LICT Advanced Blockchain Training 2020

Trainee Name: Showkot Hossain

Trainee ID: 13

Echain: A decentralized supply chain traceability system for electronics market

---

Mentored by:
Abdullah Al Omar, Lecturer
Department of Computer Science and Engineering
University of Asia Pacific

August 18, 2020

Contents

1 Project Idea

Supply chain traceability is becoming increasingly important, especially in the electronics market since the regulators and the governments are concerned about ensuring public safety and consumer satisfaction by providing the quality products for their money. The electronic devices (i.e. mobile phones, laptops, gadgets, accessories etc) have a large increasing market size and multiple stakeholders involve in the entire supply chain. Consumers and the stakeholders often fail to verify the authenticity of these products due to lack of information sharing and traceability system. As a result, consumers are often defrauded by fake products which incur financial losses and annihilate the trust among the stakeholders. Since blockchain can ensure trust and transparency in multi party environment, it can be a promising solution to address the above problem. In this project, I propose "Echain: A decentralized supply chain traceability system for electronics market" which is a end to end traceability system for electronic products using a private permissioned blockchain platform (hyperledger fabric).

2 Background

2.1 Supply Chain

A supply chain is a network between a company and its suppliers to produce and distribute a specific product to the final buyer. This network includes different activities, people, entities, information, and resources. The supply chain also represents the steps it takes to get the product or service from its original state to the customer. Companies develop supply chains so they can reduce their costs and remain competitive in the business landscape.

Supply chain management (SCM) is the management of the flow of goods and services and includes all processes that transform raw materials into final products. It involves the active streamlining of a business's supply-side activities to maximize customer value and gain a competitive advantage in the marketplace. SCM represents an effort by suppliers to develop and implement supply chains that are as efficient and economical as possible. Supply chains cover everything from production to product development to the information systems needed to direct these undertakings. Supply chain management is a crucial process because an optimized supply chain results in lower costs and a faster production cycle.

2.2 Traceability

Traceability is the ability to trace all processes from procurement of raw materials to production, consumption and disposal to clarify when and where the product was produced by whom. Due to improving product quality and the rise in safety awareness in recent years, traceability has been increasing in importance and spreading into a wide range of fields, such as automotive, electronics, food and pharmaceutical. In this project, we will work with the traceability of electronics products. The ISO standard defines product traceability as "The tracking of consumer products with respect to the origin of materials and parts; the processing history; the distribution and location of the product or service after delivery". The European Union describes traceability similarly as "An ability to track any food, feed, food-producing animal or substance that will be used for consumption, through all stages of production, processing and distribution".

2.3 Blockchain

Blockchain is a decentralized and fault-tolerant database used to maintain a continuously growing list of records known as blocks. It can serve as an accessible and distributed ledger to record transactions among several parties in an efficient, confirmable, and permanent way. Nodes on the BC stay synchronized with each other and agree on the legitimacy of the transactions using *Consensus Algorithms* such as *Proof of Work (POW)*, and *Proof of Stake (POS)*, etc. In BC, transactions are grouped into blocks and permanently sealed using a hash value. A slight change in the block changes the hash value that propagates to the blocks mined afterward. Miner nodes can detect the modification that makes it impossible to tamper the data once stored in BC.

2.4 Types of Blockchain

Blockchains are mainly classified into three types. The classification is made from different perspectives like *consensus determination*, *read permission*, *immutability*, etc. The types of blockchain is given below:

- **Public blockchain**: Public blockchains are fully decentralized blockchain where anyone can join the network, read and write to the ledger. All the transactions on public blockchain are fully transparent as all the nodes on the blockchain can take part in the consensus process. Public blockchain is also known as permission-less blockchain since any node can join the network. Bitcoin and Ethereum are examples of public blockchain.

- **Private blockchain**: Private blockchains also known as permissioned blockchains where participants need consent from administrators to join the network. Private blockchains are less decentralized than public blockchain as all participants are not involved in consensus process. However, private blockchains have higher throughput than public blockchain. Hyperledger and MultiChain are examples private blockchains.

- **Consortium blockchain**: Consortium blockchain is a hybrid of public and private blockchain but has more likenesses to private blockchain. It is also known as semi-decentralized blockchain. Unlike private blockchain it is governed by a group of organizations rather than a single entity.

2.5 Why Blockchain?

Blockchain can ensure transparency and trust among multiple stakeholders. To ensure trust and transparency, Echain must satisfy requirements such as shared information management, proper access control mechanism and authorization for users, traceability, automation of business contracts, etc. I think blockchain or more specifically private permissioned (consortium) blockchain is well suited to serve these requirements.

Three major features: *immutability*, *transparency* and *decentralization* make blockchain irresistible for traceability of electronic products.

It provides an immutable tamper-proof records of every point in the supply chain, which includes multiple third parties. Consensus from all members of a blockchain network is required to validate each transaction, and all validated transactions are permanently recorded on the blockchain. So at any given time, it is possible to trace back all previous records related to a particular product via blockchain. No one can delete the data, not even the system administrators, which allows the government or organizations to catch the counterfeiters.

Blockchain technology creates a distributed, shared system of record among the network members. It provides a single source of truth. Whatever goes on the supply network can be visible to all

or to only selected nodes. The smart contracts enhances trust and make the business transactions smooth. And as decentralization is the core feature of blockchain, there is no centralized server; no one owns the data of the supply chain; instead, everyone in the network is a part of it. Thus no one can manipulate the data without the participation of others.

Since we are using private blockchain, it provides permissioned participation of the members. Each member of the network must have identity and access privileges in order to participate and make a transaction. Furthermore, information is shared with other network members on a need-to-know basis only. Finally, as blockchain is an evolving technology, blockchain innovators and researchers are discovering how to use the technology's unique benefits to various real life problems and adapt the changes too.

