (PaaS) offers a comprehensive distributed ledger cloud platform to provision blockchain networks, join other organizations, and deploy and run smart contracts to update and query the ledger.

As organizations look for cost-effective and convenient ways to leverage blockchain technology, it is likely that blockchain-as-a-service offerings will continue to proliferate. It is recommended to keep an eye on the latest developments in this space to build scalable and robust blockchain solutions.

Blockchain and its Future Potential.

Introduction: -

Throughout this book, a lot of features and concepts related to blockchain are explained. Blockchain is a disruptive technology which could help, transform and even redefine businesses. However, the best of blockchain is still waiting to be unleashed to the world where it will become an integral part of our lives.

The following sections outline some of the key blockchain use cases that we may come across in our daily lives.

Health and Well-Being:

Blockchain will make holistic healthcare and wellness a reality. The system will connect providers, payers, pharma, employers, retail, food, and wellness centers. Complete health and wellness data of an individual will be captured on blockchain and ownership of the data will be provided to only that individual.

This holistic system will not only help individuals in obtaining better treatment and insurance, but also motivate them to monitor their health and lead a healthy lifestyle, thus resulting in preventive healthcare.

Education:

Maintaining educational documents and submitting them for various purposes such as jobs, visas, and other background check processes is a tedious process involving different kinds of attestations and validations. Even more frustrating is the process to get a new copy of the certificate if one gets lost.

Blockchain will make our life easier in future by issuing certificates on the distributed ledger. The user has to just provide consent to the respective parties such as governments, universities, and consulates to access the educational credentials stored on blockchain, and it can be accessed and validated by these parties across the globe.

Travel:

Everyone understands the importance of passport and visa in cross-border travel. The process is being made more stringent due to the risks posed by terrorism.

Passport and visa issued on blockchain will allow passengers to travel without carrying any physical documents for passport and visa across the world. This system would also easily identify fake passports and visas contributing to increased security

Voting:

Voting no doubt is the corner stone of any democracy. But it is also one of the most expensive processes in a democracy due to the time and effort involved. It requires a voter to go to a physical location and also demands meticulous planning and execution. Yet the process could not avoid controversy and ruckus over the results.

Blockchain can redefine the voting process on the ledger where every voter can cast their vote from their homes, using the credentials provided by the voting authorities. This has numerous other benefits such as conducting frequent elections instead of once in 4 or 5 years, conducting voting for any key decisions to be made in the interim, and avoiding proxy votes.

Property:

Property registration has always been a breeding ground for fraud and ownership disputes for many centuries.

Blockchain has the potential to create a universal land registry across the globe, capturing complete history of the property on the distributed ledger including the record of all the stakeholders associated with the property at any point in time. This again avoids the need for safeguarding any documents and also prevents fraudulent ownership claim on a property.

----------**********----------