3 Design and Architecture

3.1 Blockchain platform

3.1.1 Existing Platforms

There are many existing blockchain platforms. Choosing an appropriate platform on which to build our "Echain" platform is very important in our design phase. From literature review, the available options are - Hyperledger (i.e. Fabric, Sawtooth, Indy), Corda, MultiChain etc. Echain aims to provide solution to a enterprise use case, so choosing an existing and well known platform is very important for fulfilling the requirements without any issues.

3.1.2 Public vs private blockchain

Echain must have low running costs and high transaction throughput. There are some public blockchains designed for high transaction throughput, but only the private blockchains offer the combination of all requirements which are necessary to build our system. Private permissioned blockchain uses simpler consensus mechanism, which leads an opportunity for the malicious actors to corrupt the shared ledger. However, this is not an issue in Echain since it is an enterprise consortium where companies collaborate on same supply chain, the participants are limited and the consortium can verify the identity of each participant before allowing them to join the network.

3.1.3 Why Hyperledger Fabric?

To ensure traceability properly in supply chain of electronic products, Echain must satisfy various requirements such as secure data storage, high transaction throughput, proper access control mechanism and authorization for users, asset management, sharing of information among participants and enforcement of smart contract. Thus the framework on top of which our solution is going to be implemented must have support for these requirements.

Hyperledger Fabric provides a highly customizable consortium blockchain framework with high scalability and performance. It provides required support for authentication and authorization of participants in the system through the Membership Service Provider (MSP). It also has smart contract facilities known as *Chaincode*. For privacy of sensitive data it has a feature known as *Channel*. Fabric framework provides rich queries which will be helpful for querying and auditing. It also provides a Software Development Kit (SDK) which allow external access to fabric network and chaincode functionalities. Most importantly Hyperledger Fabric is an open source technology which will not add to our TCO (Total Cost of Ownership). Thus Hyperledger Fabric practically fits all of the requirements to implement our solution on top of it.

3.2 Hyperledger Fabric

3.2.1 Fabric components

For better understanding the rest of our system, I am explaining some of the components of hyperledger fabric.

- **Shared Ledger:** It contains the current world state of the network and a blockchain. The blockchain is immutable and the world state is a shared, permissioned ledger, which keep the records (as a key-value pair) serving as a single source of truth.
- **Smart Contracts:** Hyperledger Fabric smart contracts are called *chaincode*. It contains business logic of the network. *Chaincode* is invoked when an application needs to interact with the ledger. It can be written in Golang or Node.js. I use Node.js.
- **Channels:** Channels are a logical structure formed by a collection of peers. This capability allows a group of peers to create a separate ledger of transactions. Each channel represents an independent chain of transaction blocks having transaction records for that specific channel only. It enables only permissioned stakeholders to see the transactions.
- **Peers:** Peers are the participants of the network who host ledgers and chaincode. A peer executes chaincode, accesses ledger data, endorses transactions, and interfaces with applications.
- **Membership Service Providers:** Participants in a permissioned blockchain network must be authorized to join and make transaction in the blockchain network. In Hyperledger Fabric, the membership service provider, or MSP, is the component for managing user IDs and authenticating participants to join the network. MSP is implemented as a Certificate Authority to manage certificates used to authenticate member identity and roles.
- **Ordering Service:** Ordering Service is a collection of nodes that packages transactions into a block to be delivered to peers on a channel. Previously supported configuration mechanisms for the Ordering service were Solo (single orderer) and Kafka (multiple orderer). Recently Kafka has been deprecated and Raft consensus is being used instead of Kafka.
- **Assets:** Assets are anything that has a value. An asset has a state and ownership. It is represented as a collection of key-value pairs. For our system, each electronic device is an asset.
- **Network:** The Hyperledger Fabric network is formed by the peers and contributed by the different organizations that are members of the network. Peers have an identity (digital certificate) assigned by a Membership Service Provider.

3.2.2 Identity

In our Echain consortium, the system must allow the credentials for participants of the network to be issued and revoked by the authority. This is achieved by Hyperledger Fabric Certificate Authority (CA). The CA is an important component of fabric architecture which has the capabilitites including:

- register new identities, or verify the identities in existing authentication systems
- issue digital certificates

- renew or revoke a certificate

Each organization has it's own CA - see figure 1 for visualization.

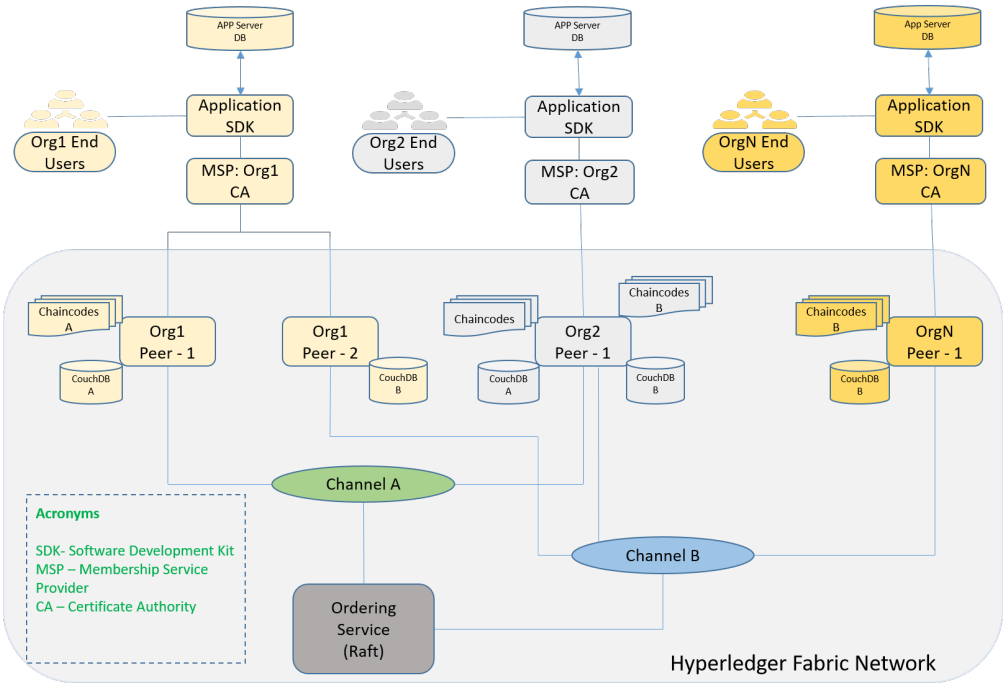


Figure 1: Sample Fabric Network.

3.2.3 Consensus

Our system must support some consensus mechanism to ensure all participants of the consortium agree on the state of the shared ledger. In this case, hyperledger fabric supports multiple "pluggable" consensus mechanism based on the requirements.

- Endorsement of the transactions is driven by the policy set by the authorities upon which participants endorse the transactions
- Ordering Service accepts the endorsed transactions and agrees on the order to commit the transactions to the ledger
- Both the endorsing peers and the committing peers take a block of ordered transactions and validate the correctness including checking the endorsement policy

3.2.4 Access control and privacy

Our system must support proper access control and privacy mechanisms including permission management of the identities and data visibility. Fabric uses *channels* to encapsulate the private data among the participants. *Channels* are restricted information sharing paths that is used to provide transaction privacy and confidentiality of data for specific subsets of the participants. It is the secret partitioning of all data, including transactions, members and channel information within a channel, which is invisible and inaccessible to any network members who are not the granted the

membership of that channel. See figure 1 for visual representation of the *channels*. To further obfuscate the data, **encryption** using cryptographic algorithms such as *AES* can be used. The decryption key will be shared only to legitimate participants to view the encrypted data.

3.2.5 Queries

The traceability information of Echain must be queryable, allowing the stakeholders to find the full history of an electronic device. We use *couchDB* as world state database which support rich query mechanisms. The consumers and the stakeholders search using device id (i.e. IMEI) to know the provenance of the authenticity.

3.3 Network Model

3.3.1 Supply chain

The real world supply chain network of electronic devices might be quite complicated. In this project, we will use a simplified generic supply chain network which can further be extended to a real world scenerio.

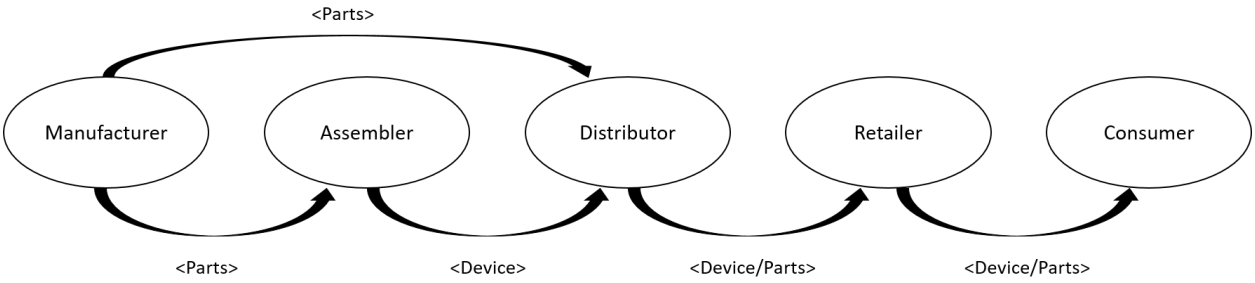


Figure 2: Supply Chain Network Model.

As shown in figure 2, there are five stakeholders in the supply chain. We will implement our solution considering this supply network.

3.3.2 Participants and their roles

- **Manufacturer:** Manufacturer produces/collects raw materials and parts for making a batch of electronic devices and send them to the assembler.
- **Assembler:** Assembler receive the parts and assembles a batch of electronic devices.
- **Distributor:** The distributor distributes the complete devices to the retailers.
- **Retailer:** The retailer shops sell the devices to the consumers.
- **Consumer:** The consumers buy the devices from the retailers upon verification of the authenticity of the devices.

3.3.3 Assets

- **Parts:** Display, microphone, speaker, battery, camera, mother board, GPU, processor etc.
- **Device:** Smart phone, smart watch, laptop, tablet, monitor etc.

3.4 Architecture

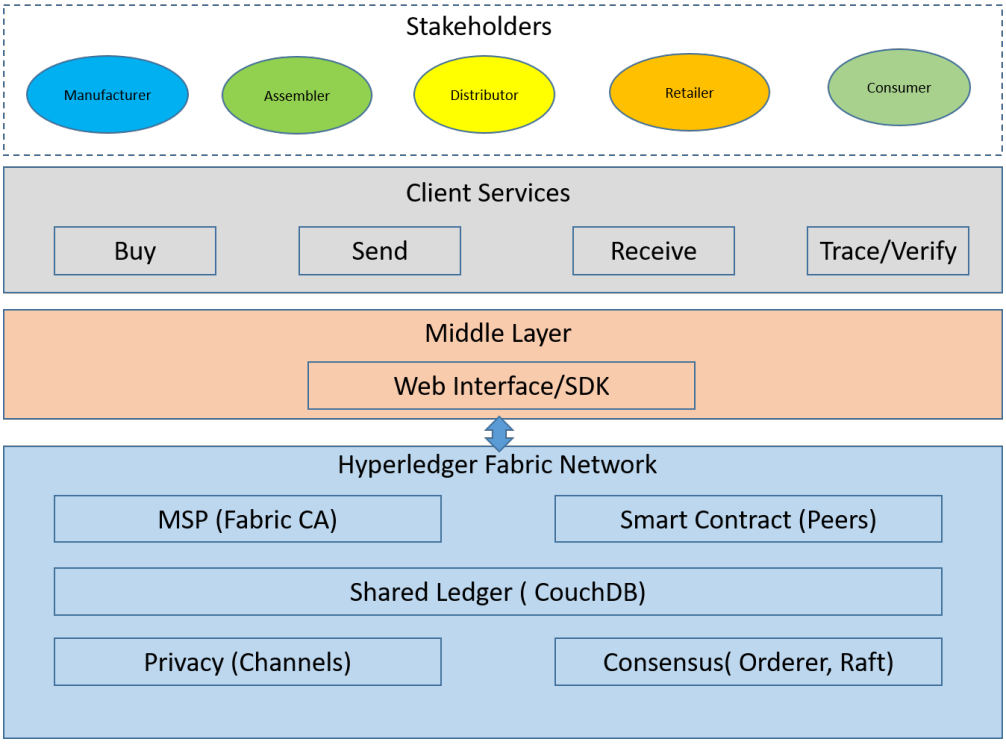


Figure 3: Architecture of the Proposed System.

The system architecture of Echain is shown in figure 3. There are mainly three components- a fabric network, web application and the participants. The hyperledger fabric network consists of Membership Service Provider for user management and authentication, Chaincode which is similar to smart contracts , CouchDB as Shared Ledger which supports rich query mechanisms, Channels to preserve privacy of data and Raft ordering service as transactions orderer. The web application provides an easy to use user interface for the clients to communicate with the fabric network. Finally there are five types of participants - manufacturer, assembler, distributor, retailer and consumer. The manufacturer produce/buy the device parts, upload the information into the blockchain using their signatures and send the parts to the assembler. Upon receiving the parts, the assembler assemble the parts to build the complete device. After quality inspection, assembler updates the complete device info into the ledger and sign the transaction with their signature. Then the batch of devices is sent to the distributor. The distributor also updates the ledger and send the products to the retail shops. Upon receiving the products, the retailer also update the ledger. At the end, when the consumers want to buy an electronic device from the retail shops, they can easily trace the entire events of the supply chain of that particular product using the unique product id.

3.5 Governance

Echain will follow open governance structure while facilitating incentives and methods for members to coordinate on decision making processes in the blockchain. Open governance will allow the users of ReliChain to elect a team that will be responsible for making technical decisions for the blockchain platform. The different actors in a blockchain network include peers, orderers, client applications, administrators, etc. For an actor to be able to consume services in the blockchain network, it must

have a valid identity. These identities are really important in the sense that they determine the exact permissions over resources and access to information that actors have in a blockchain network. Membership service provider (MSP) is the trusted authority in Hyperledger Fabric that verifies the identities of actors in the blockchain network. MSP is also responsible for turning identities into role (e.g.- client, peer, admin, etc.) by identifying specific privileges an actor has on a node or channel. This objective is achieved by defining separate MSPs locally at an actor's node and in channel configuration.

3.5.1 Member on-boarding and off-boarding

Our proposed system will manage on-boarding and off-boarding of users in the blockchain network through the Membership Service Provider in Hyperledger Fabric. Each company will define their own MSPs and each MSP will define the members and their associated roles (e.g.- peer, orderer, client, admin, etc) related to the organization. The organization admin can grant or revoke the membership of a participant.

3.5.2 Data Ownership and Privacy:

Every peer can manage their data ownership and privacy using channels.

3.5.3 Business Exchange:

Business exchanges are done through chaincodes (smart contract).

4 Implementation

4.1 Environment

I will be using Hyperledger Fabric, an open-source BC platform, to implement Echain. *JavaScript* will be used as the development language for the smart contracts. I will use Visual Studio Code as development IDE. Moreover, I will design a simple web application using *html, css, etc* to support interactions with our system. Several bash scripts were created to set up the environment. I am testing and developing my code on the system consists of Intel Core i5 processor with Ubuntu 20.04 LTS operating system.

4.2 Prerequisites

- Hyperledger Fabric v2.1.1
- Node v10.x or higher
- npm v6.x or greater
- Java 8
- Docker v17.06.2-ce or greater
- Docker Compose v1.14.0 or greater

4.3 Types of Transactions

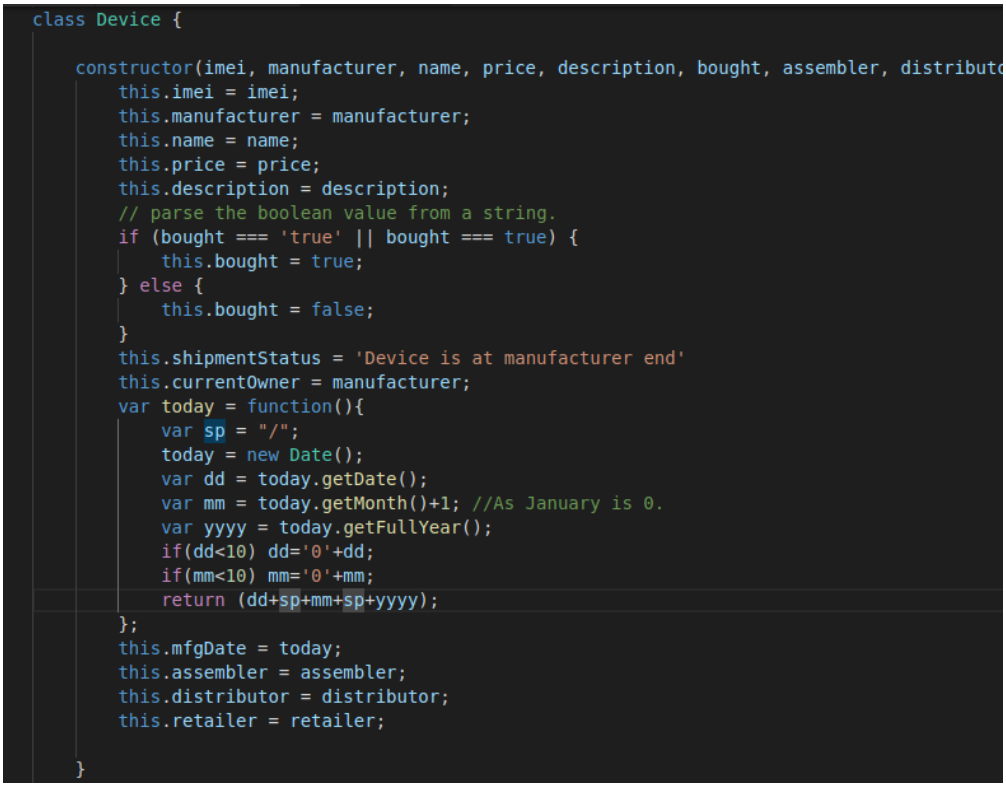
The interaction between the actors of our electronics supply chain and the assets (devices) of the system is represented through the transactions. Transactions are invoked by the users with certain parameters and there are restrictions imposed on the users for making a particular transaction. Before a transaction invocation results in read/write operation on the ledger, endorsement policies are enforced on it. Here I am describing a few basic transactions related to our Echain system below:

- `releaseDevice`: The manufacturer can create a new parts or a device. It will represent the digital identity of a device. The parts or the devices will have a identification number by which it will be tracked throughout the supply chain.
- `viewDevice`: Trace the information related to a device.
- `assembleDevice`: Assembler assembles the device parts, checks the quality and updates the status of the device.
- `distributeDevice`: The distributor receives the device and updates the shipment status.
- `buyDevice`: At consumers end point, consumer can track and verify a device. Then they can buy the authentic devices.
- `viewUnsoldDevices`: Consumers can search for the unsold devices and buy them.

4.4 Smart Contracts

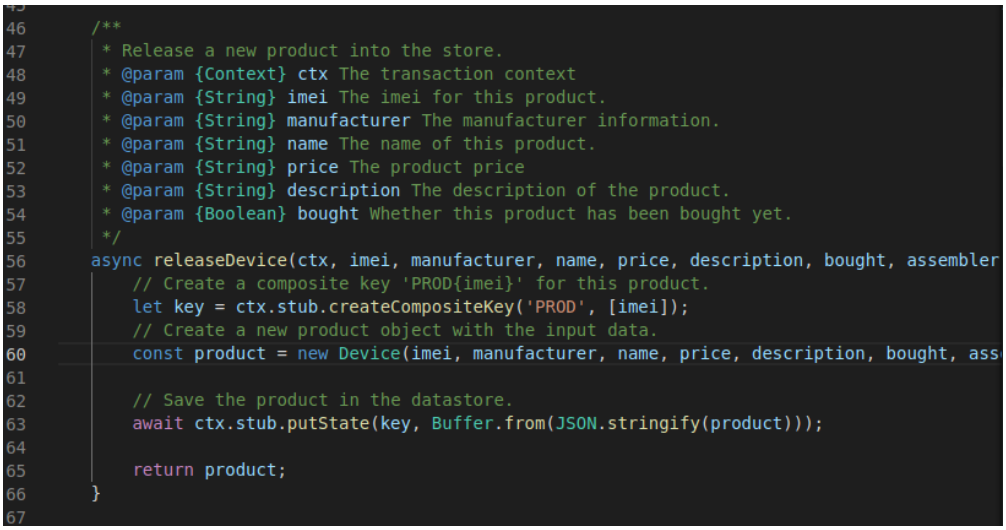
Fabric smart contract is known as *chaincodes*. I wrote the chaincodes in javascript. The echain smart contract class must import and extend the *fabric-contract-api*. It also imports a *device.js* javascript class which is shown in figure 4. The *device.js* creates the device objects/assets.

Each type of transaction discussed above is implemented in the smart contract. Here is the snapshots of *echaincontract.js* class.



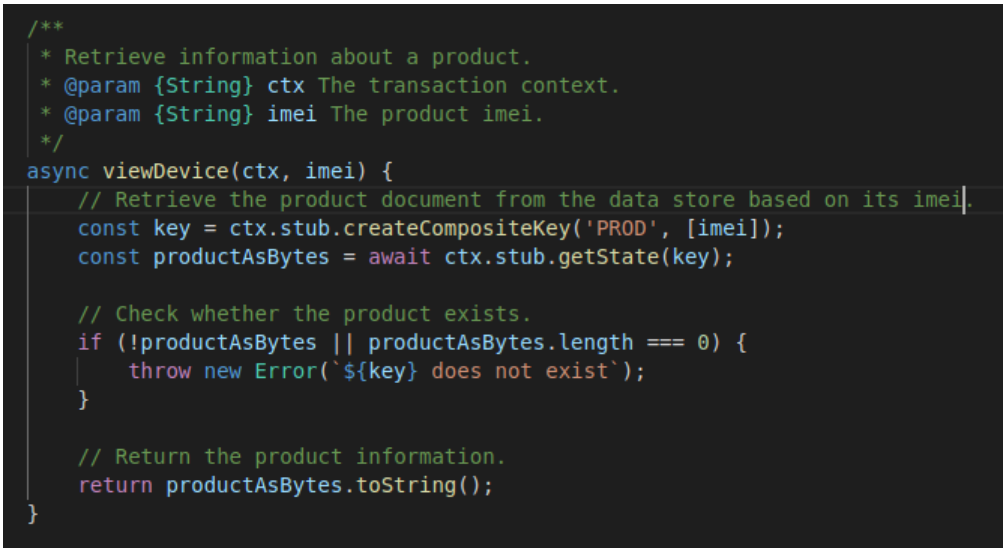
```
class Device {

    constructor(imei, manufacturer, name, price, description, bought, assembler, distributo
        this.imei = imei;
        this.manufacturer = manufacturer;
        this.name = name;
        this.price = price;
        this.description = description;
        // parse the boolean value from a string.
        if (bought === 'true' || bought === true) {
            this.bought = true;
        } else {
            this.bought = false;
        }
        this.shipmentStatus = 'Device is at manufacturer end'
        this.currentOwner = manufacturer;
        var today = function(){
            var sp = "/";
            today = new Date();
            var dd = today.getDate();
            var mm = today.getMonth()+1; //As January is 0.
            var yyyy = today.getFullYear();
            if(dd<10) dd='0'+dd;
            if(mm<10) mm='0'+mm;
            return (dd+sp+mm+sp+yyyy);
        };
        this.mfgDate = today;
        this.assembler = assembler;
        this.distributor = distributor;
        this.retailer = retailer;

    }
```

Figure 4: The device class.



```
46      /**
47       * Release a new product into the store.
48       * @param {Context} ctx The transaction context
49       * @param {String} imei The imei for this product.
50       * @param {String} manufacturer The manufacturer information.
51       * @param {String} name The name of this product.
52       * @param {String} price The product price
53       * @param {String} description The description of the product.
54       * @param {Boolean} bought Whether this product has been bought yet.
55       */
56      async releaseDevice(ctx, imei, manufacturer, name, price, description, bought, assembler
57          // Create a composite key 'PROD{imei}' for this product.
58          let key = ctx.stub.createCompositeKey('PROD', [imei]);
59          // Create a new product object with the input data.
60          const product = new Device(imei, manufacturer, name, price, description, bought, ass
61
62          // Save the product in the datastore.
63          await ctx.stub.putState(key, Buffer.from(JSON.stringify(product)));
64
65          return product;
66      }
67
```

Figure 5: Chaincode function to create a new device by the manufacturer.



```
/**
 * Retrieve information about a product.
 * @param {String} ctx The transaction context.
 * @param {String} imei The product imei.
 */
async viewDevice(ctx, imei) {
    // Retrieve the product document from the data store based on its imei.
    const key = ctx.stub.createCompositeKey('PROD', [imei]);
    const productAsBytes = await ctx.stub.getState(key);

    // Check whether the product exists.
    if (!productAsBytes || productAsBytes.length === 0) {
        throw new Error(`${key} does not exist`);
    }

    // Return the product information.
    return productAsBytes.toString();
}
```

Figure 6: Chaincode function to trace the origin of a device.

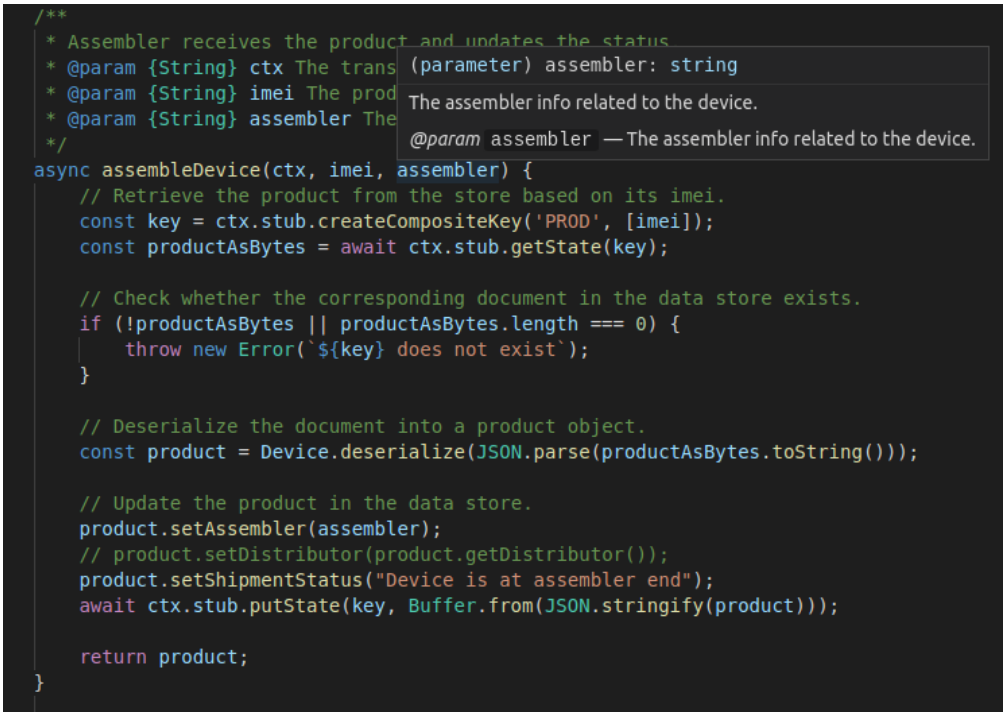


Figure 7: Chaincode function to assemble a device.

```
/**
 * Distributor receives the device and updates the status
 * @param {String} ctx The transaction context.
 * @param {String} imei The product imei.
 * @param {String} distributor The distributor info related to the device.
 */
async distributeDevice(ctx, imei, distributor) {
    // Retrieve the product from the store based on its imei.
    const key = ctx.stub.createCompositeKey('PROD', [imei]);
    const productAsBytes = await ctx.stub.getState(key);

    // Check whether the corresponding document in the data store exists.
    if (!productAsBytes || productAsBytes.length === 0) {
        throw new Error(`${key} does not exist`);
    }

    // Deserialize the document into a product object.
    const product = Device.deserialize(JSON.parse(productAsBytes.toString()));

    // Update the product in the data store.
    product.setDistributor(distributor);
    // product.setAssembler(product.getAssembler());
    product.setShipmentStatus("Device is at distributor end");
    await ctx.stub.putState(key, Buffer.from(JSON.stringify(product)));

    return product;
}
```

Figure 8: Chaincode function for distributor.

```
/**
 * View all unsold products in the store.
 * @param {String} ctx The transaction context.
 */
async viewUnsoldDevices(ctx) {
    // Retrieve all products stored in the data store.
    const results = [];
    for await (const result of ctx.stub.getStateByPartialCompositeKey('PROD', [])) {
        const strValue = Buffer.from(result.value).toString('utf8');
        try {
            let product = Device.deserialize(JSON.parse(strValue));

            // Only include those products that haven't been bought yet.
            if (!product.getIsBought()) {
                results.push(product);
            }
        } catch (error) {
            throw error;
        }
    }

    return results;
}
```

Figure 9: Chaincode function to check the unsold devices.

```
 * Buy a product from the store. The product must exist in the store first
 * and be unbought.
 * @param {String} ctx The transaction context.
 * @param {String} imei The product imei.
 * @param {String} newOwner The new description for the product.
 */
async buyDevice(ctx, imei, newOwner) {
    // Retrieve the product from the store based on its imei and name.
    const key = ctx.stub.createCompositeKey('PROD', [imei]);
    const productAsBytes = await ctx.stub.getState(key);

    // Check whether the corresponding document in the data store exists.
    if (!productAsBytes || productAsBytes.length === 0) {
        throw new Error(`${key} does not exist`);
    }

    // Deserialize the document into a product object.
    const product = Device.deserialize(JSON.parse(productAsBytes.toString()));

    // Check whether the product has already been bought.
    if (product.getIsBought()) {
        throw new Error(`${key} is not available for purchase`);
    }

    // Update the product in the data store.
    product.setOwner(newOwner);
    product.setIsBought();
    product.setShipmentStatus("Device is at consumer end");
    await ctx.stub.putState(key, Buffer.from(JSON.stringify(product)));

    return product;
}
```

Figure 10: Chaincode function for consumer to buy a device.

4.5 Application Demonstration

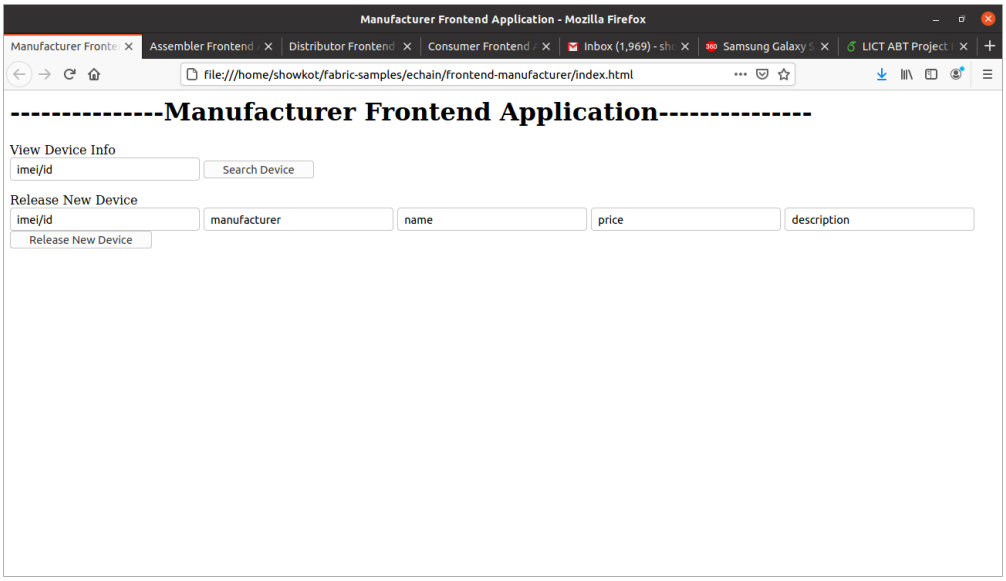


Figure 11: Frontend application for manufacturer.

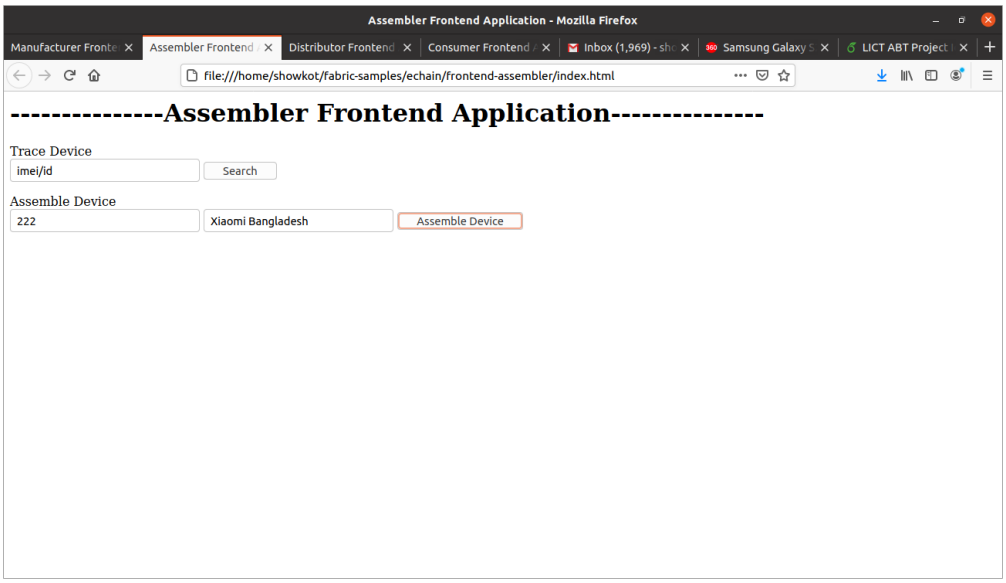


Figure 12: Frontend application for assembler.

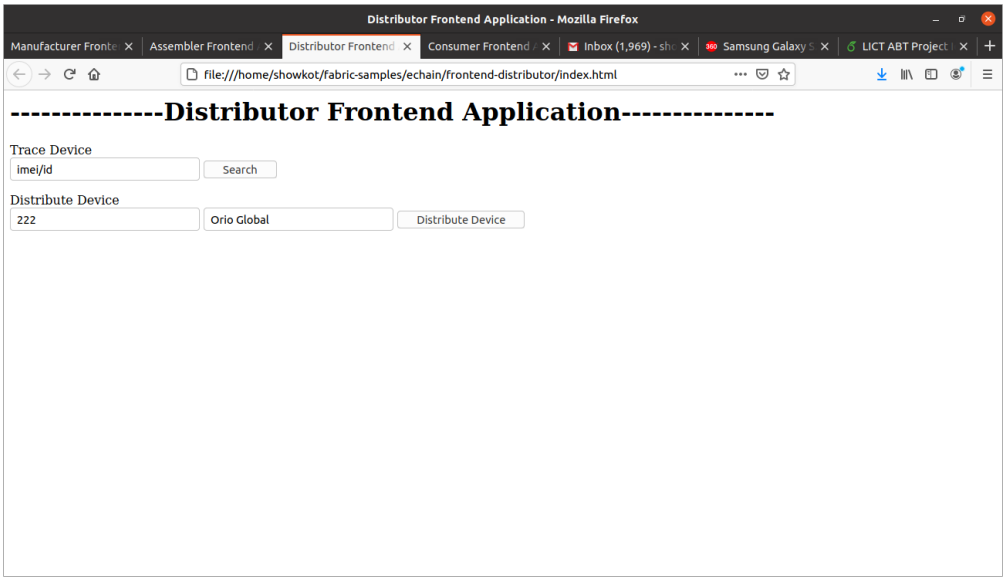


Figure 13: Frontend application for distributor.

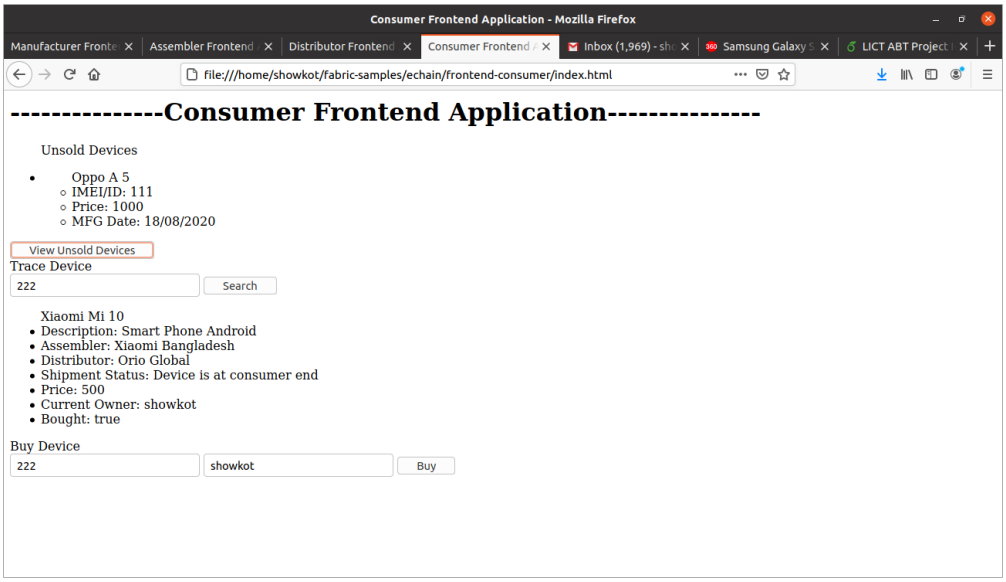


Figure 14: Frontend application for consumer.

5 Compliance and Legal Issues

The Echain network is composed of multiple participants that could span different countries or territories. This entire system is designed based on the requirements of each regulatory body in the network so that it can meet up with both local and international regulations. Our blockchain-based solution will further enhance corporate governance by providing information in real-time and seamlessly distributing data to the proper stakeholders. It will have a proper access control mechanism that hackers would not be able to breach. Only the people who have proper identification can enter the chain, and it will preserve the privacy of data based on the policies set by the members and the authorities within the entire supply chain. However, in this phase, I am developing a prototype as a part of the entire solution. This prototype system might not obey all the regulatory compliance.

6 Marketability & Partnership

In Bangladesh, there is no traceability system for electronic devices. Consumers usually have to worry about the authenticity of the devices because there are lots of fake/refurbished devices in the market. So this solution is definitely marketable. There might be a partnership with the government and tech giant companies.

7 Competition & Risks

To the best of my knowledge, there is no existing competitors in Bangladesh. Globally there are many supply chain traceability system implemented using hyperledger fabric and other platforms. These companies might be considered as competitors.

Blockchain being a new technology, many companies might not feel comfortable to adopt Echain giving up the existing old solution. But in course of time, companies will adopt blockchain because of versatile advantages of this technology. Lack of expertise and experience of deploying private blockchain platform may result in longer time to market this solution. Substantial upfront cost, due to on-premise deployment of hyperledger platform, may intimidate planned time to go live and make the system operational. Also at the beginning, partners might be reluctant to pay for the initial investment and it may result in longer time to market the solution.